ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Марчук Юлія Ярославівна

УДК 004.4

Розробка програмного забезпечення для цифрової підтримки діяльності кондитерського підприємства

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Марчук Ю.Я.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

к.т.н., доц.

_____Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

к.т.н., доц.

_____Слабінога М.О.

(підпис, дата, розшифрування підпису)

Івано–Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Марчук Юлія Ярославівна

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи:

Розробка програмного забезпечення для цифрової підтримки діяльності кондитерського підприємства

керівник роботи:

Слабінога Мар’ян Остапович, к.т.н, доцент

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: процес звітної та облікової діяльності на кондитерському підприємстві

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Аналіз предметної області. Пошук та аналіз аналогічних застосунків
2. Вибір методів та засобів моделювання застосунку, проєктування архітектури програмного забезпечення
3. Визначення структури даних, вибір технологій для розробки

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Пошук та аналіз предметної області	18.03.2024	Виконано
2.	Моделювання та проєктування ПЗ	29.03.2024	Виконано
3.	Визначення структури даних, вибір технологій для розробки	15.04.2024	Виконано
4.	Формування висновків	06.05.2024	Виконано
5.	Оформлення пояснювальної записки	15.05.2024	Виконано
6.	Оформлення графічного матеріалу та підготовка до захисту роботи	25.05.2024	Виконано

Студент _____ Марчук Ю.Я.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Слабінога М.О.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
15	CRM–система HugeProfit	55	Сторінка статистики користувача
29	Архітектура застосунку (версія головного кондитера)	57	Пропонована к–сть сировини для замовлення
30	Архітектура застосунку (версія адміністратор)	58	Сторінка "Денний звіт" секція "Випечена продукція"
31	Попередній вигляд UI–дизайну застосунку	59	Вікно додавання даних в звіт випеченої продукції
32	UI Kit застосунку	60	Вікно редагування випеченої продукції
50	Звіт випеченої продукції	61	Сторінка "Сировина", секція "Використана сировина"
50	Звіт використаної сировини	61	Сторінка "Сировина", секція "Поставка сировини"
51	Звіт залишку сировини	62	Сторінка "Зарплата"
51	Звіт відправленої продукції	63	Сторінка "Авторизація"
53	Бокове меню та головна сторінка		

АНОТАЦІЯ

Основною метою проєкту було створення програмного рішення для автоматизації обчислення заробітної плати та обліку залишків сировини на кондитерському підприємстві. Воно включає в себе також ведення електронного звіту та його генерацію для друку з метою оптимізації та підвищення ефективності роботи.

У першому розділі проведено детальний аналіз предметної області, включаючи опис процесів на кондитерському підприємстві, методи збирання інформації та аналіз аналогічних застосунків. Специфікація вимог до програмного забезпечення була розроблена на основі зібраних даних та аналізу аналогів, що дало змогу визначити основні функціональні та нефункціональні вимоги до системи.

Другий розділ включає проєктування системи. Тут описано методи та засоби моделювання, проєктування діаграм прецедентів, ER–діаграм, діаграм класів, а також архітектуру застосунку. Особлива увага приділена розробці UI–дизайну в Figma та вибору технологій для розробки, включаючи порівняння WPF і Windows Forms, вибір мови програмування та СКБД.

Третій розділ описує практичну реалізацію проєкту. Розроблено інформаційну структуру даних, створено таблиці бази даних та SQL–запити для базових функцій, а також розрахунків заробітної плати та залишку сировини. Також створено необхідні звіти, розроблено візуальну частину застосунку. Завершено валідацію та тестування застосунку для забезпечення його конкретної роботи та відповідності вимогам.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ, КОНДИТЕРСЬКЕ ПІДПРИЄМСТВО, ОБЧИСЛЕННЯ ЗАРОБІТНОЇ ПЛАТИ, ОБЛІК СИРОВИНИ, ЕЛЕКТРОННІ ЗВІТИ, БАЗИ ДАНИХ.

ABSTRACT

The project's main goal was to develop software for automating payroll calculations and inventory tracking at a confectionery enterprise. It also includes electronic reporting and printing to optimize and enhance workflow efficiency.

The first section involves a detailed analysis of the subject area, including descriptions of processes at the confectionery enterprise, methods of data collection, and analysis of analogous applications. Software requirements specification was developed based on gathered data and analysis of analogs, defining the system's main functional and non–functional requirements.

The second section encompasses system design. It describes methods and tools for modeling, designing use case diagrams, ER diagrams, class diagrams, and application architecture. Special attention is given to UI design in Figma and technology selection for development, including a comparison of WPF and Windows Forms, programming language choice, and DBMS.

The third section detailed the practical implementation of the project. It includes developing the data structure, creating database tables, SQL queries for basic functions, payroll, and inventory calculations. Additionally, necessary reports and the visual part of the application were developed. The application was validated and tested to ensure its functionality and compliance with requirements.

KEYWORDS: AUTOMATION, CONFECTIONERY ENTERPRISE, PAYROLL CALCULATIONS, INVENTORY TRACKING, ELECTRONIC REPORTING, DATABASES.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ..... 9
ВСТУП..... 10
РОЗДІЛ 1. ЗАГАЛЬНА ЧАСТИНА..... 12
1.1 Опис предметної області..... 12
1.2 Методи та засоби спостереження та збирання інформації..... 13
1.3 Аналіз аналогічних застосунків..... 14
1.4 Специфікація вимог до ПЗ..... 16
Висновки до розділу 1..... 19
РОЗДІЛ 2. ПРОЄКТНА ЧАСТИНА..... 20
2.1 Методи та засоби моделювання..... 20
2.2 Проєктування діаграми прецедентів..... 21
2.3 Проєктування ER–діаграми..... 23
2.3.1 Концептуальна модель даних..... 24
2.3.2 Логічна модель даних..... 26
2.3.3 Фізична модель даних..... 27
2.4 Проєктування діаграми класів..... 27
2.5 Архітектура застосунку..... 28
2.6 Розробка UI–дизайну в Figma..... 30
2.7. Вибір технологій для розробки..... 33
2.7.1 Вибір мови програмування..... 33
2.7.2 Технології розробки GUI. Порівняння WPF і Windows Forms..... 33
2.7.3 Вибір системи керування базами даних..... 34
Висновки до розділу 2..... 35

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА....36
3.1 Розробка інформаційної структури даних, створення таблиць бази даних.. 36
3.2 Створення SQL–запитів до таблиць бази даних....41
3.2.1 Додавання даних до таблиці....41
3.2.2 Видалення даних з таблиці....43
3.2.3 Оновлення/Редагування даних з таблиці.... 44
3.2.4 Розрахунок заробітної плати.... 46
3.2.5 Розрахунок залишку сировини за поточний місяць.... 47
3.3 Розробка необхідних звітів даної предметної області....48
3.4 Розробка візуальної частини ПЗ....52
3.4.1 Бокове меню та UserControls....52
3.4.2 Розробка статистики для користувача. Діаграми LiveCharts....54
3.4.3 Сторінка «Денний звіт».... 58
3.4.4 Сторінка «Сировина»....60
3.4.5 Сторінка «Зарплата»....62
3.4.6 Сторінка «Авторизація»....63
3.5 Валідація застосунку....64
3.6 Тестування застосунку....68
Висновки до розділу 3....69
ВИСНОВКИ....70
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ....71
ДОДАТКИ....73
Додаток А.... 73
Додаток Б....74
Додаток В.... 75
Додаток Г.1.... 76

Додаток Г.2..... 77

Додаток Д..... 79

Додаток Е.....80

Додаток Є.....81

Додаток Ж.....83

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД	–	База даних
ПЗ	–	Програмне забезпечення
ООП	–	Об’єктно–орієнтоване програмування
СКБД	–	Система керування базами даних
UML	–	Unified Modeling Language
GUI	–	Graphical User Interface
UI	–	User Interface
UX	–	User Experience
ER	–	Entity–Relationship
WPF	–	Windows Presentation Foundation
CRM	–	Customer Relationship Management
MVVM	–	Model – View – ViewModel
SQL	–	Structured Query Language

ВСТУП

Актуальність теми. Щодня, в кінці робочого дня, кондитери на підприємстві повинні заповнювати звіти про випечену та відправлену продукцію, а також використану сировину. Однак, виникає ряд проблем, що зменшують ефективність роботи працівників та призводять до великих втрат часу. Проблеми включають в себе:

- неструктурованість звіту – звіт не має чіткої структури, також присутні дані, які працівники більше не заповнюють, що ускладнює заповнення;
- загублені звіти – іноді через необачність працівників (кондитерів чи бухгалтерів) звіти можуть загубитись. Це призводить до необхідності створення копій, але працівники не мають змоги точно відтворити всю інформацію, яку вони вносили в денний звіт;
- обрахунок заробітної плати – працівники досить часто стикаються з обрахуванням зарплати на декілька сотень/тисяч гривень. Бухгалтери не вносять всі дані зі звіту до системи;
- втрата часу на «паперову роботу» – замість того, щоб зосередитись на виготовлені випічки, працівники–кондитери витрачають багато часу на заповнення звітів. Це суттєво впливає на продуктивність, а призводить до зниження якості виробництва;
- невідповідність даних при обчисленні залишку сировини – головні кондитери повинні точно розраховувати залишок сировини на підприємстві. Однак, невідповідність даних у звітах може призвести до неточних обчислень та недостовірних результатів.

Мета дослідження. Створити програмне забезпечення для працівників–кондитерів кондитерського підприємства, яке включатиме ведення електронного звіту та його друку, автоматизоване обчислення заробітної плати та залишку сировини.

Завдання дослідження. На основі обраної теми в роботі визначено завдання: а) аналіз та опис предметної області; б) пошук та аналіз аналогічного програмного забезпечення; в) розробка сучасного та зручного UI–дизайну; г) вибір технологій та інших програм для розробки застосунку; д) проведення валідації та тестування застосунку.

Методи дослідження. Як засіб спостереження, обрано метод дослідження користувачів, зокрема проведення інтерв'ю з працівниками кондитерського підприємства. Як метод збирання інформації, вирішено провести опитування, щоб дізнатися відгуки потенційних користувачів застосунку.

Наукова новизна одержаних результатів. Розроблене програмне забезпечення для цифрової підтримки діяльності кондитерського підприємства не має аналогів на ринку, що робить його інноваційним рішенням. Нові алгоритми обробки даних та архітектура забезпечують підвищену точність, швидкість обробки інформації та гнучкість системи.

Практичне значення одержаних результатів. Результатом кваліфікаційної роботи є застосунок для ведення електронного звіту кондитера, автоматизоване обчислення заробітної плати, залишку сировини за місяць, генерація та друк звіту, що дозволяє зручно та ефективно цифро візувати процеси обліку та звітування працівників підприємства.

Структура. Розділи – 3. Загальний обсяг основної частини 72 с. Список використаних джерел містить – 22 позиції.

РОЗДІЛ 1. ЗАГАЛЬНА ЧАСТИНА

1.1 Опис предметної області

Звіт є ключовим інструментом у роботі кондитера, відіграючи вирішальну роль у відстеженні виробничого процесу.

Він дозволяє контролювати виготовлену готову продукцію, використану сировину, продукцію, яка була відправлена у продаж, а також нарахування заробітної плати.

Відстежуючи ці ключові показники, кондитерське підприємство може оптимізувати свої процеси, забезпечити постійну якість продукції та підтримувати прибутковість.

В кінці кожного дня кондитер повинен зафіксувати в звіті такі дані:

– використана сировина за день – у звіті ведеться облік всіх продуктів, які використовуються для приготування кондитерських виробів. Продукти в себе включають: борошно, шоколад, цукор та інші інгредієнти. Це дозволяє переконатися, що виробничий процес відповідає стандартам якості;

– виготовлена готова продукція – у звіті відстежується загальна вага готової продукції (у кілограмах). Перш ніж записати дані про виготовлену продукцію, працівник повинен додати вагу одного найменування випеченої продукції, яку було виготовлено протягом дня, і тільки після записати у звіт;

– продажі – у звіті фіксуються продажі товарів, які можна продавати у власному магазині, а також в інших магазинах, куди надсилається певна кількість товарів. Включається продукція, яка була виготовлена протягом дня, так і та, що залишилася в залишку. Це дозволяє переконатися, що запасів достатньо для задоволення попиту, а також уникати перевиробництва;

– залишки продуктів – звіт також включає інформацію про продукцію, яка не була відправлена на магазини і залишилася на складі.

1.2 Методи та засоби спостереження та збирання інформації

Як засіб спостереження, обрано метод дослідження користувачів, зокрема, проведення інтерв'ю з працівниками кондитерського підприємства. Інтерв'ю є важливим інструментом для збору інформації про людей та їхній досвід. В даному випадку, співбесіда проводилась особисто, в подальшій співпраці спілкування проводилось онлайн. У контексті розробки застосунків, співбесіди особливо важливі для розуміння потреб і переваг цільової авдиторії. Проводячи інтерв'ю, можна краще зрозуміти завдання, які виконують користувачі, інструменти, які вони зараз використовують, і проблеми, з якими вони стикаються. Ця інформація може бути використана для розробки дизайну програми та забезпечення її відповідності потребам цільової авдиторії. Окрім надання цінної інформації, інтерв'ю також є ефективним способом налагодження стосунків із потенційним користувачами. Коли потенційні користувачі відчувають, що їхня думка цінується, вони, швидше за все, дадуть чесний відгук і залучать себе до процесу розробки. Тому важливо ставитися до інтерв'ю з максимальною повагою та переконатися, що учасники інтерв'ю відчувають себе комфортно, діляться своїм досвідом і думками.

Як метод збирання інформації, вирішено провести опитування, щоб дізнатися відгуки потенційних користувачів застосунку. Опитування користувачів – важливий метод збору інформації про потреби користувачів, уподобання, поведінку та ставлення до продукту. Результати опитувань можуть дати цінну інформацію про мотивацію, проблеми та потреби користувачів. Така інформація може інформувати про такі рішення як, чи зосередитися на додаванні нової функції чи вдосконаленні існуючих. Ці методи використано для збору розуміння та даних про поведінку користувачів, уподобання та проблемні точки, які можна використати для розробки та тестування.

Після опитування потенційних користувачів, тобто працівників кондитерського підприємства, було виявлено наступні проблемні точки:

- звіт не структурований – у таблиці для заповнення звіту включені дані, які працівники вже давно не заповнюють, їх варто вилучити;
- необхідність копії заповненого звіту – через необачність працівника, чи бухгалтерії звіт можуть загубити. Саме тому, звіт необхідно заповнювати повторно, а дані про випечену продукцію за всі дні неможливо пам’ятати;
- витік заробітної плати – через неструктурованість даних у звіті, а також недосвідченість бухгалтерів втрачаються дані про вироблену продукцію. Обрахунок зарплати на декілька сотень/тисяч гривень. Працівнику доводиться перераховувати щодня свою заробітну плату, щоб не бути обрахованим;
- відсутність інформації про всю випечену продукцію – працівники не мають можливості дізнатися дані про всю випечену продукцію протягом місяця, а саме скільки було відправлено на магазини, вироблено на весілля та персональні замовлення від клієнтів;
- проблема обчислення залишку сировини в кінці місяця – як і звіт про вироблену продукцію, звіт залишку сировини може загубитись. Недостовірні дані після обчислення залишку сировини, поява нестач.

1.3 Аналіз аналогічних застосунків

Починаючи розробку застосунку, важливо проаналізувати схожі програми, щоб краще зрозуміти конкуренцію та поточні тенденції ринку. Ця інформація може допомогти скерувати процес проєктування та розробки, та переконатися, що нова програма відповідає потребам цільової авдиторії, а також виділяється серед конкурентів. Аналіз подібних додатків також може надати цінну інформацію про поведінку користувачів, уподобання та проблеми. Розуміючи, що користувачу подобається, а що не подобається в подібних програмах, можна краще зрозуміти, на яких типах функцій і удосконаленнях зосередитися. Крім того, аналіз подібних додатків може допомогти виявити потенційні прогалини на ринку. Розуміючи типи додатків, які зараз існують, а також їхні сильні та

слабкі сторони, можна визначити області, де можуть бути можливості створити щось нове та інноваційне, що відповідає потребам цільової авдиторії.

При детальному аналізі аналогічних застосунків, не знайдено програм, які були б спеціально розроблені для кондитерів та мали таку ж саму специфіку як розроблюваний застосунок. Однак, в Україні існують декілька програм, які можуть використовуватися для фінансового та складського обліку. Для аналізу було вибрано одну з українських CRM–систем для товарного бізнеса HugeProfit (рис. 1.1). Ця система пропонує вирішення таких задач як простий облік товарів, фінансовий облік, мультивалютність, облік продажів, Telegram Bot, робота з клієнтами, канали продажів, Новою поштою та Укрпоштою [2].

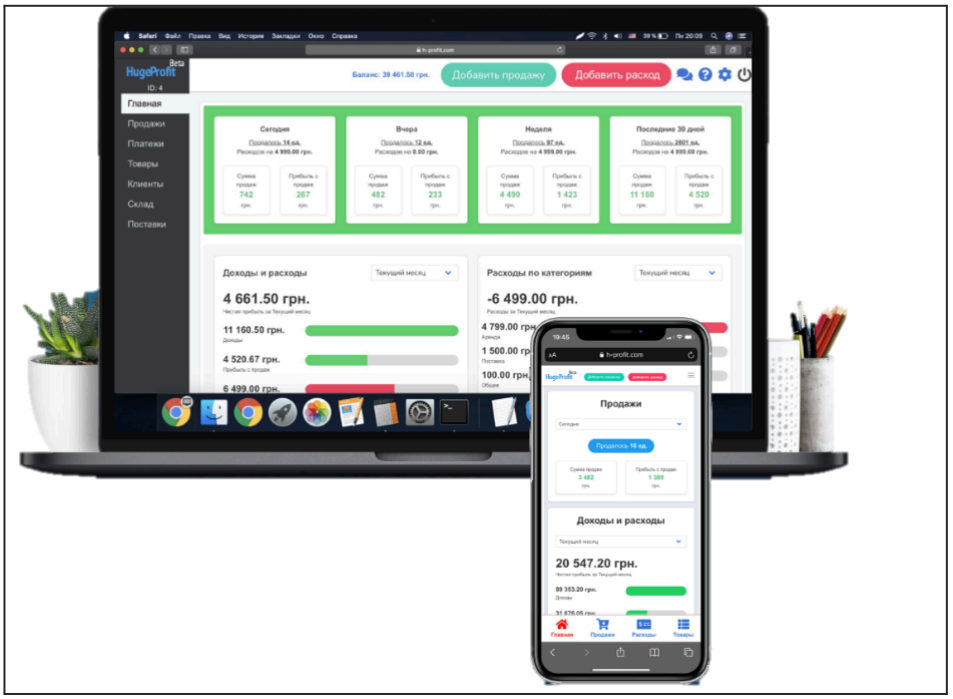


Рисунок 1.1 – CRM–система HugeProfit

Головна сторінка системи добре структурована, адже користувач відразу бачить дані щодо продаж за певний день, загальну статистику про доходи, витрати та витрати по категоріям.

Застосунок дає можливість відповісти на питання, які можна порівняти з застосунком що розробляється [2] :

- «Скільки чистого прибутку Ви отримали за вчора, сьогодні, місяць?» – працівники кондитерського підприємства також стикаються з проблемою обчислення заробітної плати, і їм доводиться щодня це рахувати;
- «Скільки і яких товарів у Вас залишилося на складі?» – кондитери не одразу відправляють на продаж всю випечену продукцію. Тому та продукція, яку не було відправлено на магазини, залишається в «залишку». Вести облік продукції в «залишку» також є необхідним;
- «Який канал продажів найприбутковіший?» – підприємство має як свої власні магазини, так і співпрацює з іншими магазинами, які замовляють випічку. Кількість замовлень від клієнтів різна. Отож, підприємству необхідна статистика, яка б дозволяла відповісти на дане питання.

Ведення денного звіту кондитера буде схожим до технології представлення даних у фінансових та складських програмах для обліку. Зберігається табличний вид представлення даних, друк звіту згідно норм підприємства, але зручний дизайн застосунку, який економить час працівнику для заповнення денного звіту та використання застосунку в цілому.

1.4 Специфікація вимог до ПЗ

Вимоги до програмного забезпечення визначають певні властивості, функції або можливості, якими повинна володіти система програмного забезпечення, щоб задовольняти потреби своїх користувачів. Вимоги до програмного забезпечення можуть бути функціональними або не функціональними. Функціональні вимоги описують, що має робити програмна система, наприклад, конкретні завдання, які вона повинна виконувати, необхідні вихідні дані, а також будь–які бізнес–правила, яких система повинна дотримуватися. Нефункціональні вимоги описують характеристики продуктивності системи, такі як надійність, безпека та масштабованість. Під час розробки ПЗ важливо визначити та задокументувати вимоги до системи,

щоб переконатися, що кінцевий продукт відповідає потребам і очікуванням зацікавлених сторін.

Після дослідження та аналізу предметної області було визначено наступні функціональні вимоги:

− зручний вхід в систему – кожному працівнику буде надано персональний код та пароль, за яким можна буде увійти в систему;

− перегляд інформації про заробітну плату зароблену за день, тиждень, місяць та рік – після того як працівник успішно авторизувався, перед ним з’являється вікно аналітики про діяльність працівника протягом місяця, а також заробітна плата яку він заробив протягом дня, тижня, місяця та року. Інформація в полях оновлюється після додавання випеченої продукції в звіт;

− додавання, видалення, редагування даних денного звіту – необхідно забезпечити базові функції для денного звіту, такі як додавання нових даних, видалення, та редагування у разі введення неправильних даних;

− додавання, видалення, редагування використаної сировини – так само як і денний звіт, звіт використаної сировини повинен включати базові операції;

− друк денного звіту за будь який день – для генерації звіту працівник повинен вибрати дату за яку йому потрібно отримати звіт. По замовчуванню в календарі застосунку буде сьогоднішня дата, але також потрібно передбачити вибір конкретної дати у разі необхідності повторного друку. Після підтвердження дії працівник отримає звіт та зможе його роздрукувати;

− друк використаної сировини за будь який день – для друку використаної сировини необхідно зробити всі ті самі дії, що і для друку денного звіту;

− друк залишку сировини за місяць – для друку залишку сировини за місяць необхідно додати можливість вибирати місяць, за який потрібно відобразити залишок сировини. Але по замовчуванню працівник буде бачити дані за поточний місяць та зможе згенерувати за мить;

− автоматизоване обчислення заробітної плати – працівнику не доведеться більше рахувати свою заробітну плату за день вручну. Система автоматично буде рахувати все та покаже точні дані;

− автоматизоване обчислення залишку сировини за місяць – обчислення залишку сировини буде обчислюватись автоматично. Але працівник на початку місяця після отримання сировини повинен внести дані про поставку сировини в систему. Наступні дії для обчислення сировини зробить система.

Вимоги до надійності та безпеки системи:

− обробка всіх помилок, які можуть призвести до аварійного завершення програмного забезпечення;

− валідація вхідних даних;

− доступ до системи має бути різний, в залежності від ролі користувача – в системі є три типи користувача: головний кондитер, кондитер та адміністратор. Кожен з них має свої завдання, тому інтерфейс буде відрізнятися;

− адміністративна панель має бути захищена від користувачів–кондитерів, доступ має мати тільки адміністратор – кондитери не мають права доступу до редагування даних щодо користувачів системи;

− додавання, редагування та видалення даних – в денному звіті, звіті використаної сировини, звіті відправленої продукції та даних підприємства повинно працювати без помилок, а користувач має отримати відповідь та результат в таблиці;

− генерація звіту користувачем має відбуватись миттєво;

− автоматичне обчислення заробітної плати повинно відбуватись без помилок та надати користувачу точні дані про зарплату за день, тиждень, місяць, рік;

− автоматичне обчислення залишку сировини повинно показувати точні дані, без відхилень.

Вимоги щодо дизайну інтерфейсу користувача включають:

− простий і сучасний дизайн – інтерфейс має бути візуально привабливим, простим у використанні та відповідати сучасним тенденціям дизайну;

− зручність та інтуїтивність – інтерфейс користувача має бути простим у навігації, а всі функції які мають бути видимими з першого погляду, а

користувачі повинні мати можливість швидко та з мінімальними зусиллями знайти те, що їм потрібно;

− продуктивність – дизайн повинен бути масштабованим і здатним обробляти великі обсяги даних і складну взаємодію з користувачем. Це включає такі міркування, як швидкість реагування, час завантаження.

Висновки до розділу 1

У першому розділі проведено аналіз предметної області, зокрема, було визначено, які дані кондитер на підприємстві повинен включати до свого щоденного звіту, також при цьому було визначено специфіку ведення звіту.

Дослідження потенційних користувачів застосунку включало проведення інтерв’ю та опитування, під час якого було визначено уподобання користувачів, проблемні точки, з якими вони стикаються працівники на роботі. Під час аналізу аналогічних застосунків не було знайдено жодного, який би відповідав специфіці даної області та був би розроблений для кондитерів. Тому для порівняння було обрану українську CRM–систему HugeProfit [2], в якій було виявлено декілька аспектів, які можуть бути корисними для розробки програмного забезпечення. Крім того, було затверджено функціональні вимоги до майбутнього застосунку, вимоги до дизайну системи, а також вимоги до безпеки та надійності системи.

РОЗДІЛ 2. ПРОЄКТНА ЧАСТИНА

2.1 Методи та засоби моделювання

Моделювання предметної області – важливий компонент розробки програмного забезпечення, який передбачає представлення характеристик визначеної предметної області, таких як сутності, їхні зв’язки та атрибути, у логічний і структурований спосіб. Це є особливо важливим в об’єктно–орієнтованому програмуванні, оскільки воно дозволяє визначати та описувати класи та об’єкти, які складають предметну область, а також їхні зв’язки та взаємодії. Моделювання є найкращим способом візуалізації дизайну та перевірки його відповідності вимогам перед початком кодування. Саме тому, для проєкту побудовано такі діаграми як, діаграма прецедентів, ER–діаграма та діаграма класів.

Побудову діаграм виконано за допомогою методу моделювання UML та програми Draw.io. UML (Unified Modeling Language) – один з найбільш часто використовуваних методів моделювання, спеціально розроблений для моделювання систем програмного забезпечення. UML забезпечує візуальне представлення структури, поведінки та взаємодії. Однією з ключових переваг використання UML у розробці проєкту є те, що це допомагає створити чітке та повне розуміння вимог, цілей та обмежень проєкту.

Діаграма прецедентів – це тип моделювання, який використовується для представлення залежностей і взаємодій між різними компонентами в системі. Вони допомагають визначати та моделювати дії та поведінку, які користувачі або зовнішні системи виконуватимуть у системі. Діаграми прецедентів також визначають взаємодію між системою та її користувачами, і їх можна використовувати для визначення та моделювання вимог системи [1].

ER–моделювання – це ще один тип моделювання, який зазвичай використовується в розробці програмного забезпечення. ER–моделювання

представляє сутності та їхні зв’язки у візуальний спосіб і є особливо корисним у проєктуванні бази даних, оскільки дозволяє зрозуміти як відбувається зв’язок між таблицями та стовпцями [3].

Діаграма класів – графічне представлення класів та їхніх зв’язків в об’єктно–орієнтованій програмній системі. Це важливий компонент проєктування та розробки програмного забезпечення. Використовується для моделювання структури системи, визначення поведінки кожного класу [5].

2.2 Проєктування діаграми прецедентів

Діаграма прецедентів (Use–Case Diagram) є важливим інструментом у розробці програмного забезпечення, який використовується для представлення взаємодії та функцій системи з її користувачами [1]. Це візуальна діаграма, яка допомагає зрозуміти функціональні вимоги системи з точки зору користувача.

У діаграмах прецедентів використовується декілька моделей, зокрема:

- об’єкти – представляють акторів або об’єкти прецедентів у системі; Актором вважається користувач, який взаємодіятиме з системою. А користувачем може бути людина, організація або іншим об’єктом чи системою;
- зв’язки – вони представляють взаємодію та зв’язок між різними сценаріями використання або учасниками системи;
- варіанти використання – в UML представляють функціональні вимоги системи з точки зору користувача.

У розроблювальній системі буде 3 актори: головний кондитер, кондитер, адміністратор (рис. А.1). Перший актор діаграми – це «Працівник», до якого, згідно з діаграмою, належать два типи користувачів: «кондитер» і «головний кондитер». Щоб уникнути дублювання зв’язків з варіантами використання, було об’єднано їх в одного актора «Працівник».

Для цих користувачів доступні наступні функції:

- увійти в систему – функція включає (include) підтвердження авторизації, а також відображення помилки авторизації (extend). Під час введення даних

користувач може допустити помилку, написавши неправильно персональний код або ввівши неправильний пароль. Тому необхідно повідомити користувача, що авторизацію не виконано;

− перевірити заробітну плату – працівнику нічого не потрібно обчислювати самостійно, щоб отримати дані про заробітну плату. Його завдання – коректно ввести дані в денний звіт. Система виконує всі обчислення і повинна показати користувачу точні дані;

− редагувати денний звіт, редагувати використану сировину та редагувати відправлену продукцію – кожен звіт містить однаковий функціонал: додати, редагувати та видалити дані з таблиці. Після успішного додавання, користувач повинен отримати повідомлення, про те що дані було додано до звіту (include) та повідомлення про помилку, якщо дані не було додані у звіт;

− переглянути рецепти – рецепт це ще один важливий інструмент, який необхідний кондитеру у роботі. Якщо щось не використовувати у своїй роботі довгий час, то можна і забути. Тому користувач має змогу знайти необхідний йому рецепт та переглянути, що потрібно для приготування;

− роздрукувати необхідний звіт – після заповнення звіту, користувачу необхідно згенерувати звіт, який потім потрібно віднести до бухгалтерії. Система передбачає наступні сценарії: успішно згенерувати (include) або надіслати користувачу повідомлення, що під час генерації звіту виникла помилка (extend).

Наступний актор – головний кондитер. Як було згадано вище, головний кондитер може робити все те саме, що і кондитер. Однак, він має декілька функції, які лише йому необхідно виконувати, а саме:

− редагувати дані виробництва – включає в себе редагування сировини та випічки, а також магазинів, з якими співпрацює підприємство. Підприємство може вносити зміни, а саме додати нову випічку в продаж, або вилучати, так і з сировиною – можна купляти нові продукти або більше не використовувати в роботі. Саме тому, всі зміни потрібно зробити і в системі. Співпраця з деякими магазинами може припинятися або можна знаходити нових партнерів;

- додавати дані про поставку сировини – щомісяця підприємство закуповує сировину. Головний кондитер повинен занести дані в систему про поставку, для того щоб надалі система автоматично обчислила залишок сировини.

Третій актор – адміністратор. Адміністратор матиме доступ до всіх даних підприємства, однак йому належить наступні варіанти використання:

- відстеження історії користувачів;
- редагування даних існуючих користувачів;
- редагувати дані про користувачів – адміністратор може як додавати нових користувачів, так і видаляти, а також редагувати дані про занесених користувачів в системі.

2.3 Проєктування ER–діаграми

ER–діаграми (Entity–Relationship) – це графічне представлення сутностей та їхніх зв’язків у базі даних. Вони є важливим інструментом у проєктуванні бази даних і допомагають визначити структуру та організацію даних, а також зв’язки між різними об’єктами [3; 19].

Діаграми ER складаються з сутностей, які є будівельними блоками бази даних, і зв’язків між цими сутностями. Сутності представлені овалами, а зв’язки – лініями, що з’єднують сутності. На ER–діаграмі сутності описуються своїми атрибутами, які визначають властивості або характеристики сутності. Наприклад, у БД сутність працівника має атрибути такі як, ім’я, прізвище, номер телефону, електронна пошта, місце проживання. ER–діаграми корисні для візуалізації структури бази даних і виявлення потенційних проблем на ранніх етапах процесу проєктування.

Загалом, ER–діаграми є важливою частиною розробки бази даних, оскільки вони допомагають переконатися, що база даних є добре структурованою, ефективною та відповідає потребам користувачів.

2.3.1 Концептуальна модель даних

Концептуальна модель даних відноситься до високорівневого представлення даних, які мають зберігатися в базі даних або сховищі даних. Це логічне представлення даних, яке показує зв'язок між різними об'єктами та те, як вони пов'язані один з одним. Це важливий крок у процесі керування даними, оскільки забезпечує чітке розуміння вимог до даних, допомагає переконатися, що дані правильно структуровані та організовані [3]. На розроблюваній концептуальній моделі даних визначено наступні сутності (рис. Б.1):

− сутність «Користувач» (user) містить наступні атрибути: id, повне ім'я, ім'я користувача, пароль, роль, день народження, місто, область, країна, вік, номер телефону, електронна пошта, поштовий код;

− сутність «Торти» (cakes) містить такі атрибути як id, найменування торта та ціна за кілограм. Оскільки нарахування заробітної плати кондитера залежить від виробітку – необхідно знати ціну торта за кілограм;

− сутність «Сировина» (raw) містить атрибути id, найменування сировини;

− сутність «Магазини» (shops) містить атрибути id, найменування магазину;

− сутність «Денний звіт» (daily report) містить такі атрибути як id, найменування торта, ціна за кг, вага торта (у кг), тип замовлення, кондитер, дата виготовлення, магазин, статус відправлення, зароблено за день. Денний звіт розділено на дві умовні таблиці: дані про випечену продукцію, і продукцію яку було відправлено у продаж. Дані про відправлену продукцію у продаж буде оновлено на сторінці збуту, тим самим змінювати статус «залишок» на «відправлено» і магазин, куди було відправлено продукцію;

− сутність «Тип замовлення» (type of order) містить атрибути id та найменування типу замовлення. Підприємство виконує три типи замовлень: співпраця з магазинами, весілля та персональні замовлення клієнтів, наприклад день народження або іншу святкову подію. Особливо ці дані важливі для обчислення статистики, яка необхідна підприємству для аналізу діяльності кондитерського підприємства;

− сутність «Поставка сировини» (obtaining of raw) містить атрибути id, найменування сировини, дата поставки. Для того щоб розробити автоматизоване обчислення залишку сировини, необхідно знати коли отримано поставку сировини, яку повинен заповнювати головний кондитер щомісяця;

− сутність «Звіт використаної сировини» (used raw report) містить такі атрибути як id, найменування сировини, використана сировини, дата використання, кондитер, кількість дана на місяць, загальна використана сировина, день поставки, день обчислення залишку, статус залишку.

Під час розробки концептуальної моделі даних визначено умовні зв’язки. Таблиці бази даних умовно залежать від інших таблиць, оскільки використовують їхні дані. Наприклад, таблиці «денний звіт» необхідні дані з таблиці «торти»: найменування тортів та їхня ціна за кілограм. Тому, концептуальна модель містить наступні зв’язки:

− таблиці «Користувач» і «Денний звіт» – зв'язок 1:М. Одному користувачу належать багато звітів. Кондитер повинен щодня записувати звіти, тому, в системі зберігаються всі звіти користувача;

− таблиці «Денний звіт» і «Торти» – зв’язок 1:М. Один денний звіт може містити багато випеченої продукції різного найменування;

− таблиці «Денний звіт» і «Магазини» – зв’язок 1:М. Один денний звіт може містити дані про відправку випічки в декілька магазинів. Згідно дослідження, підприємство співпрацює з кількома магазинами, тому як правило відправлення продукції відбувається щодня та від трьох до восьми магазинів;

− таблиці «Денний звіт» і «Тип замовлення» – зв’язок 1:М. В одному денному звіті можуть бути різні типи замовлення: на магазини, весілля, персональне замовлення;

− таблиці «Звіт використаної сировини» і «Сировина» – зв’язок 1:М. Звіт використаної сировини може містити різну сировину. Звіт умовно ділиться на дві таблиці: дані про використану сировину за день і обчислення залишку;

− таблиці «Звіт використаної сировини» і «Поставка сировини» – зв’язок 1:М. Звіт використаної сировини може включати різну сировину та її дані про поставку.

2.3.2 Логічна модель даних

Логічна модель даних – це тип моделі даних, яка забезпечує більш детальне представлення даних, які зберігатимуться в базі даних [3].

Це більш детальне представлення даних, ніж концептуальна модель даних, і містить інформацію про фізичне зберігання даних, наприклад структуру бази даних. Логічна модель даних зазвичай створюється з використанням різноманітних методів, у тому числі методів моделювання даних, таких як діаграми сутності та зв’язку, діаграми потоку даних і карти даних. Ці методи допомагають гарантувати, що логічна модель даних точно представляє вимоги до даних програми, а також допомагають забезпечити правильну структуру та організацію даних. Логічна модель даних важливим кроком у процесі керування даними, оскільки вона забезпечує детальне розуміння вимог до даних і допомагає переконатися, що дані правильно структуровані та організовані (рис. В.1).

Як було згадано вище у пункті 2.3.1, зв’язки у базі дані є умовними. Саме тому, під час розробки бази даних у середовищі для розробки БД, зв’язки між таблицями не будуть встановлені. Це означає, що хоча зв’язки між таблицями визначені на концептуальному рівні, вони не відображаються явно на рівні бази даних. Замість цього, вони використовуються для керування процесом вибору та маніпуляції даними.

Таким чином, логічна модель даних є важливим кроком у процесі керування даними, оскільки вона забезпечує більш детальне представлення даних, які зберігатимуться в базі даних або сховищі даних, і гарантує, що дані правильно структуровані та організовані.

2.3.3 Фізична модель даних

Фізична модель даних – це найнижчий рівень моделювання даних у базі даних або сховищі даних. Вона представляє базову структуру бази даних.

Дана модель – це рівень нижче логічної моделі в ієрархії базі даних, і вона надає детальне пояснення того, як дані фізично організовані та зберігаються в базі даних. Вона містить такі деталі, як типи даних, які використовуються для зберігання даних, розмір даних, формат даних. Дана модель також використовується, щоб гарантувати, що система бази даних оптимізована для продуктивності, та вона здатна обробляти потреби програми. Фізична модель даних є важливим аспектом процесу керування даними, оскільки вона допомагає переконатися, що дані належним чином зберігаються та витягуються з фізичного пристрою зберігання, а також допомагає мінімізувати ризик втрати або пошкодження даних. Використовуючи добре розроблену фізичну модель даних, можна створювати ефективні рішення для зберігання та отримання даних. У фізичній моделі даних таблиці використовуються для представлення сутностей, які складають систему , а стовпці використовуються для зберігання атрибутів цих сутностей. Створюючи фізичну модель даних, важливо враховувати такі фактори, як надлишковість даних і нормалізація, які можуть допомогти підвищити загальну ефективність бази даних. Також важливо створити чітку та стислу фізичну модель даних, щоб забезпечити її зручність для розуміння та підтримку.

Фізичну модель даних було згенеровано в середовищі розробки бази даних MySQL Workbench (рис. Г.1). Окрім того, наведено список всіх таблиць бази даних розроблюваного застосунку та їх атрибутів з описом (табл. Г.2.1).

2.4 Проєктування діаграми класів

Діаграма класів – це тип діаграми UML, яка представляє класи та їхні зв’язки в дизайні об’єктно–орієнтованого програмування (ООП). Діаграма

класів є ключовим інструментом для проєктування та планування проєктів програмного забезпечення ООП, оскільки вона дозволяє чітко та лаконічно відобразити зв’язки між різними класами та їхніми взаємодіями [5].

Ключові компоненти діаграми класів включають:

− класи – це схема створення об’єкта з певним набором атрибутів і поведінки. На діаграмі класів класи представлені у вигляді прямокутників із записаною назвою всередині. Назва класу зазвичай відповідає реальній сутності, яку представляє клас;

− атрибути – це властивість або характеристика класу. На діаграмі класів атрибути представлені як поля всередині прямокутника класу. Ім’я атрибута зазвичай відповідає властивостям реального світу представленої сутності;

− методи – це поведінка або дія, яку може виконувати клас. Назва методу зазвичай відповідає реальній дії, яку може виконувати клас.

Слід включити те, що під час розробки застосунок містить набагато більше класів різного типу (рис. Д.1). Дана діаграма класів показує які методи повинні бути включені в розробку для роботи застосунку. Ця діаграма відіграє важливу роль, допомагаючи зрозуміти структуру програми та те, як різні елементи повинні взаємодіяти один з одним.

Зв’язки, що зображені на рисунку є уявними, про які детальніше було описано у пункті 2.3.1 . Такі класи як Cake (торт), Shop/Client (магазин/клієнт) , Raw (сировина), та OrderType (тип замовлення) необхідні для зберігання даних для створення подальших списків, які потім потрібно використовувати у ComboBox (випадаючий список).

2.5 Архітектура застосунку

Архітектура програми – це структура та організація програми, яка включає різні компоненти програми та спосіб їх взаємодії один з одним. Це важливий аспект розробки програмного забезпечення, оскільки він впливає на загальну продуктивність, масштабованість і зручність обслуговування

програми. Добре розроблена архітектура програми допомагає забезпечити ефективну взаємодію різних компонентів програми (рис. 2.1).

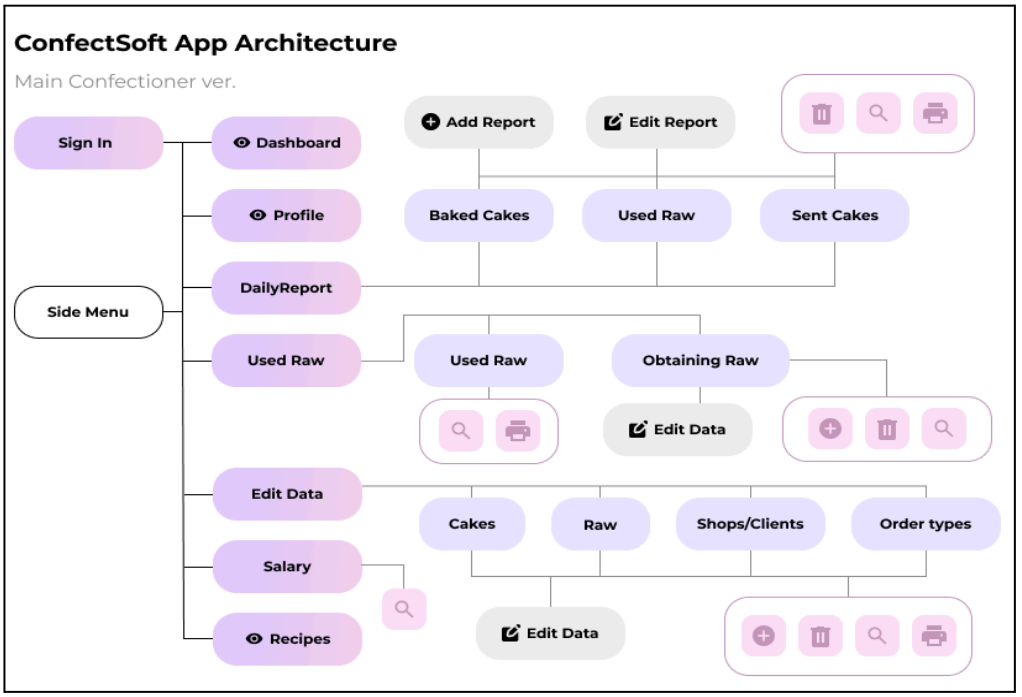


Рисунок 2.1 – Архітектура застосунку (версія головного кондитера)

Версія для звичайного кондитера не включає такі дані, як «Редагувати дані» та «Залишок сировини». Це ті обов'язки які повинен виконувати лише головний кондитер (рис.2.1). Архітектура застосунку включає наступні компоненти (рис.2.2) :

- головні сторінки в застосунку – «Main Page». Тобто це профіль (Profile), головна (Dashboard), денний звіт (Daily Report), редагувати дані (Edit Data), використана сировина (Used Raw), зарплата (Salary), рецепти (Recipes).
- секції – «Sections», які розроблені за принципом tab menu. Кожна секція містить сторінки, які включають певну інформацію, таблиці, поля чи дії.
- дії – «Actions». Для того щоб, не дублювати інформацію на схемі, було додано значки, які позначають певну дію. Це допоможе зробити більш читабельним рисунок. Тим самим, дія не потребує додаткового вікна.
- Однак, є певні сторінки які крім назви включають себе знак–дію. На рисунках можна побачити біля назви знак «ока» (рис. 2.1 та рис. 2.2). Це

означає, що дані сторінки призначені тільки для перегляду інформації, інші дії недоступні. У свою чергу, є сторінки, які позначені з знаком додавання чи редагування. Це означає, що це додаткове вікно, яке буде відкриватися поверх головного, і воно призначене для редагування чи додавання.

– Архітектура вікна для адміністратора містить сторінки які обмежені для перегляду працівникам кондитерського підприємства (рис. 2.2). Це дані про користувачів, які можна редагувати, додавання нових та історія дій.

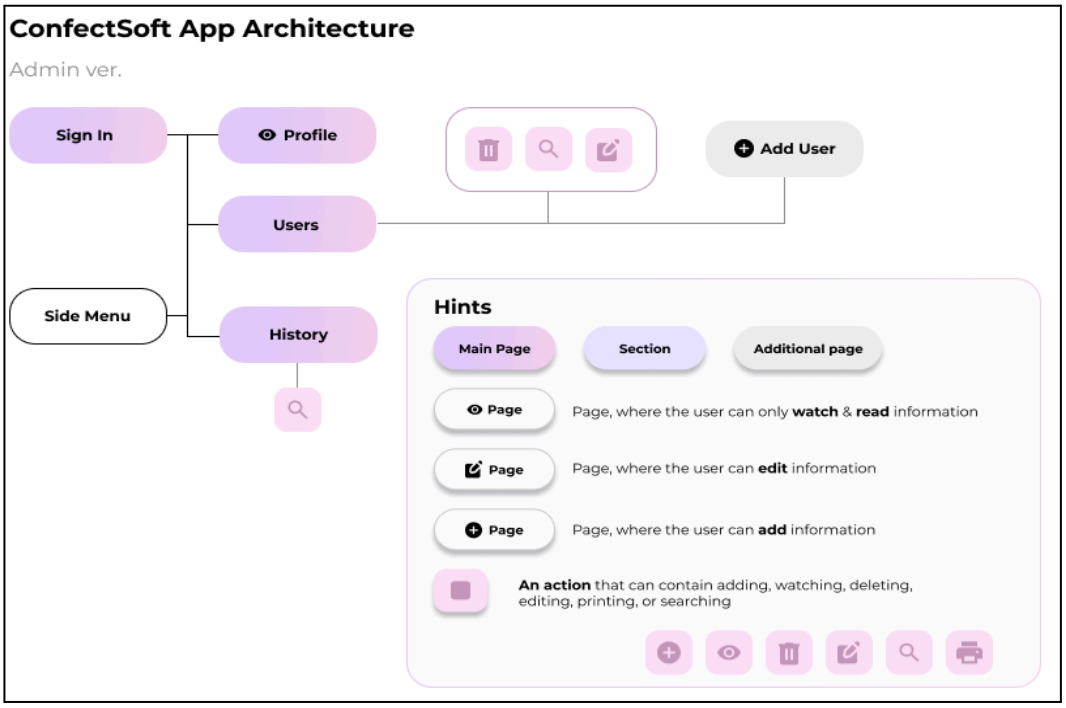


Рис 2.2 – Архітектура застосунку (версія адміністратора)

2.6 Розробка UI–дизайну в Figma

Дизайн інтерфейсу користувача (UI Design) є важливим аспектом у будь–якому проєкті розробки програмного забезпечення, оскільки він допомагає переконатися, що кінцевий продукт є зручним для користувача, візуально привабливим і відповідає потребам користувачів.

Дизайн інтерфейсу користувача передбачає створення візуального вигляду застосунку, включаючи макет, колірну схему, типографіку, та інші візуальні елементи, які взаємодіють з користувачем. У дизайні інтерфейсу користувача

метою є створення інтерфейсу, який є візуально привабливим, простим для розуміння та інтуїтивно зрозумілим у використанні, а також відповідає потребам кінцевих користувачів. Для проєктування дизайну інтерфейсу користувача обрано інструмент для проєктування інтерфейсу користувача – Figma [6;7;8]. За допомогою онлайн–редактору Figma було розроблено UI Kit, в якому відображено необхідні компоненти для створення дизайну застосунку, каркаси (Wireframes), які дозволили візуалізувати структуру застосунку перед детальним проєктуванням та UI–дизайн – дизайн для розробки майбутнього застосунку. Для UI–дизайну застосунку було обрано ніжну палітру та водночас сучасний дизайн, який не відволікав би користувача від його перш початкової мети користування застосунком (рис. 2.3).

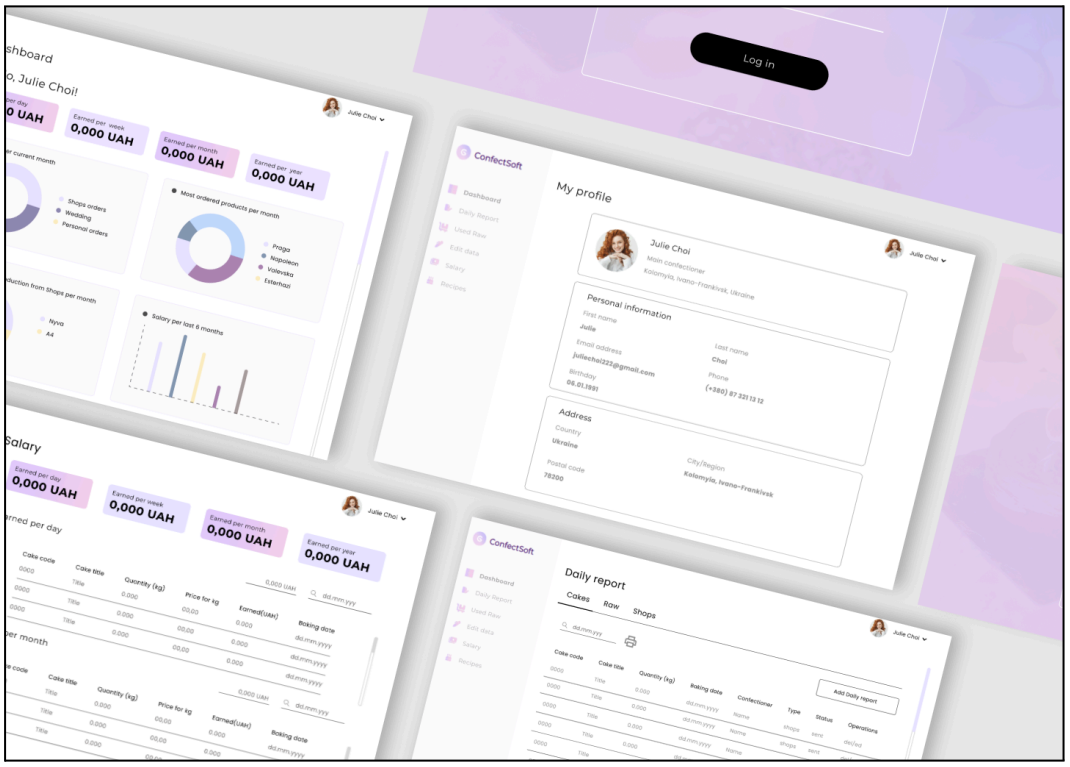


Рис 2.3 – Попередній вигляд UI–дизайну застосунку

Набір інтерфейсу користувача (UI Kit) – це набір попередньо розроблених компонентів інтерфейсу користувача, які можна використовувати для швидкого й ефективного створення інтерфейсів користувача [9]. В розроблений UI Kit

входить: палітра кольорів, шрифти, фотографії користувачів, логотип, іконки, кнопки (при наведенні, активна, неактивна) (рис. 2.4).

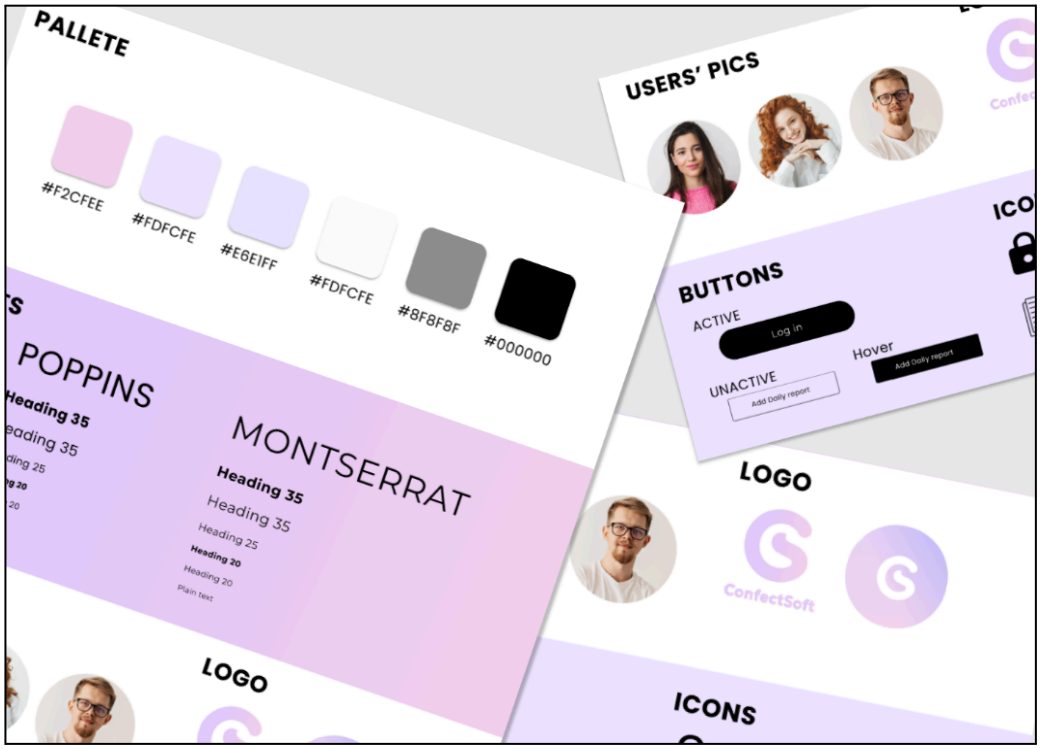


Рис 2.4 – UI Kit застосунку

Каркаси (Wireframe) – це тип дизайну інтерфейсу користувача, який зазвичай створюється на етапі попереднього проєктування, щоб допомогти дизайнерам і зацікавленим сторонам візуалізувати загальну структуру та макет інтерфейсу користувача перед переходом до детального етапу проєктування [7].

Створення вайрфрейму зазвичай виконується за допомогою інструменту вайрфрейму, який є програмним додатком, який дозволяє дизайнерам створювати прості візуальні проєкти для інтерфейсу користувача за допомогою простих об’єктів, таких як прямокутники, лінії та стрілки. Вайрфрейм є корисним інструментом на ранніх етапах процесу проєктування, оскільки він дозволяє створювати просте візуальне представлення програми, яке можна використовувати для тестування та вдосконалення інтерфейсу користувача перед переходом до детального проєктування та розробки (рис. Е.1).

2.7. Вибір технологій для розробки

2.7.1 Вибір мови програмування

Для розробки проєкту вибрано мову програмування C#. C# – популярна мова програмування, яка широко використовується для розробки різних типів додатків, від настільних до веб–додатків, мобільних додатків тощо. Вибираючи C# для свого проєкту, є кілька причин, чому це може бути правильним вибором:

- Однією з головних причин, чому C# є популярним вибором, є його високий рівень переносимості. Це можна використовувати на багатьох платформах, включаючи операційні системи Windows, Mac і Linux, а також мобільні операційні системи, такі як iOS і Android.
- Ще одна причина популярності C# полягає в тому, що вона пропонує високий рівень продуктивності. C# – це потужна мова, яка надає широкий спектр функцій і можливостей, що робить її ідеальним вибором для широкого кола програм. Наприклад, C# добре підходить для розробки програм корпоративного рівня, таких як ті, що використовуються для управління бізнесом, аналізу даних і звітності тощо.

2.7.2 Технології розробки GUI. Порівняння WPF і Windows Forms

При виборі технології для розробки GUI (Graphical User Interfaces) між WPF (Windows Presentation Foundation) та Windows Forms, було вибрано WPF.

WPF і Windows Forms – це дві популярні технології для створення GUI у програмах .NET. Хоча обидві технології широко використовуються, WPF вважається більш сучасним і потужним вибором для розробки настільних програм, тоді як Windows Forms є більш традиційною технологією, яка існує довше. Одна з головних відмінностей між WPF і Windows Forms полягає в тому, як вони обробляють поділ завдань між логікою програми та інтерфейсом користувача. WPF побудовано на архітектурі модель–вид–вид–модель (MVVM), що означає, що інтерфейс користувача відокремлено від логіки програми, і їх

можна розробляти незалежно. Це може полегшити тестування та підтримку програми з часом.

WPF також пропонує ряд додаткових функцій, недоступних у Windows Forms. Вони включають підтримку зв'язуючих даних, створення шаблонів і стилів. Ще одна перевага WPF полягає в тому, що він пропонує кращу продуктивність, ніж Windows Forms, оскільки йому не потрібно постійно перебудовувати інтерфейс користувача. Це може бути особливо важливо для програм, які відображають велику кількість даних, наприклад діаграми.

2.7.3 Вибір системи керування базами даних

При розробці застосунку вибір між реляційними та нереляційними базами даних є важливим рішенням. Обидва типи систем мають свої переваги та недоліки, і вибір залежить від конкретних потреб програми. Реляційна база даних – це система, яка зберігає дані в рядках і стовпцях і використовує структуровану мову запитів (SQL) для доступу та обробки даних. Реляційні бази даних розроблені для високої оптимізації для запитів і отримання даних із великих обсягів структурованих даних і зазвичай використовуються в традиційних бізнес–додатках, таких як бухгалтерський облік і системи управління взаємовідносинами з клієнтами (CRM). З іншого боку, нереляційна база даних – це система, яка не використовує табличну структуру для зберігання даних і не використовує SQL для запитів і отримання даних. Нереляційні бази даних розроблені для високої оптимізації для зберігання та отримання напівструктурованих або неструктурованих даних, таких як текст, зображення або мультимедійні файли. Нереляційні бази даних зазвичай використовуються в програмах, які вимагають зберігання даних у дуже гнучкому та універсальному форматі, таких як сховища даних і програми соціальних мереж.

Після порівняння реляційних і нереляційних баз даних, вирішено обрати реляційні бази даних для цієї роботи. Це рішення прийнято на основі того, що реляційні бази даних найкраще підходять для специфіки застосунку, а саме для

обробки структурованих даних і широкого використання в бізнес–додатках. Серед різних реляційних баз даних було обрано MySQL. MySQL є однією з найпопулярніших реляційних баз даних, яка відома своєю надійністю, продуктивністю та гнучкістю.

Висновки до розділу 2

У другому розділі детально розглядалися методи та засоби моделювання. Зокрема, було використано уніфіковану мову моделювання UML, розроблено діаграму прецедентів (Use Case Diagram), концептуальну та логічну моделі даних, а також діаграму класів. Для цього було використано середовище розробки діаграм Draw.io. Фізичну модель даних було згенеровано в середовищі MySQL Workbench.

Для створення архітектури застосунку, UI Kit–а, UI–дизайну та вайрфреймів використано популярний інструмент для проєктування інтерфейсів – Figma. Було вибрано мову програмування для розробки застосунку, технологію GUI та СКДБ, проведено порівняльний аналіз реляційних та нереляційних баз даних.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА

3.1 Розробка інформаційної структури даних, створення таблиць бази даних

Для розробки бази даних MySQL було використано середовище для проєктування баз даних MySQL Workbench [10;11;12]. За допомогою команди CREATE DATABASE було створено базу даних «sweet_bakery_app» для програмного забезпечення:

```
CREATE DATABASE sweet_bakery_app;
```

База даних включає наступні таблиці:

1. Таблиця «users» містить дані про користувачів. Вони включають як працівників, які працюють на кондитерському підприємстві, а також і адміністраторів, які будуть відповідати за коректну роботу застосунку та допомогу працівникам кондитерського підприємства.

Наступний код, демонструє створення таблиці користувачі (users):

```
CREATE TABLE users( user_id INT PRIMARY KEY AUTO_INCREMENT,
user_name VARCHAR(25), user_surname VARCHAR(25), username INT
CHECK(username >=100000 AND username <=999999), user_password
VARCHAR(8), user_role VARCHAR(25), user_age INT CHECK(user_age >=16 AND
user_age <=100), user_birthday DATE, user_email VARCHAR(80), user_phone
VARCHAR(50), user_country VARCHAR(35), user_city VARCHAR(35),
user_region VARCHAR(35), user_postal_code INT CHECK(user_postal_code
>=10000 AND user_postal_code <=99999)).
```

Таблиця має такі поля:

- user_id – ID користувача в системі;
- user_name – ім’я користувача, обмеження покладено в 25 символів;
- user_surname – прізвище користувача;

– username – персональний код користувача, необхідний для авторизації в систему. Також код складається з 6 цифр, тому під час створення таблиці додано перевірку на діапазон;

– user_password – пароль користувача, за допомогою якого він буде здійснювати вхід в систему;

– user_role – роль користувача. Наразі є три ролі в системі: головний кондитер, кондитер та адміністратор. Кожен з них має свої завдання та різні доступи до системи;

– user_age – вік користувача. Під час створення таблиці включається перевірка на діапазон який допустимий для віку, тобто від 16 до 100;

– user_birthday – день народження;

– user_email – електронна пошта користувача;

– user_phone – номер телефону;

– user_country – країна;

– user_city – місто;

– user_region – область;

– user_postal_code – поштовий код, до якого також було додано перевірку на кількість чисел. В даному випадку, довжина поштового коду залежить від українських поштових кодів.

2. Таблиця «cakes» – в таблиці зберігається список випічки, а також ціна за кілограм для кожної випічки. Містить такі поля:

– cake_id – ID випічки;

– cake_title – найменування;

– cake_price_kg – ціна за кілограм випічки.

В нижче представленому коді представлено код, що створює таблицю торти (cakes):

```
CREATE TABLE cakes(
cake_id INT PRIMARY KEY,
cake_title VARCHAR(35), cake_price_kg DOUBLE);
```

3. Таблиця «raw» – таблиця включає в себе список всієї сировини яка доступна на підприємстві та містить такі поля:

- raw_id – ID сировини;
- raw_title – назва сировини.

Щоб створити таблицю необхідно наступне:

```
CREATE TABLE raw(
raw_id INT PRIMARY KEY,
raw_title VARCHAR(35));
```

4. Таблиця «shops» – дані в таблиці включають клієнтів (магазини), з якими співпрацює підприємство. Потрібно для створення списку всієї клієнтської бази.

- shop_id – ID клієнта;
- shop_title – найменування магазину (клієнта).

Для створення таблиці магазини (shops) виконується наступний код:

```
CREATE TABLE shops(
shop_id INT PRIMARY KEY,
shop_title VARCHAR(35));
```

5. Таблиця «order_type» – типи замовлень. Необхідна для розробки статистики користувача в застосунку. Включає:

- order_type_id – ID типу замовлення;
- order_type_title – назва типу замовлення.

За допомогою наступної команди CREATE TABLE, створюється таблиця типи замовлень (order_type):

```
CREATE TABLE order_type(
order_type_id INT PRIMARY KEY,
order_type_title VARCHAR(35));
```

6. Таблиця «raw_obtaining» – поставка сировини. Таблиця складається з таких полів, як:

- raw_obtaining_id – ID який позначає порядковий номер у таблиці;
- raw_id – ID сировини;
- raw_title – найменування сировини;
- raw_quantity – кількість сировини, яка була поставлена;
- obtaining_date – дата поставки сировини;
- final_calculation_date – дата, до якої потрібно використати сировину.

Виконуючи наступний код, створюється таблиця поставка сировина (raw_obtaining):

```
CREATE TABLE raw_obtainiing(
raw_obtaining_id INT PRIMARY KEY,
raw_id INT,
raw_title VARCHAR(35),
raw_quantity DOUBLE,
obtaining_date DATE,
final_calculation_date DATE);
```

7. Таблиця «used_raw_report» або «Звіт використаної сировини» включає наступні поля:

- used_raw_id – ID використаної сировини;
- raw_id – ID сировини;
- raw_title – назва сировини;
- used_raw_quantity – кількість використаної сировини;
- used_raw_date – дата використання сировини;
- confectioner – кондитер, який використав сировину;
- raw_quantity_per_month – кількість сировини, яку було поставлено на даний місяць;
- obtaining_date – дата поставки;
- final_calculation_date – кінцева дата використання сировини;
- raw_remaining_quantity – залишок сировини після використання.

Використовуючи команду CREATE TABLE створюється таблиця звіт використаної сировини (used_raw_report):

```
CREATE TABLE used_raw_report(
used_raw_id INT PRIMARY KEY AUTO_INCREMENT,
raw_id INT, raw_title VARCHAR(35),
used_raw_quantity DOUBLE, used_raw_date  DATE,
confectioner VARCHAR(60),raw_quantity_per_month DOUBLE,
obtaining_date DATE,final_calculation_date DATE,
raw_remaining_quantity DOUBLE);
```

8. Таблиця «users_actions» представляє дані про дії користувачів. Таблиця включає такі поля:

- action_id – ID дії;
- action_title – назва дії;
- user_username – персональний номер користувача;
- user_fullname – прізвище та ім’я працівника;
- action_time – дата та час, коли було здійснено дію.

Створення таблиці історія користувачів (users_actions) виконує команда CREATE TABLE:

```
CREATE TABLE users_actions(
action_id INT PRIMARY KEY AUTO_INCREMENT,
action_title VARCHAR(1000),
users_username INT CHECK(users_username >=100000
AND users_username <=999999),user_fullname VARCHAR(65),
action_time DATETIME);
```

9. Таблиця «daily_report_cakes» або «Звіт випеченої продукції» включає поля:

- daily_report_id – ID денного звіту;
- cake_title – назва випічки;
- cake_price_kg – ціна за кілограм;
- quantity_kg – кількість яка була випечена;
- baking_date – дата випікання;

- order_type – тип замовлення;
- confectioner – кондитер, який випікав продукцію;
- cake_status – статус випічки. По замовчуванню статус випічки включає «Залишок». У випадку коли продукція буде відправлена, буде встановлено новий статус «Відправлено»;
- sent_to_shop – клієнт, який отримає замовлення. Дане поле необхідне тільки під час додавання відправленої продукції. Під час додавання випеченої продукції, клієнт не включається в звіт;
- date_sent – дата відправлення. Дата відправлення продукції включається так само лише в «Звіт відправленої продукції». Під час додавання випеченої продукції, поле буде недоступне;
- earned_per_day – зароблено за день. Поле, яке перш ніж додати дані до таблиці, буде обчислювати дані про зарплату за одиницю випічки.

Виконуючи наступний код, відбувається створення таблиці звіт випеченої продукції (daily_report_cakes):

```
CREATE TABLE daily_report_cakes(
daily_report_id INT PRIMARY KEY AUTO_INCREMENT,
cake_title VARCHAR(35),cake_price_kg DOUBLE,
quantity_kg DOUBLE,baking_date DATE,
order_type VARCHAR(35),confectioner VARCHAR(60),
cake_status VARCHAR(25),date_sent DATE,
sent_to_shop VARCHAR(35) DEFAULT 'Залишок готової продукції',
earned_per_day DOUBLE);
```

3.2 Створення SQL–запитів до таблиць бази даних

3.2.1 Додавання даних до таблиці

Для додавання всіх даних в таблиці використовується подія для кнопки, яка обробляє додавання та запит до необхідної користувачу таблиці з бази даних. В даному випадку додавання даних відбувається на базі таблиці «daily_report_cakes» (або «Звіт випеченої продукції») [10;12].

Спершу створюється новий екземпляр класу DB_Connect, який відповідає за підключення до бази даних [16]:

```
using (DB_Connect db = new DB_Connect())
```

Створюється рядок query, який містить SQL–запит, що включає команду INSERT для вставки даних в таблицю daily_report_cakes. Після створюється новий об’єкт класу MySqlCommand, який приймає SQL–запит і об’єкт підключення до бази даних:

```
string query = "INSERT INTO `daily_report_cakes`(`cake_title`,
`cake_price_kg`, `quantity_kg`,`baking_date`, " + "`order_type`,
`confectioner`, `earned_per_day`) " + "VALUES (@newCakeTitle,
@newCakePriceKg, @newBakedQuantity, @newBakingDate,
@newOrderType,"
+"@confectioner, @earner_per_day);";MySqlCommand command = new
MySqlCommand(query, db.getConnection());
```

В свою чергу, клас MySqlCommand використовується для виконання SQL–запитів до баз даних. Також додаються параметри до команди SQL:

```
command.Parameters.AddWithValue("@newCakeTitle", newCakeTitle);
command.Parameters.AddWithValue("@newCakePriceKg",
newCakePriceKg);
command.Parameters.AddWithValue("@newOrderType", newOrderType);
command.Parameters.AddWithValue("@confectioner", confectioner);
command.Parameters.AddWithValue("@earner_per_day", result);
```

Це не тільки дозволить покращити продуктивність, але і допоможе запобігти атакам SQL–ін’єкцій, оскільки вони автоматично обробляють вхідні дані таким чином, що не можуть змінити структуру SQL–запита. Потім відкривається підключення до бази даних. Виконується SQL–запит за допомогою методу ExecuteNonQuery(), який повертає кількість рядків, які були змінені SQL–запитом. Якщо метод ExecuteNonQuery() повертає 1, це означає,

що один рядок був успішно доданий, і відображається повідомлення про те, що дані було успішно додано в базу даних. Якщо ні, користувач отримує повідомлення про помилку. Після всіх операцій закривається підключення до бази даних:

```
db.openConnection();
if (command.ExecuteNonQuery() == 1)
{ MessageBox.Show("Дані успішно додано!");}
else {
MessageBox.Show("Помилка при додаванні даних!");}
db.closeConnection();
```

3.2.2 Видалення даних з таблиці

Перш ніж виконати запит видалення даних, необхідно отримати дані з рядка таблиці, який користувач хоче видалити. Саме тому, для того, щоб працювати з кнопкою видалення, яка знаходиться в таблиці біля кожного рядка з записом, необхідна змінна button. Рядок коду button = (Button)sender приводить об’єкт sender до типу Button, щоб можна було працювати з ним як з кнопкою. В наступному рядку виконується код DataRowView rowView = (DataRowView)button.DataContext:

```
Button button = (Button)sender;
DataRowView rowView = (DataRowView)button.DataContext;
```

Саме це дозволяє отримати дані, пов’язані з кнопкою. Це ті дані які користувач хочу видалити з таблиці. Після, змінна daily_report_id отримує ідентифікатор запису, який потрібно видалити. Створюється екземпляр класу DB_Connect, який безпосередньо відповідає за підключення до бази даних [16]:

```
int daily_report_id = Convert.
ToInt32(rowView["daily_report_id"].ToString());
DB_Connect db = new DB_Connect();
```

Також створюється новий об’єкт MySqlCommand, який приймає SQL–запит, що містить команду DELETE, яка відповідає за видалення даних з бази даних, а також об’єкт підключення до бази даних [10]:

```
MySqlCommand command = new
MySqlCommand("DELETE FROM `daily_report_cakes`
WHERE `daily_report_id` = @otid", db.getConnection());
```

Додається параметр до команди SQL, відкривається підключення до бази даних. Виконується SQL–запит за допомогою методу ExecuteNonQuery(), який повертає кількість рядків, що було змінено SQL–запитом:

```
command.Parameters.Add("@otid", MySqlDbType.VarChar).Value
= daily_report_id; db.openConnection();
if (command.ExecuteNonQuery() == 1){
MessageBox.Show("Запис було успішно видалено");}
else {
MessageBox.Show("Виникла помилка при видаленні запису");}
db.closeConnection();
```

Якщо метод повернув 1, це означає, що один рядок був успішно видалений з бази даних, і користувач отримує повідомлення «Запис було успішно видалено». Якщо ні, відображається повідомлення про помилку.

3.2.3 Оновлення/Редагування даних з таблиці

Як приклад оновлення даних в таблиці вибрано частину коду, що представлено нижче, який відповідає за оновлення даних в таблиці денного звіту, а саме випеченої продукції. Щоб оновити дані в таблиці базі даних виконуються наступні кроки [11;12]:

– створюється новий об’єкт класу DB_Connect, який відповідає за підключення до бази даних [16]. Після, використовується оператор using, який автоматично закриває підключення до бази даних після завершення блоку коду:

```
DB_Connect db = new DB_Connect(); using (db){
```

– далі створюється SQL–запит, який містить команду UPDATE для оновлення даних в базі даних. В даному випадку запит оновлює поля випеченої кількості (quantity_kg), дату випікання (baking_date) та скільки зароблено за одиницю випічки (earned_per_day):

```
string query = $"UPDATE `daily_report_cakes`
SET `quantity_kg`= @newCakeQuantity, " +
$"`baking_date`=@bakingDate,`earned_per_day`=
ROUND(`cake_price_kg`*@newCakeQuantity, 2) " +
"WHERE `daily_report_cakes`.`daily_report_id`
= @cakeId;";
```

– створюється новий об’єкт MySqlCommand, що приймає SQL–запит та об’єкт для підключення до бази даних. Додаються параметри до запиту, які підвищують продуктивність виконання запиту та запобігання SQL–ін’єкцій:

```
MySqlCommand command = new MySqlCommand
(query, db.getConnection());
command.Parameters.AddWithValue("@newTitle",
newCakeTitle);command.Parameters.AddWithValue("
@newCakeQuantity", newCakeQuantity);
command.Parameters.AddWithValue("@cakeId", cakeId);
command.Parameters.AddWithValue("@bakingDate",
formattedBakingDate);
```

– відкривається підключення до баз даних. Виконується запит за допомогою ExecuteNonQuery(), який повинен повернути кількість рядків, що було змінено SQL–запитом. Якщо метод ExecuteNonQuery() повернув одиницю, це означає, успішне оновлення даних в базі даних. Після, користувач отримує повідомлення про успішну зміну даних. Якщо ж метод не повернув нічого, буде відображено повідомлення про помилку. Закривається підключення до бази даних:

```
db.openConnection();
if (command.ExecuteNonQuery() == 1){
MessageBox.Show("Успішно змінено!");}
else{MessageBox.Show("Помилка!");}
db.closeConnection();
```

3.2.4 Розрахунок заробітної плати

Згідно з правилами клієнта на виробництві, зарплата залежить від кількості виробленої продукції працівником. Кожна випічка має свою ціну за кілограм. Після цього, ціну за кілограм помножити на ту кількість, яка була спечена (у кілограмах). Саме тому, після кожної доданої випеченої продукції, обчислюється зарплату за одиницю випеченої продукції. Для відображення даних про зарплату за день, тиждень, місяць чи рік необхідно підсумувати поле з таблиці, яке відповідає за зарплату, а також вказати діапазон для дати випікання (MONTH(), YEAR() чи YEARWEEK()), в залежності від умови обчислення заробітної плати.

Для відображення даних про зарплату, яка була зароблена за день, тиждень місяць чи рік використовується наступний код:

```
string confectioner = LoginViewModel.UserInfo; DB_Connect db
= new DB_Connect();MySqlCommand command = new MySqlCommand
("SELECT SUM(earned_per_day) FROM `daily_report_cakes` " +
"WHERE confectioner = @confectioner AND DATE(baking_date) =
CURDATE()", db.getConnection());command.
Parameters.Add("@confectioner", MySqlDbType.VarChar).Value =
confectioner; db.openConnection(); var result =
command.ExecuteScalar(); double totalEarned = 0;
if (result != DBNull.Value){totalEarned =
earnedPerDayText.Text = totalEarnedStr;
```

Спершу необхідно зробити підключення до бази даних та здійснити запит. Поле зарплати в базі даних є типом DOUBLE, тому після результату можна отримати великий результат, який буде незручно переглядати користувачу. В

цьому випадку, перед виведенням даних користувачу було використано функцію Math.Round(Convert.ToDouble(результат), кількість чисел після коми). У випадку коли дані про зарплату відсутні в поточний день, користувач отримує «0.00 UAH», що показує що дані на момент роботи користувача в системі відсутні. Дані про зарплату користувач отримує одразу, як буде додано нову випечену продукцію.

Окрім того, по запиту було додано параметр, що підвищить продуктивність виконання запиту, а також запобігти атакам SQL–ін’єкцій.

3.2.5 Розрахунок залишку сировини за поточний місяць

Розрахунок залишку сировини в застосунку включає безліч процесів, які повинні працювати коректно, перш ніж користувач отримує результат з точними даними про залишок. Перш за все, важливо щоб головний кондитер контролював дані про поставку сировини. Якщо головний кондитер не додав до системи дані про поставку конкретної сировини, то буде проблема не тільки з розрахунком залишку сировини, але і додаванням використаної сировини для денного звіту.

Процес розрахунку залишку сировини, в даному випадку за поточний місяць який представлено нижче в коді, включає:

1. Встановлення з’єднання з базою даних за допомогою класу DB_Connect [16]. Створюється SQL–запит для вибору даних про використання сировини за поточний місяць для даного користувача:

```
DB_Connect db = new DB_Connect(); MySqlCommand command = new
MySqlCommand ("SELECT raw_id, MAX(raw_title) as raw_title, " +
"SUM(used_raw_quantity) as total_used, " +
"MAX(raw_quantity_per_month) as raw_quantity_per_month, " +
"MAX(obtaining_date) as obtaining_date,
MAX(final_calculation_date) " + "as final_calculation_date FROM
used_raw_report " + "WHERE MONTH(obtaining_date) =
@currentMonth)AND " + "used_raw_date BETWEEN obtaining_date AND
final_calculation_date GROUP BY raw_id", db.getConnection());
```

2. Після виконується виконання SQL–запиту і читання результатів. Під час читання результатів він створює список об’єктів RawMaterial, кожен з яких буде містити інформацію про певну сировину. Для кожної сировини обчислює залишок, віднімаючи загальне використання від кількості на місяць, і встановлюється статус відповідно до залишку:

```
MySqlDataReader reader = command.ExecuteReader();
List<RawMaterial> materials = new List<RawMaterial>();
while (reader.Read()){double totalUsed =
Convert.ToDouble(reader["total_used"]);double quantityPerMonth =
Convert.ToDouble(reader["raw_quantity_per_month"]);
double remainder = Math.Round(quantityPerMonth – totalUsed,3);
```

3. На останньому кроці закривається з’єднання з базою даних та відбувається прив’язання списку матеріалів до таблиці remainingRawCurMonth:

```
materials.Add(new RawMaterial{
RawId = Convert.ToInt32(reader["raw_id"]),
RawTitle = reader["raw_title"].ToString(),
TotalUsed = Math.Round(totalUsed,3),
QuantityPerMonth = quantityPerMonth,
ObtainingDate = Convert.ToDateTime(reader["obtaining_date"]),
FinalCalculationDate = Convert.ToDateTime(
reader["final_calculation_date"]),Status = status,
Remainder = remainder});} db.closeConnection();
```

3.3 Розробка необхідних звітів даної предметної області

Як було згадано у розділі 1, звіт є невід’ємною частиною у роботі кондитера. В обов’язки кожного кондитера входить заповнювати дані у звіт про випечену продукцію, використану сировину та відправлену продукцію.

Для генерації звіту в застосунку, застосована подія, яка дозволяє генерувати необхідний кондитеру звіт та відправити на друк, або зберегти його в форматі PDF, чи використати інший спосіб з наявних.

Лістинг коду для генерації звіту кондитера знаходиться у додатку Є, він включає в собі наступні кроки:

1. Спершу відбувається виклик методу bindDailyReportTableForPrinting(). Даний метод необхідний для створення вигляду звіту про випечену продукцію. Тобто, якщо протягом дня кондитер випікає випічку одного найменування декілька разів, необхідно все додати та вивести дані без повторень. Згідно з специфікою діяльності підприємства, працівники повинні саме у такому вигляді подавати звіт бухгалтеру, однак працівник в застосунку всі записи бачить окремо. Саме тому, потреба у функції, що буде показувати дані у звіті згідно стандарту підприємства, зростає.

2. Перед тим як друкувати звіт, треба зробити перевірку, чи є дані для друку. Якщо даних немає, відображається повідомлення «Немає даних для друку» і метод завершується.

3. Всі дані отримуються з таблиці. Тому, перш ніж згенерувати звіт, обов’язково потрібно перевірити, чи дата за яку працівник хоче роздрукувати звіт, збігається з датою, яка знаходиться у полі для пошуку у календарі.

4. Якщо ж перевірка даних була успішною, створюється новий FlowDocument для друку з визначеними параметрами сторінки та шрифту. Також додається базова інформацію про дату та час генерації звіту, заголовок, ім’я кондитера та його посада.

5. Після цього, створюється нова таблиця з визначеними параметрами і дані з таблиці застосунку додаються до таблиці документу, а потім таблиця додається до FlowDocument, який було створено на початку.

6. Наступним кроком є створення нового PrintDialog. PrintDialog дозволяє відкрити діалогове вікно для друку, яке є стандартним від Microsoft Windows. Саме діалогове вікно для друку дозволить користувачу вибрати необхідний йому принтер чи інший вид друку, вказати діапазон сторінок які слід роздрукувати. Якщо користувач погодився з діалоговим вікном друку, документ надсилається на друк.

Звіт для друку має певну різницю від звіту, який бачить користувач в системі. Як було згадано вище, згідно з політикою клієнта на виробництві, працівники–кондитери записують у звіт загальну вагу випеченої продукції одного найменування. Тобто, якщо протягом дня працівник випік 15 разів випічку одного найменування випічки він повинен сам порахувати та записати повну кількість. Саме тому було створено запит, який буде автоматично порахувати загальну вагу випеченої продукції, і кондитеру не доведеться робити це власноруч. Кожен звіт є індивідуальним, тому для кожного звіту включено ім’я працівника та його посада (рис. 3.1).

Дата та час генерації звіту: 21.04.2024 19:19:56

Звіт випеченої продукції за 21 квітня 2024 р.

Кондитер: Таня Шевченко

Посада: Головний кондитер

Код	Випічка	Всього випечено	Ціна за кг	Дата випікання
43	Спартак	1,345	15,2	21.04.2024
47	Мохіто	5,241	19,55	21.04.2024
46	Наполеон	3,545	10,33	21.04.2024

Рисунок 3.1 – Звіт випеченої продукції

«Звіт використаної сировини», досить схожий до звіту випеченої продукції. Однак у цьому випадку, виконується метод який показує загальну кількість використаної сировини без повторень (рис. 3.2).

Дата та час генерації звіту: 21.04.2024 19:28:11

Звіт використаної сировини за 21 квітня 2024 р.

Кондитер: Таня Шевченко

Посада: Головний кондитер

Код сировини	Сировина	Всього використано	Дата використання
2000	Шоколад	13,565	21.04.2024
2001	Згущене молоко	30,33	21.04.2024
2003	Шоколадна крихта	2,222	21.04.2024

Рисунок 3.2 – Звіт використаної сировини

«Звіт відправленої продукції» включає загальну кількість продукції яку магазин замовляв, тип замовлення, самого клієнта та дату відправлення. Згідно з вимогами підприємства, кількість у звіті також записують у кг, а не у штуках (рис. 3.3). Деталі про кожну відправлену випічку заповнюють у накладній.

Дата та час генерації звіту: 28.04.2024 16:31:59

Звіт відправленої продукції за 21 квітня 2024 р.

Кондитер: Таня Шевченко

Посада: Головний кондитер

Випічка	К-сть (кг)	Тип замовлення	Клієнт	Дата відправлення
Наполеон	5,569	Весілля	А5	21.04.2024
Спартак	1,325	Весілля	Нива	21.04.2024
Наполеон	3,975	Магазини	А2	21.04.2024
Мохіто	1,235	Весілля	А2	21.04.2024
Мохіто	2,133	Персональне	А2	21.04.2024

Рисунок 3.3 – Звіт відправленої продукції

У свою чергу звіт «Залишок сировини», містить більше інформації, а саме: код та найменування сировини, скільки було всього використано за місяць, кількість яка була поставлена на місяць, дата поставки, дата фінального розрахунку, статус, та скільки залишилося сировини згідно результату обчислення (рис. 3.4). Якщо працівники вказували точну кількість використання сировини протягом місяця, система покаже точні дані, які потім можна звірити.

Дата та час генерації звіту: 28.04.2024 16:35:03

Залишок сировини за квітень 2024

Кондитер: Таня Шевченко

Посада: Головний кондитер

Код	Сировина	Всього використано	Кількість на місяць	Дата отримання	Дата фінального розрахунку	Статус	Залишок
2000	Шоколад	13,565	20,22	01.04.2024	30.04.2024	Добре, є залишок	6,655
2001	Згущене молоко	30,33	80,11	01.04.2024	29.04.2024	Добре, є залишок	49,78
2003	Шоколадна крихта	2,222	101,1	01.04.2024	30.04.2024	Добре, є залишок	98,878

Рисунок 3.4– Звіт залишку сировини

3.4 Розробка візуальної частини ПЗ

3.4.1 Бокове меню та UserControls

Для того, щоб уникнути дублювання коду та не прописувати щоразу у кожному вікні, бокове меню було створено в головному файлі, де прописано всі необхідні контейнери, такі як, бокове меню, панель заголовка, а також контейнер для вмісту кожної сторінки (рис. 3.5).

Для вмісту сторінок було використано тип файлу, як UserControl. UserControl використовується для створення самостійних, багаторазових використовуваних компонентів інтерфейсу користувача для вікна чи сторінки. Він може містити елементи керування, прив’язки даних і обробники подій, що полегшує розробку та підтримку складних інтерфейсів користувача. Також, User Controls можна вкладати один в одного. В головному файлі необхідно створити контейнер, який буде містити всі User Controls. Тому, під час розробки кожного UserControl, увагу зосереджено саме на інформації, яка повинна відображатись в нашому контейнері.

Для того, щоб переключати сторінки в боковому меню, слід підключити в головному файлі XAML, де знаходиться бокове меню та інші контейнери, які міститимуть User Controls, всі файли інших сторінок типу ViewModel [22]:

1. <Window.DataContext> – це контекст даних для вікна. Він вказує на модель представлення Dashboard_ViewModel_mainConfectioner, яка буде використовуватися для прив’язки даних у цьому вікні:

```
<Window.DataContext>
<viewModel:Dashboard_ViewModel_mainConfectioner/>
</Window.DataContext>
```

2. <Window.Resources> – це ресурси, які доступні для цього вікна. Вони включають набір шаблонів даних (DataTemplate), які визначають, як саме відобразити різні типи моделей представлення:

```
<Window.Resources> <DataTemplate DataType=
"{x:Type viewModel:DashboardDataViewModel}">
<local:DashboardDataView/> </DataTemplate>
```

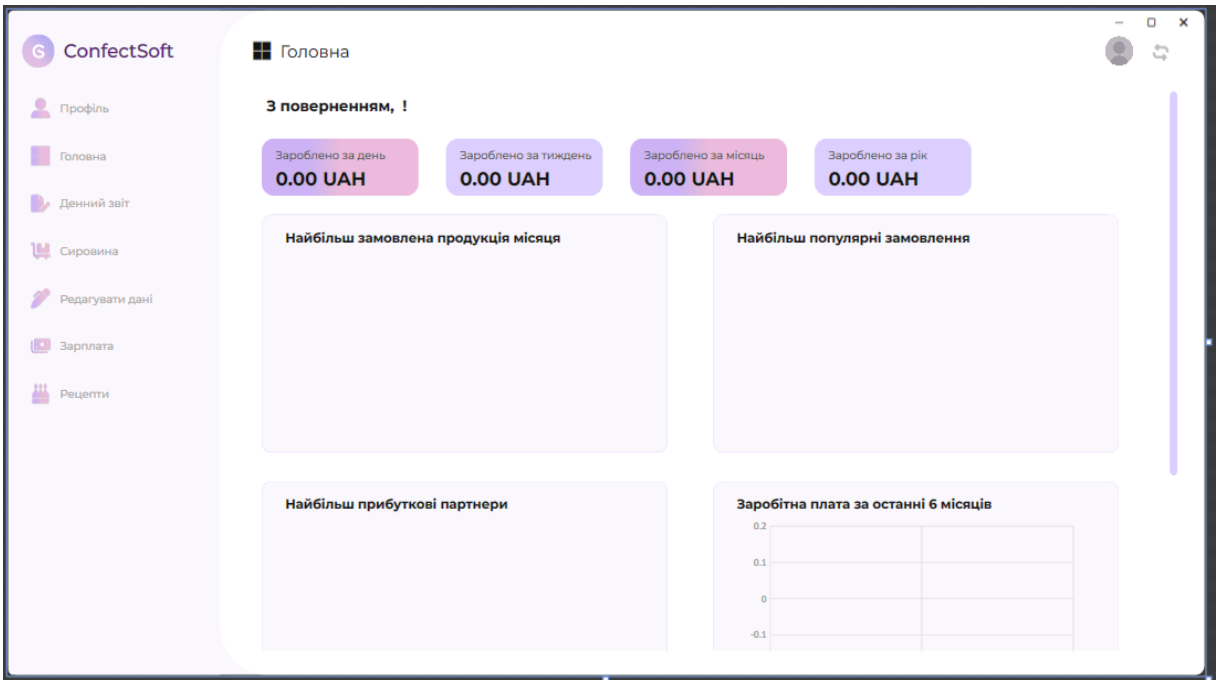


Рисунок 3.5 – Бокове меню та головна сторінка

Як було згадано вище, DataTemplate визначає, як відобразити певний тип моделі представлення. Наприклад, DasboardDataViewModel буде відображатись за допомогою DashboardDataView. Це означає, що коли користувач використовує DashboardDataViewModel у своєму коді, код автоматично використовує DashboardDataView для відображення цих даних.

Для написання логіки переключень між сторінками створюється файл, який знаходиться в папці ViewModel [22]. У файлі знаходиться клас Dashboard_ViewModel_mainConfectioner, який використовується для управління представленнями (views) у застосунку.

Файл містить ряд команд (ICommand), які використовуються для переключення між цими представленнями. Кожна команда пов’язана з відповідним методом (ExecuteShow…Command), яка встановлюватиме поточне представлення (CurrentChildView), заголовок (Caption) та іконку (Icon), які відображатимуться в інтерфейсі користувача:

```
public ICommand ShowDashboardDataViewCommand
{ get; } // Show "Dashboard" page
public ICommand ShowEditDataViewCommand
{ get; } // Show "Edit data" page
public ICommand ShowSalaryDataViewCommand
{ get; }// Show "Salary" page
```

Наприклад, якщо виконується команда ShowDashboardDataViewCommand, вона викликає метод ExecuteShowDashboardDataViewCommand. Цей метод, встановлює CurrentChildView, як новий екземпляр DashboardDataViewModel, Caption як «Головна», а іконка як «IconChar.Microsoft»:

```
public string Caption{
get {return _caption;}
set { _caption = value; OnPropertyChanged(nameof(Caption));}}
public IconChar Icon{
get { return _icon; }
set { _icon = value; OnPropertyChanged(nameof(Icon));}}
```

Таким чином, цей клас дозволяє динамічно змінювати представлення, яке відображається користувачу, в залежності від обраної команди. Це допоможе зберегти чистоту коду та легкість управління представленнями в застосунку.

3.4.2 Розробка статистики для користувача. Діаграми LiveCharts

Сторінка «Статистика користувача» або «Головна» поділена на три секції: дані про зарплату за день, тиждень, місяць та рік без деталей, діаграми та пропонована кількість сировини яку варто замовити на наступний місяць, посилаючись на дані, які користувач вносив протягом місяця про використану сировину. Для розробки діаграм у застосунку було використано бібліотеку LiveCharts.Wpf [17]. Дані оновлюються щомісяця. Діаграми відображають дані тільки за поточний місяць, а саме: «Найбільш замовлена продукція місяця», «Найбільш популярні замовлення» та «Найбільш прибуткові партнери». Тому,

починаючи з початку місяця користувач отримуватиме нові дані. Щоразу як користувач додавати нові дані в системі – дані в діаграмах також будуть оновлюватися (рис. 3.6).

Рисунок 3.6 – Сторінка статистики користувача

Щоб відобразити дані у вигляді діаграми було використано два типи діаграм з бібліотеки LiveCharts: стовпчаста (CartessianChart) та кругова (PieChart). Нижче представлений код, який дозволяє генерувати стовпчасту діаграму. Перш ніж відобразити дані в діаграму, було здійснено підключення до бази даних, після формування SQL–запиту, який знайде дані про зарплату за останні 6 місяців та виконання запиту і заповнення таблиці:

```
DB_Connect db = new DB_Connect();DataTable table = new DataTable();
MySqlDataAdapter adapter = new MySqlDataAdapter(); string query = $"SET lc_time_names = 'uk_UA'; " + $"SELECT DATE_FORMAT( baking_date,'%M %Y') as month, " + $"SUM(earned_per_day) as total_earned FROM daily_report_cakes " + $"WHERE baking_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH) " + $"AND confectioner =
```

```
'{confectioner}' " + $"GROUP BY month ORDER BY month;";
```

Після цих кроків відбувається створення діаграми. Код створює нову діаграму columnChart з використанням бібліотеки LiveCharts.Wpf [17]. Він також створює колекцію даних columnData і список міток labels. Код проходить через кожен рядок в таблиці table, а після додає дані до columnData і Lables, потім додає columnData до columnChart.Series. На останньому кроці діаграма додається до дочірнього елемента salaryForLastSixMonth:

```
CartesianChart columnChart = new CartesianChart()
{LegendLocation = LegendLocation.Bottom};
SeriesCollection columnData = new SeriesCollection();
List<string> labels = new List<string>();
int i = 0; foreach (DataRow row in table.Rows){string month =
row["month"].ToString();double totalEarned =Math.Round
(Convert.ToDouble(row["total_earned"]),3);columnData.Add(new
ColumnSeries { Values = new ChartValues<double> {Math.Round(
totalEarned,3)},Title = month, Fill = colors[i % colors.Count] });
labels.Add(month);i++;}columnChart.Series = columnData;
```

Кольори в діаграмі встановлюються по замовчуванню [17]. Тому при непротребі, це можна проігнорувати. Однак для створення естетичного дизайну діаграм, який буде підходити під стиль застосунку можна задати кольори самостійно, створивши:

```
List<Brush> colors = new List<Brush>{
new SolidColorBrush((Color)ColorConverter.
ConvertFromString("#DFE7FD")), // lavender (web)
new SolidColorBrush((Color)ColorConverter.
ConvertFromString("#FDE2E4")), // misty rose
new SolidColorBrush((Color)ColorConverter.
ConvertFromString("#EAE4E9")), // magnolia};
```

Наступний код використовується для отримання даних для третьої секції, а саме «Пропонована кількість сировини для замовлення на наступний місяць»

(рис. 3.7). Спершу необхідно визначити період часу, тобто змінні monthStart і monthEnd встановлюють початок і кінець поточного місяця:

```
string monthStart = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 1).ToString("yyyy-MM-dd");
string monthEnd = new DateTime(DateTime.Now.Year,
DateTime.Now.Month,
DateTime.DaysInMonth(DateTime.Now.Year,
DateTime.Now.Month)).ToString("yyyy-MM-dd");
```

Після створюється з'єднання з базою даних, формується SQL–запит, а потім виконується SQL–запит, створюється таблиця. На останньому кроці результати прив'язуються до usedRawReport_DailyReportForm.ItemsSource для відображення даних в таблиці:

```
DB_Connect db = new DB_Connect();
DataTable table = new DataTable();
MySqlDataAdapter adapter = new MySqlDataAdapter();
MySqlCommand command = new MySqlCommand($"SELECT `raw_title`, SUM
(used_raw_quantity) as raw_result " + $"FROM `used_raw_report`
WHERE `used_raw_date` " + $"BETWEEN '{monthStart}' AND
'{monthEnd}'
GROUP BY raw_title", db.getConnection());
adapter.SelectCommand = command;
adapter.Fill(table);
usedRawReport_DailyReportForm.ItemsSource = table.DefaultView;
```

Рисунок 3.7 – Пропонована кількість сировини для замовлення

3.4.3 Сторінка «Денний звіт»

Сторінка денний звіт поділена на три секції які також є UserControls. Перша секція «Випечена продукція» містить інформацію про випечену продукцію за весь час (рис. 3.8). Також можна виконати пошук та знайти необхідну інформацію, а після роздрукувати звіт. Доступні дії: редагування, видалення та додавання. Додавання та редагування даних здійснюється в іншому вікні Друга секція – «Використана сировина». Так само як і секція «Відправлена продукція» секція «Використана сировина» містить інформацію про використану сировину за весь час. Можна виконати пошук за необхідною користувачу датою, а після роздрукувати. Доступні дії: редагування, видалення та додавання. Додавання та редагування відправленої сировини здійснюється в іншому вікні. Третя секція – «Відправлена продукція». Секція включає дані про всю відправлену випічку за весь час. Серед доступних дій: додавання, видалення, друк звіт та пошук за датою відправлення.

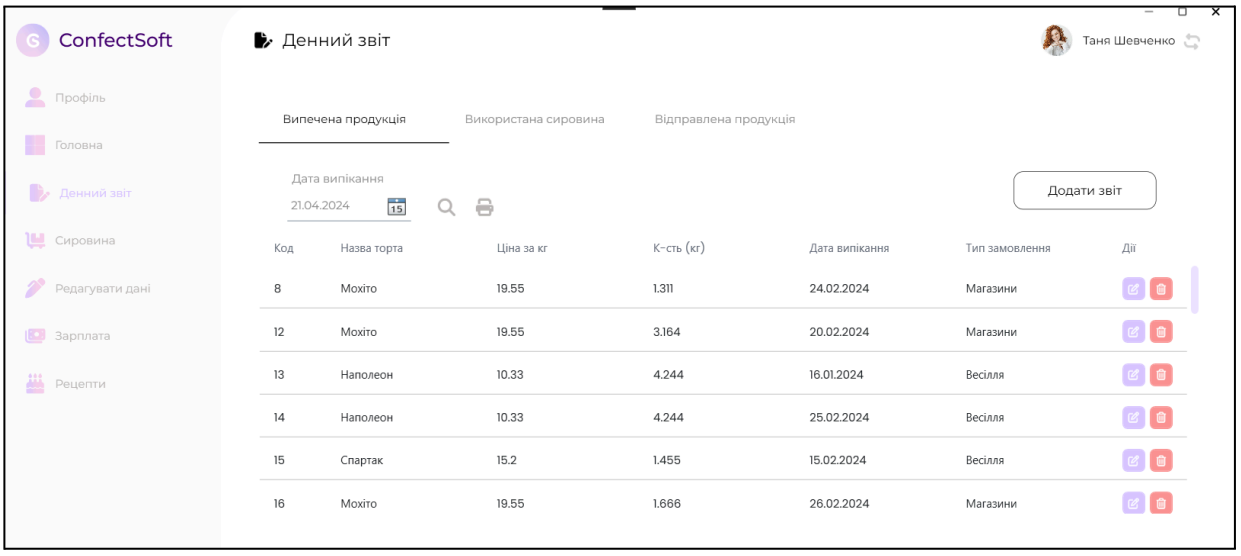


Рисунок 3.8 – "Сторінка Денний звіт", секція "Випечена продукція"

Після того, як користувач натиснув на кнопку «Додати звіт», відкривається вікно поверх головного, яке менше розміру. Вікно поділене на дві частини: меню, де користувач вводить всі необхідні йому дані про випечену продукцію, і таблиця, в якій можна переглянути дані, які були додані

користувачем за поточний день. Це також дозволяє видалити дані, у випадку якщо користувач ввів невірні дані.

Для додавання у меню користувачу необхідно: вибрати з списку назву випічки, ввести кількість у кілограмах, скільки важить торт, вибрати дату випікання (по замовчуванню календар показує сьогоднішню дату) та тип замовлення. Попередньо, кондитер має змогу переглянути ціну за кілограм кожного торта. Також в поле «Випечено (кг)» додано перевірку, яка дозволяє вводити тільки десяткові числа. Це забезпечить швидше додавання даних та уникнення небажаних помилок при додавання випеченої продукції (рис. 3.9).

У розділі 2 та 3 було згадано про те, що такі таблиці як, «Торти», «Сировина», «Магазини» та «Типи замовлення» необхідні для створення списків. Саме завдяки списку у ComboBox (або ж випадаючий список) можна додати список всієї випічки, яку продає підприємство, чи список всієї сировини, яка використовується на підприємстві, чи список всієї клієнтської бази, з якою безпосередньо співпрацює підприємство.

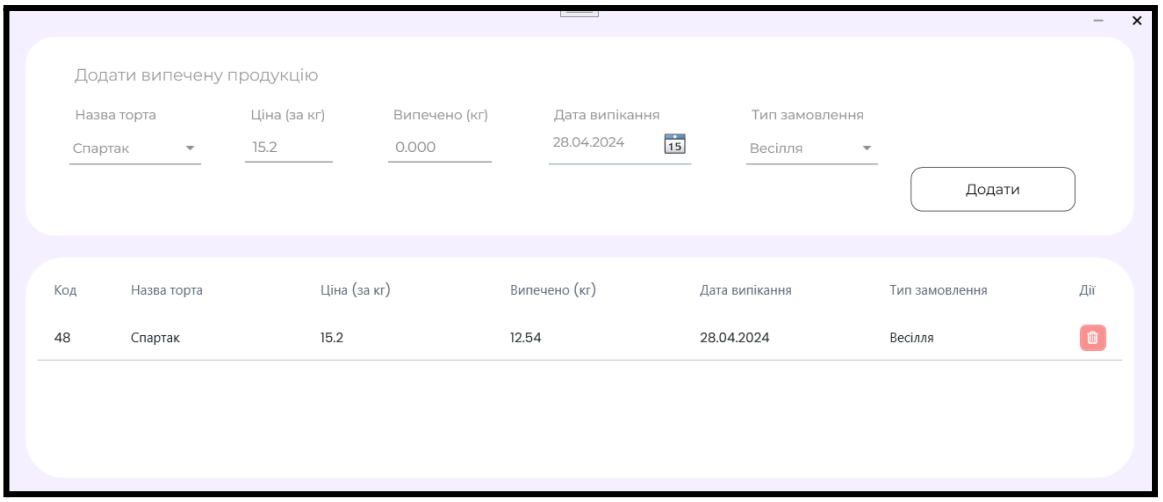


Рисунок 3.9 – "Вікно додавання даних в звіт випеченої продукції"

Якщо ж користувач, вирішив відредагувати дані з таблиці він отримує вікно (рис. 3.10). Однак для користувача є попередження, що в даному випадку можна редагувати випечену кількість та дату випікання. Такі поля, як «назва випічки», «код» та «тип замовлення» залишаються незмінними.

Рисунок 3.10 – "Вікно редагування випеченої продукції"

Такі секції як «Використана сировина» та «Відправлена продукція» містять такий самий інтерфейс та логіку як секція «Випечена продукція», однак різниця полягає в заповненні інших даних.

3.4.4 Сторінка «Сировина»

Сторінка «Сировина» містить дві секції: дані про залишок сировини за місяць, а також секція про поставку сировини (рис. 3.11).

Тобто головний кондитер має змогу переглянути список всієї використаної сировини протягом місяця, а також залишок що залишився і статус сировини. Статус залишку сировини може бути трьох видів:

1. «Добре, залишку не залишилось» – статус цей може бути у тому випадку якщо використали повністю сировину і тепер залишок дорівнює 0.

2. «Добре, є залишок» – даний статус буде показано тоді, коли залишок сировини не дорівнює 0 та більше 0.

3. «Недостача» – статус буде показано тоді коли кількість використаної сировини перевищила кількість сировини яку було поставлено на місяць.

Окрім статусу залишку сировини, в таблиці можна переглянути кількість залишеної сировини. Особливо це важливо для випадку 2 та 3. Також користувач має змогу роздрукувати звіт та при потребі звірити з кількістю сировини на виробництві. Якщо користувачі вносили дані правильно протягом місяця та не робили жодних непотрібних дій таблиця покаже точні дані.

Якщо працівнику необхідно знайти дані про залишок сировини за інший місяць, він може здійснити пошук, а після при необхідності роздрукувати.

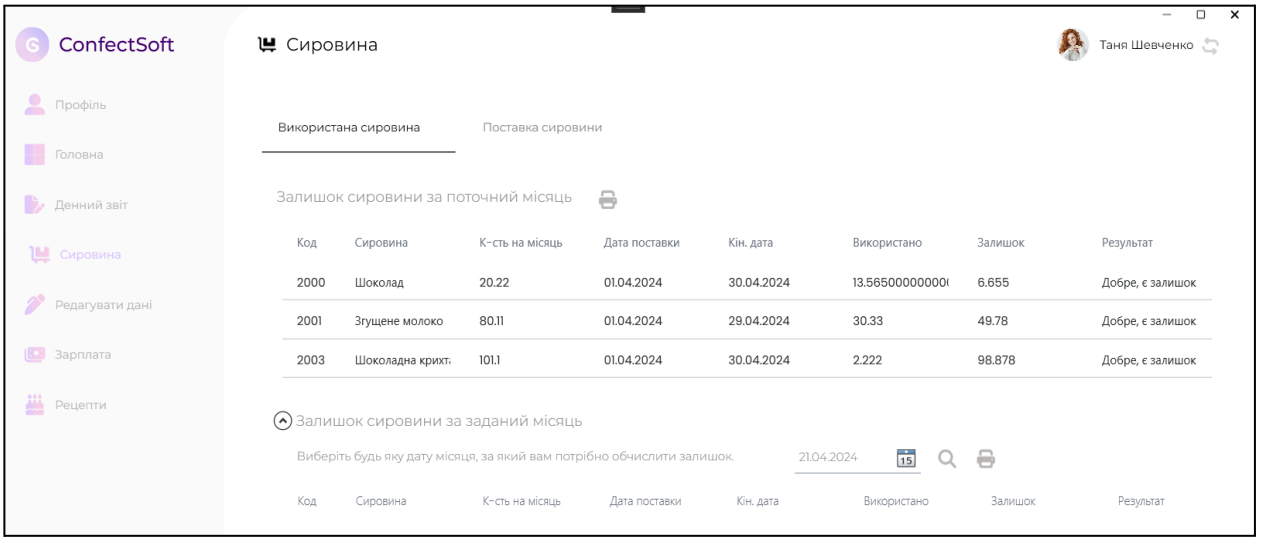


Рисунок 3.11 – Сторінка "Сировина", секція "Використана сировина"

Друга секція – це поставка сировини (рис. 3.12). Головний кондитер зобов’язаний щомісяця поновлювати дані в системі про поставку сировини. Якщо ж працівник не додав поставку сировини на цей місяць інші працівники не можуть вносити дані про те що вони використовують цю сировину та отримуватимуть повідомлення про помилку.

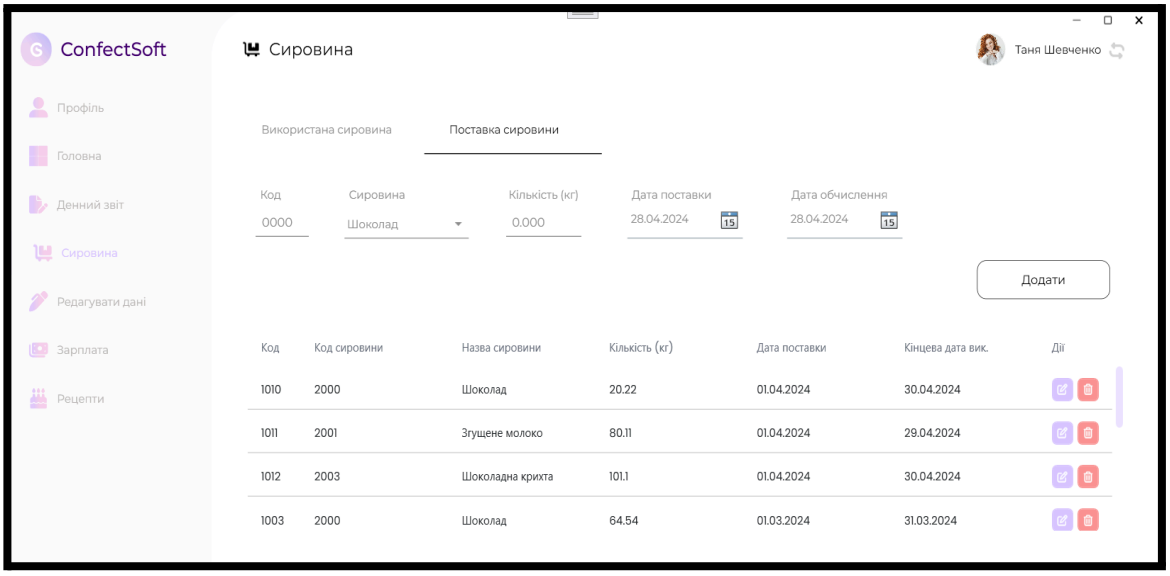


Рисунок 3.12 – Сторінка "Сировина", секція "Поставка сировини"

Під час додавання даних користувач повинен вказати дату поставки а також дату кінцевого використання. Дата кінцевого використання надзвичайно важлива, оскільки може бути сировина, яку не можна довго тримати на складі, тому її поставляють менше, і відповідно, дата використання менша.

Також бувають випадки повторної поставки, якщо сировину передчасно було використано. У такому разі повторної поставки сировини користувач повинен додати новий запис до системи, яка буде додана вже до існуючої.

3.4.5 Сторінка «Зарплата»

Вікно сторінки “Зарплата” містить дані про заробітну плату за поточний день, тиждень, місяць та рік (рис. 3.13). Також користувач має змогу здійснити пошук та знайти заробітну плату за потрібний йому день чи місяць.

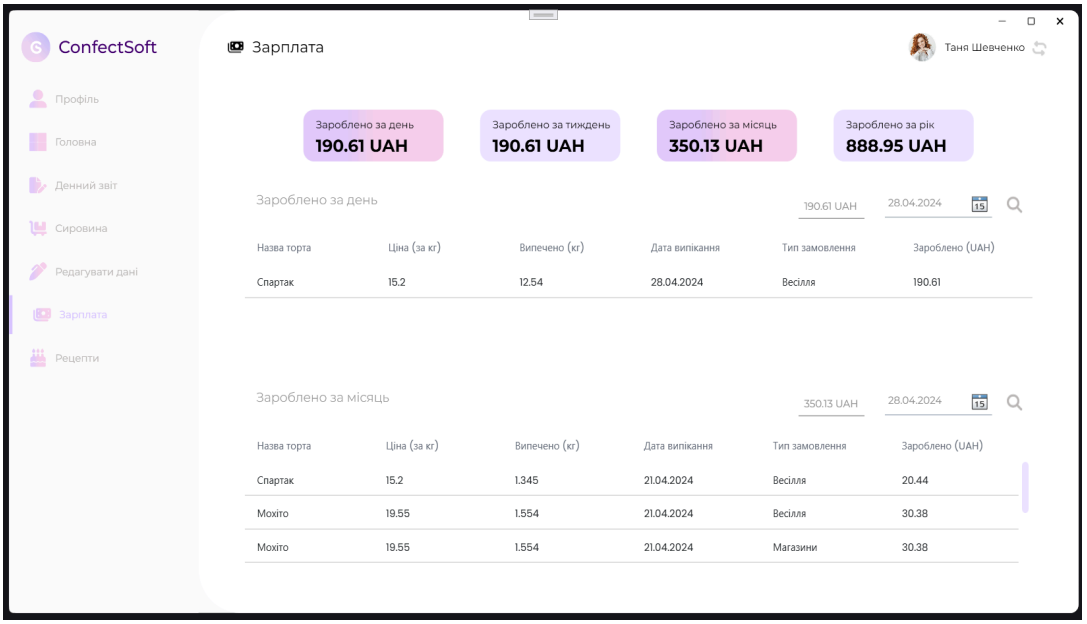


Рисунок 3.13 – Сторінка “Зарплата”

Сторінка “Зарплата” розділена на три секції:

– поточні дані за день, тиждень, місяць та рік – дані оновлюються одразу після додавання нових даних користувачем до системи;

– дані за поточний день – дана секція містить більш детальну інформацію про зарплату. Працівник має змогу переглянути всі дані які він додавав протягом дня, та скільки він заробив;

– пошук зарплати за заданим днем, місяцем, тижнем чи місяцем – функція допомагає користувачу здійснити пошук необхідних йому даних. Секція розділена на меню з пошуком та результатом даних, а також таблицю де користувач має змогу переглянути всю зароблену зарплату за заданий період.

Раніше, у пункті 3.2.4, було детально описано як відбувається автоматичний розрахунок заробітної плати працівника. Кожен з попередніх пунктів, які було написано вище має ту саму логіку, однак, кожен з них різні типи даних, про які також було згадано у підрозділі 3.1 та пункті 3.2.4.

3.4.6 Сторінка «Авторизація»

Сторінка «Авторизація» – це перший екран, з яким стикаються користувачі під час доступу до програми. Вона розроблена так, щоб вона була зручною та легкою для навігації, із чітким введенням полів імені користувача та пароля, а також кнопкою авторизації (рис. 3.14).

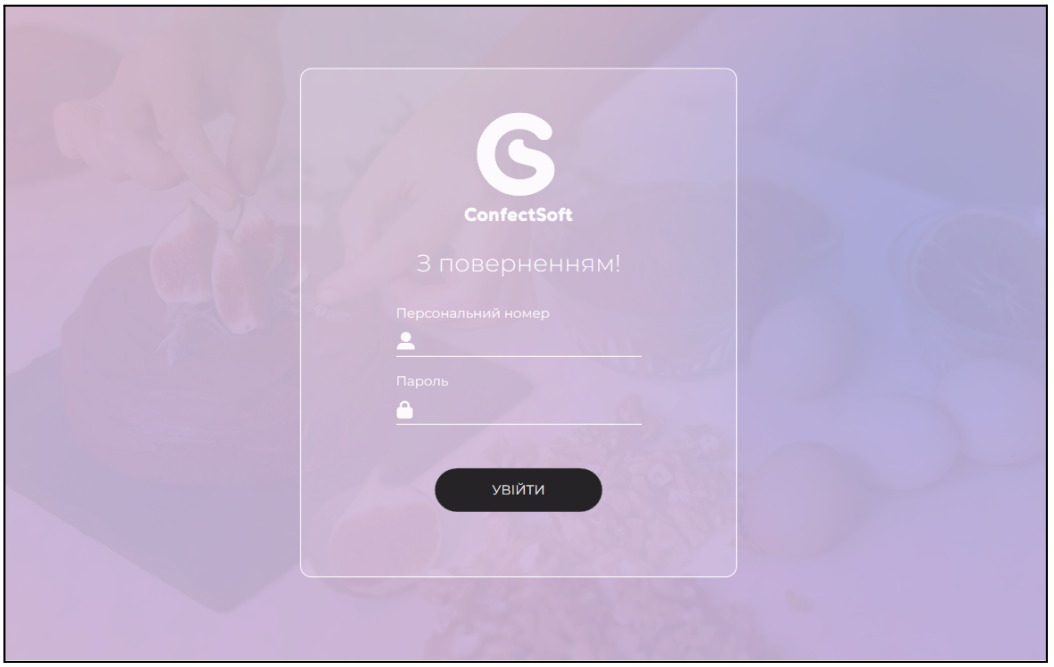


Рисунок 3.14 – Сторінка "Авторизація"

Кнопка є основною метою сторінки, оскільки вона дозволяє користувача входити в застосунок. Окрім полів введення та кнопки авторизації, сторінка містить логотип застосунку, який знаходиться зверху полів для введення та привітання користувача з поверненням.

Також авторизація включає перевірку правильно введених даних у поля та даними з бази даних. Додатково включені обмеження на кількість символів у полях та типи символів які можна вводити в полях, про які детальніше буде описано у підрозділі 3.5, а саме валідація застосунку.

3.5 Валідація застосунку

Валідація або перевірка є важливим аспектом в розробці застосунку, оскільки вона допомагає переконатися, що введені користувачем дані є точними та відповідають вимогам програми. Без перевірки користувачі можуть ввести неправильні дані або недійсну інформацію, що може призвести до неочікуваних помилок або навіть фатальних збоїв застосунку. Валідація особливо важлива в програмах, які обробляють конфіденційні дані або вимагають точного введення для правильної роботи. Саме тому в застосунку було включено наступні перевірки:

1. Перевірка формату – під час додавання чи редагування даних є певні поля, в які потрібно вводити інформацію певного типу. Наприклад, найменування випічки повинно містити тільки літери українського алфавіту, кількість випеченої продукції має десяткові числа, а код сировини має включати тільки цілі числа. Нижче представлений код, який відповідає за перевірку формату:

– метод numberValidationTextBox – перевіряє, чи введене значення є числом від 0 до 9. Якщо введене значення не є числом, то воно не буде оброблено і в полі значення не буде відображено:

```
Regex regex = new Regex("[^0-9]+");
e.Handled =regex.IsMatch(e.Text);}
```

– метод lettterValidationTextBox – перевіряє, чи введене значення є українською літерою. Якщо значення, яке було введене не є українською літерою, то воно не буде оброблено:

```
private void letterValidationTextBox(object sender,
TextCompositionEventArgs e){
Regex regex = new Regex("[^a-щьюяїієґA-ЩЬЮЯЇІЄҐ ]+");
e.Handled = regex.IsMatch(e.Text);}
```

– метод numberFValidationTextBox – перевіряє, чи введене значення є десятковим числом (наприклад 2.212). Якщо введене значення не є десятковим числом, то воно не буде оброблено. Однак, за винятком випадку, коли введено «.», і воно ще не присутнє в текстовому полі:

```
private void numberFValidationTextBox(object sender,
TextCompositionEventArgs e){
TextBox textBox = (TextBox)sender;
if (e.Text == "." && !textBox.Text.Contains(".")){
e.Handled = false;}
else { Regex regex = new Regex("[^0-9]+");
e.Handled = regex.IsMatch(e.Text);}}
```

Саме для того, щоб перевірити введення символів користувачем, використовується регулярний вираз Regex. Саме він формує шаблон пошуку, і це використовується для знаходження певних рядків або груп рядків у тексті.

2. Перевірка на діапазон – під час створення бази даних, кожне поле отримало обмеження на кількість символів. Саме тому, в застосунку необхідно передбачити перевірку на кількість введених символів в поля. Наприклад, код випічки повинен містити тільки 4 цифри, або ім’я користувача при авторизації повинно становити тільки 6 цифр.

Тому, щоб уникнути введення зайвих символів в полях використовується властивість в файлі XAML – Max.Length, що відповідає за довжину у полях введення.

3. Перевірка на безпеку – одна з важливих перевірок, яка включає в себе перевірку на наявність шкідливих або небажаних введень, такі як SQL–ін’єкції. Тому, для того щоб запобігти SQL–ін’єкціям, до кожного SQL–запиту було додано параметри, про які вже було згадано у підрозділі 3.2. Під час використання параметрів, дані, які були введені користувачем ніколи не вставляються безпосередньо в запит, що робить неможливим виконання шкідливого коду. Окрім безпеки, параметризовані запити можуть покращити продуктивність застосунку, а також використання параметрів може зробити SQL код більш читабельним і легких для розуміння. Також, серед заходів безпеки було розглянуто «моніторинг і журналювання». Всі дії які доступні в застосунку, наприклад, додавання чи просто перегляд сторінки, фіксуються в базі даних. Це допоможе перевірити наявність підозрілих дій серед користувачів. Перегляд дій користувачів доступні на сторінці адміністратора, про яку було згадано в підрозділі 3.4.

4. Перевірка на наявність – в застосунку, кожне поле для введення даних є обов’язковим. В нижче представленому коді, валідація полягає саме у перевірці на наявність підказок для користувача. Якщо користувач не ввів значення в поле «код» («newId») чи назву («newTitle»), підказка залишається незмінною. В цьому випадку, користувач отримує повідомлення про відсутність даних у полях:

```
if (newId == "0000" || newTitle == "Назва..."){
MessageBox.Show("Поля порожні, будь ласка введіть дані");return;}
```

5. Перевірка унікальності – в базі даних, є певні поля які вимагають перевірки на унікальність даних, наприклад персональний номер користувача, чи код клієнта. Саме тому, перед додаванням даних включено перевірку, яка буде порівнювати з даними які вже є в системі, і даними, які користувач хоче додати. Нижче представлений код, який виконує перевірку на унікальність даних, на базі таблиці магазини («shops»):

```
string checkQuery = "INSERT INTO `shops` (`shop_id`, `shop_title`)
VALUES (@newId, @newTitle);"; MySqlCommand checkCommand = new
MySqlCommand(checkQuery,db.getConnection());
checkCommand.Parameters.AddWithValue("@newId", newId);+
checkCommand.Parameters.AddWithValue("@newTitle", newTitle);
db.openConnection();
```

Спершу створюється параметризований SQL–запит, який містить команду INSERT для додавання даних. Після відкривається з'єднання з базою даних та виконується SQL–запит, виконується перевірка результату:

```
int count = Convert.ToInt32(checkCommand.ExecuteScalar());
if (count > 0){MessageBox.Show("Такий код вже існує, створіть
інший"); db.closeConnection();return;}
```

Якщо count більше 0, то це означає що запис з таким кодом (shop_id) вже існує в базі даних. Користувач отримує повідомлення про помилку, з'єднання з базою даних закривається.

6. Підказки для користувачів є досить важливими у користуванні, оскільки це дозволить побачити приклад того, які дані необхідно ввести в поле. У наступному блоці коду використовуються події для взаємодії з текстовими полями. Подія this.PreviewMouseDown += IsFieldActive_PreviewMouseDown викликається, коли користувач натискає мишу.

Функція IsFieldActive_PreviewMouseDown буде викликана при події. Після встановлюються початкові значення текстових полів назви магазину («txtShopTitle») та коду («txtCode»). Наступна подія .GotFocus викликається, коли текстове поле отримує фокус, тобто коли користувач натискає на нього. При цій події буде викликана функція RemoveText(…).

В свою чергу, подія .LostFocus викликається тоді, коли текстове поле втрачає фокус, а саме коли користувач натискає за межами поля.

Як було вище згадано, функція RemoveText(…) викликається при події .GotFocus. Функція перевіряє чи поле містить підказку, в даному випадку

«Назва…» для поля назви магазину. Якщо це так, він змінює текст на пустий рядок і повертає фокус на це поле:

```
public void RemoveTextShopTitle(object sender, EventArgs e){
if (txtShopTitle.Text == "Назва..."){
txtShopTitle.Text = "";
txtShopTitle.Focus();}}
```

Функція AddText(…) викликається коли текстове поле втрачає фокус. Якщо текст у полі пустий або містить лише пробіли, він змінює текст на підказку «Назва…»:

```
public void AddTextShopTitle(object sender, EventArgs e){
if (string.IsNullOrWhiteSpace(txtShopTitle.Text)){
txtShopTitle.Text = "Назва...";}}
```

Код, що подано нижче використовує подію для перевірки, чи є активним текстові поля, і якщо вони активні та пусті, встановлює їх значення на значення за замовчуванням:

```
private void IsFieldActive_PreviewMouseDown(object sender,
MouseButtonEventArgs e){if (txtShopTitle != null &&
txtShopTitle.IsFocused && string.IsNullOrWhiteSpace
txtShopTitle.Text)){txtShopTitle.Text = "Назва...";}
if (txtCode != null && txtCode.IsFocused &&
string.IsNullOrWhiteSpace(txtCode.Text)){
txtCode.Text = "0000";}}
```

3.6 Тестування застосунку

Тестування програми – це процес перевірки її відповідності вимогам і очікуваній поведінці. Тестування застосунку має вирішальне значення, оскільки воно допомагає переконатися, що програмне забезпечення вільне від помилок, без помилок і відповідає всім поставленим вимогам. Без тестування майже неможливо гарантувати, що програмне забезпечення працюватиме так, як

очікувалося чи призначено, що може призвести до розчарування та зниження довіри користувачів.

У контексті розробки програмного забезпечення, обрано ручне тестування (див. табл. Ж.1). Це рішення було прийнято з урахуванням того, що розробляється невеликий проєкт, і цікавить глибока, і детальна перевірка. Ручне тестування допомагає виявити помилки, які автоматизоване тестування може не виявити, особливо ті, що стосуються взаємодії з користувачем і функціональності. Воно допомагає виявити будь–які проблеми на ранніх стадіях процесу розробки, що може запобігти їх переростанню в більші проблеми пізніше.

Ручне тестування також дозволяє краще зрозуміти, як користувачі повинні взаємодіяти з продуктом, що, в свою чергу допомагає покращити дизайн застосунку і зробити його більш привабливим і зручним для користувачів.

Висновки до розділу 3

У третьому розділі ретельно досліджено розробку інформаційної структури даних. Створення таблиць баз даних виконано з урахуванням всіх необхідних вимог до структуризації даних. Необхідні SQL–запити до бази даних розроблені для забезпечення швидкого та ефективного доступу до інформації. Також розглянуто детальну розробку користувацьких звітів. Візуальна частина застосунку розроблена з урахуванням принципів зручності користування. У свою чергу, валідація та тестування застосунку були проведені для забезпечення його надійності та стабільності роботи.

ВИСНОВКИ

В процесі розробки кваліфікаційної роботи було створено програмне забезпечення «Розробка програмного забезпечення для цифрової підтримки діяльності кондитерського підприємства».

В результаті виконаної роботи, було розроблено опис предметної області, методи та засоби спостереження та збирання інформації, що допомогли виявити проблемні точки працівників у роботі на підприємстві. Проаналізовано ринок на наявність аналогічних застосунків, в результаті чого не знайдено жодного програмного забезпечення з такою самою специфікою, як розроблюваний застосунок. Визначено функціональні вимоги, вимоги до дизайну системи, а також вимоги до безпеки та надійності. Спроєктовано діаграму прецедентів, що допомогла зрозуміти як користувач повинен взаємодіяти з системою. В свою чергу, ER–діаграми визначили структуру та організацію даних, а також зв’язки між ними, Діаграма класів дозволила чітко відобразити зв’язки між різними класами та їхніми взаємодіями. Окрім діаграми, розроблено архітектуру та UI–дизайн застосунку в Figma, вибрано технології для розробки. Проєкт також включає детальний опис розробки кожної сторінки застосунку, який був розроблений в Microsoft Visual Studio 2019, автоматичного обчислення зарплати працівника та залишку сировини за місяць. Водночас, розробка застосунку включає валідацію та проведено ручне тестування.

Даний застосунок є цінним для потенційних користувачів, оскільки він вирішує конкретні проблеми, з якими вони стикаються на роботі щодня. Він автоматизує обчислення зарплати, залишку сировини та допомагає підвищити ефективність роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. UML Use Case Diagram Tutorial. *YouTube*: веб–сайт. URL: https://youtu.be/zid–M (дата звернення: 20.03.2024).

2. CRM система для товарного бізнесу, простий облік фінансів. *Huge Profit:* веб–сайт. URL: https://h–profit.com/ua/ (дата звернення: 22.03.2024).

3. Conceptual, Logical & Physical Data Models. *YouTube*: веб–сайт. URL: https://youtu.be/RJ9 (дата звернення 28.03.2024).

4. Лавріщева К.М. ПРОГРАМНА ІНЖЕНЕРІЯ: підручник. Київ: Академічна книга, 2008. 319 с. URL: https://csc.knu. (дата звернення 29.03.2024).

5. UML Class Diagrams. *YouTube:* веб–сайт. URL: https://youtu.be/6XrL5jXmTwM?si=Z (дата звернення 01.04.2024).

6. Introducing Figma: A Beginners Tutorial (2023 UI UX Design). *YouTube*: веб–сайт. URL: https://youtu.be/JGL (дата звернення: 08.04.2024).

7. Figma UX tutorial for beginners – Wireframe. *YouTube*: веб–сайт. URL: https://youtu.be/D4NyQ5iOMF0?si=t (дата звернення: 08.04.2024)

8. Figma Learn. *Figma Learn:* веб–сайт. URL: https://help.figma.com/hc/en–us (дата звернення: 08.04.2024).

9. The Greatest Design System UI Kits for Figma! Full tutorial. *YouTube*: веб–сайт. URL: https://youtu.be/gnRxVg (дата звернення 09.04.2024).

10. MySQL Full Course for free (2023). *YouTube*: веб–сайт. URL: https://youtu.be/5OdVJbN (дата звернення 10.04.2024).

11. MySQL Documentation. *Dev MySQL*: веб–сайт. URL: https://dev.mysql.com/doc/ (дата звернення 11.04.2024).

12. Мулеба О.Ю. Інформаційні системи та реляційні бази даних: навчальний посібник. Електронне видання, 2018. 118 с. URL: https://dspace.uzhnu.edu.ua/ (дата звернення 13.04.2024).

13. C# WPF Tutorial. *YouTube*: веб–сайт. URL: https://youtube.com/playlist?list=PLih2K (дата звернення: 20.04.2024).

14. WPF in Visual Studio 2019 | Getting Started. *YouTube:* веб–сайт. URL: https://youtu.be/73PsdMMINRk?si= (дата звернення: 20.04.2024).

15. C# GUI Tutorial using WPF | XAML | – Windows Presentation Foundation. *YouTube*: веб–сайт. URL: https://youtu.be/oSeYvMEH7jc? (дата звернення 21.04.2024).

16. Connection to a Database C# – WPF and MySQL. *YouTube:* веб–сайт. URL: https://youtu.be/OPDPI5exPp8?si=x (дата звернення 02.05.2024).

17. LiveCharts2 Docs. *LiveCharts 2*: веб–сайт. URL: https://livecharts.dev/docs/wpf/2.0.0–rc2/gallery (дата звернення 05.05.2024).

18. C# Language Documentation. *Microsoft Learn*: веб–сайт. URL: https://learn.microsoft.com/en–us/dotnet/csharp/ (дата звернення 08.05.2024).

19. Система Дистанційної Освіти УКД. *СДО УКД:* веб–сайт. URL: https://online.ukd.edu.ua/. (дата звернення 13.05.2024).

20. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ щодо виконання кваліфікаційних робіт для здобувачів освітнього рівня бакалавр за освітньою програмою «Розробка та тестування програмного забезпечення спеціальності 121 «Інженерія програмного забезпечення» У ЗВО «Університет Короля Данила». Івано–Франківськ: ЗВО «Університет Короля Данила», 2021. 56 с. (дата звернення 14.05.2024).

21. Windows Presentation Foundation documentation. *Microsoft Learn:* веб–сайт. URL: https://learn.microsoft.com/ (дата звернення 15.05.2024).

22. XAML Syntax in Detail. *Microsoft Learn:* веб–сайт. URL: https://learn.microsoft.com/en–us/dotnet/desktop/wpf/advanced/xaml–syntax–in–detail? (дата звернення: 16.05.2024).

ДОДАТКИ

Додаток А

Розробка діаграми прецедентів (Use Case diagram) для підрозділу 2.2.

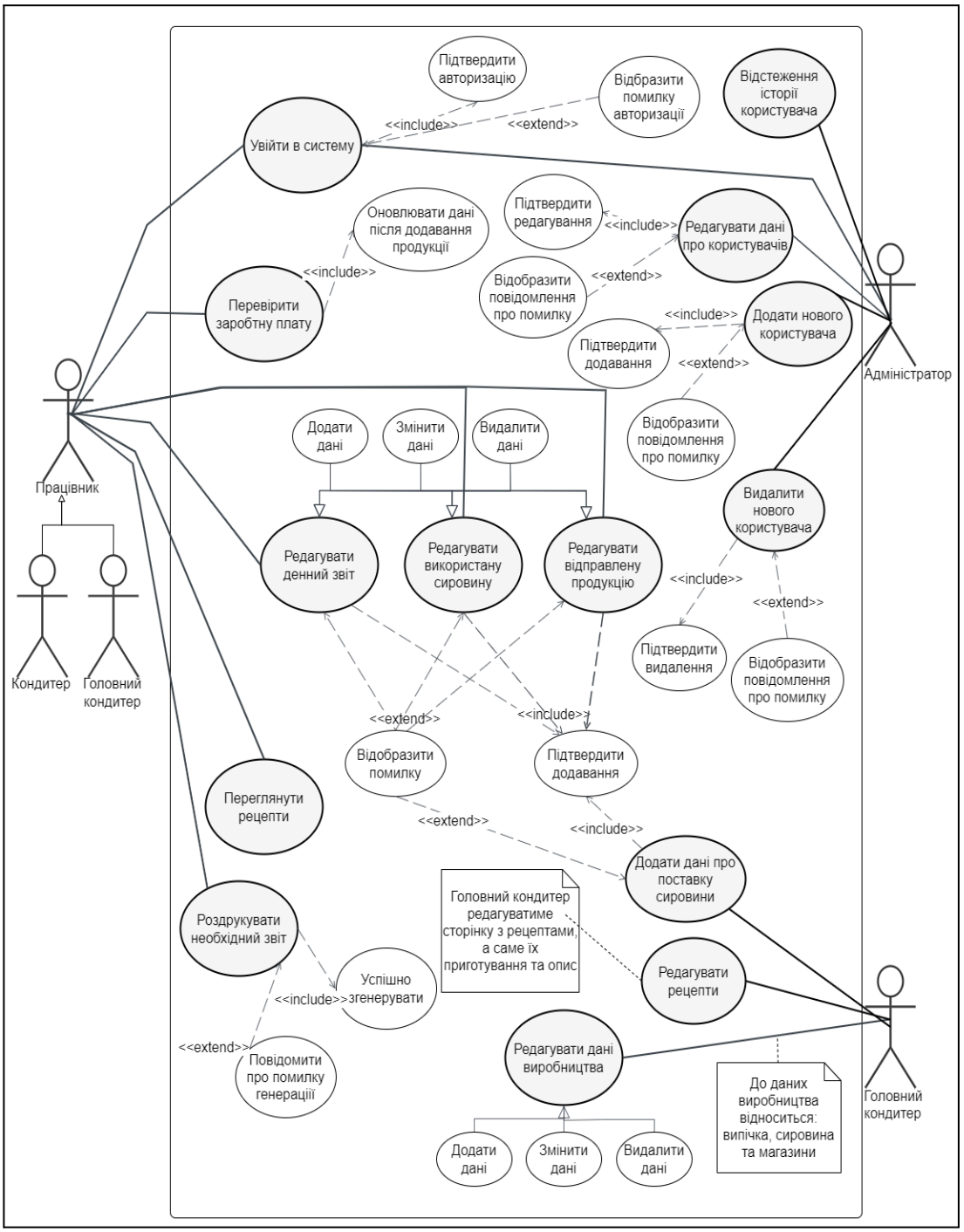


Рисунок А.1 – Діаграма прецедентів системи ведення звітів кондитерського підприємства

Додаток Б

Моделювання концептуальної моделі для пункту 2.3.1 «Концептуальна модель даних».

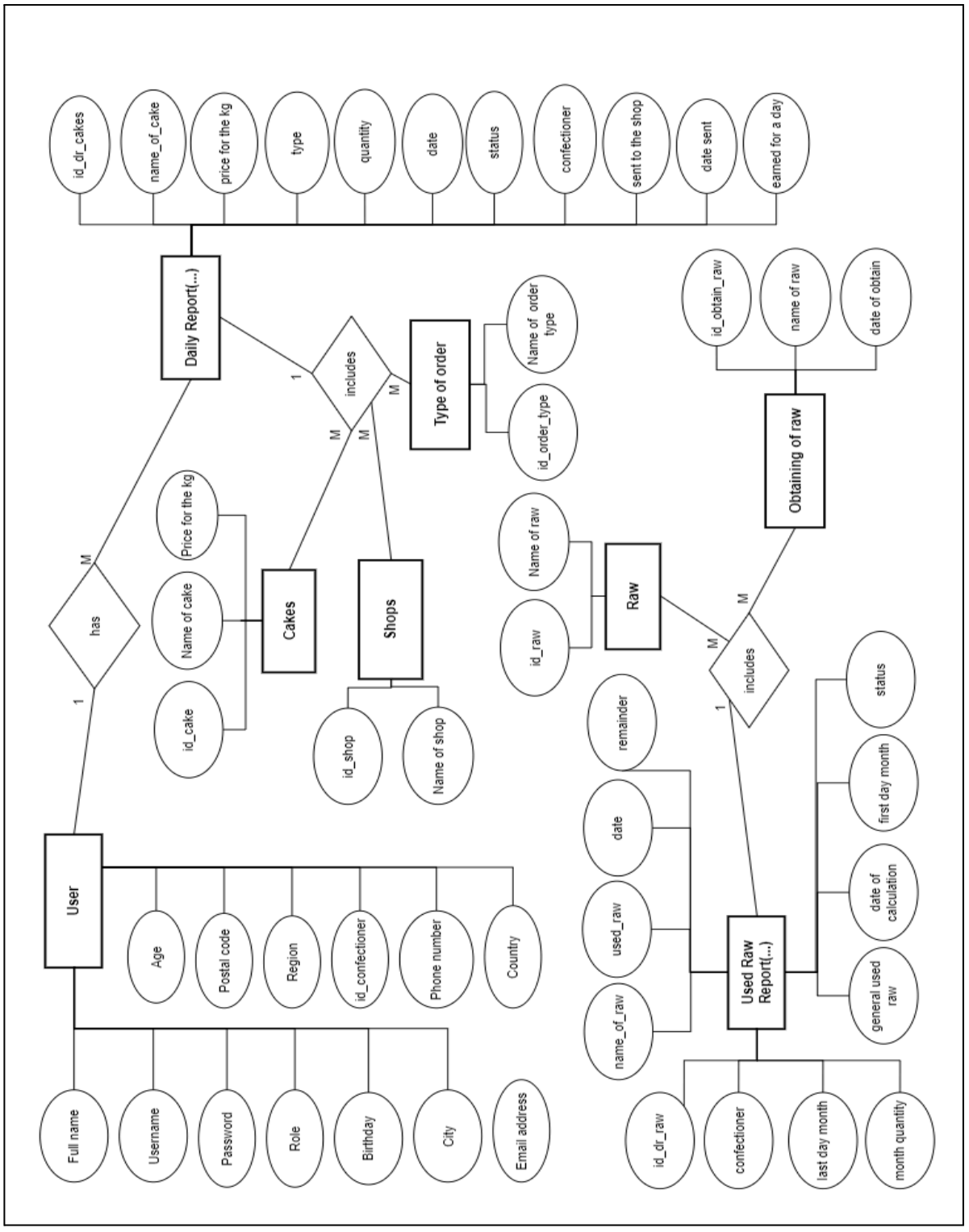


Рисунок Б.1 – Концептуальна модель даних

Додаток В

Розробка логічної моделі даних для пункту 2.3.2 «Логічна модель даних».

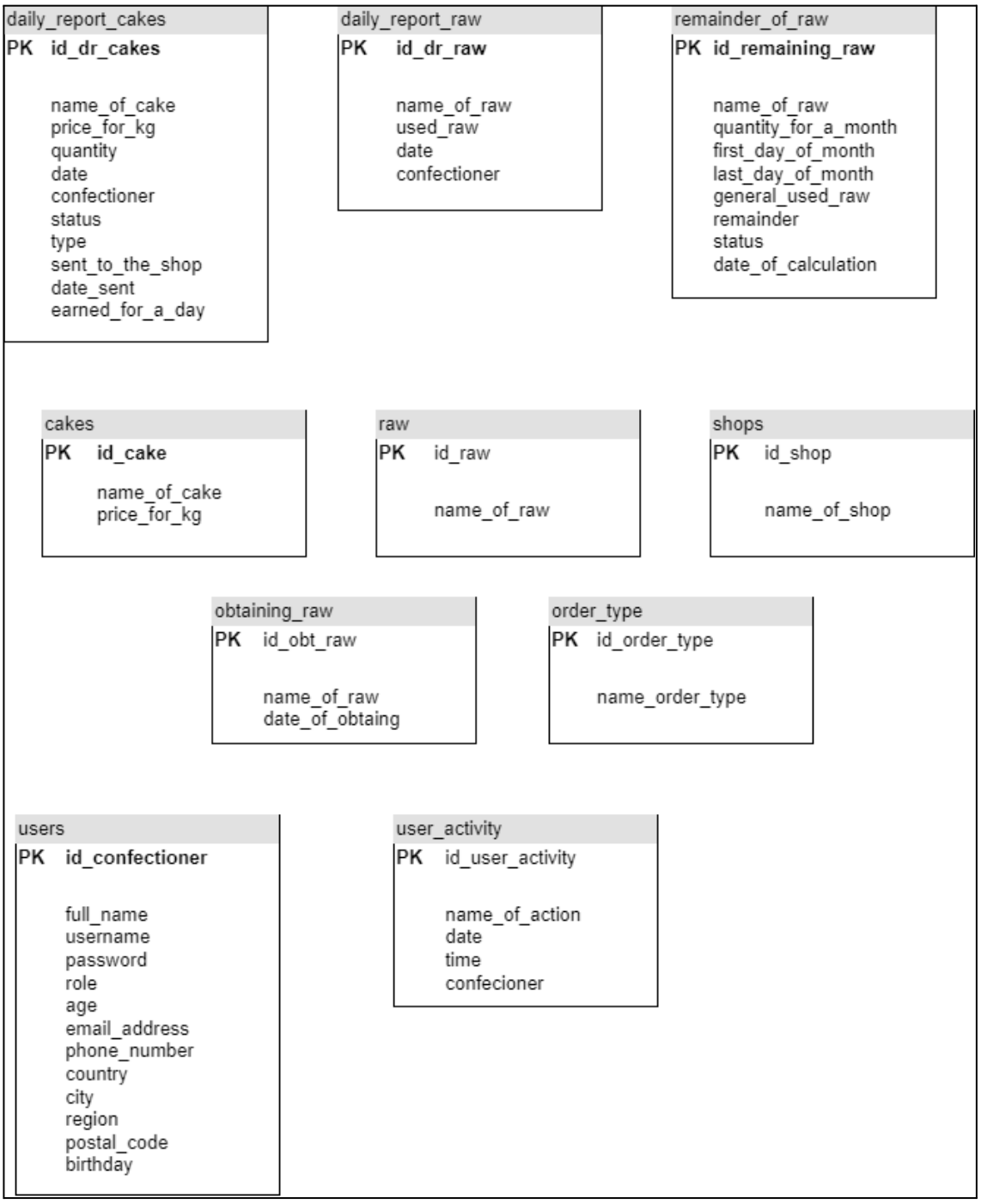


Рисунок В.1 – Логічна модель даних

Додаток Г.1

Генерація фізичної моделі даних бази даних в середовищі MySQL Workbench для пункту 2.3.3.

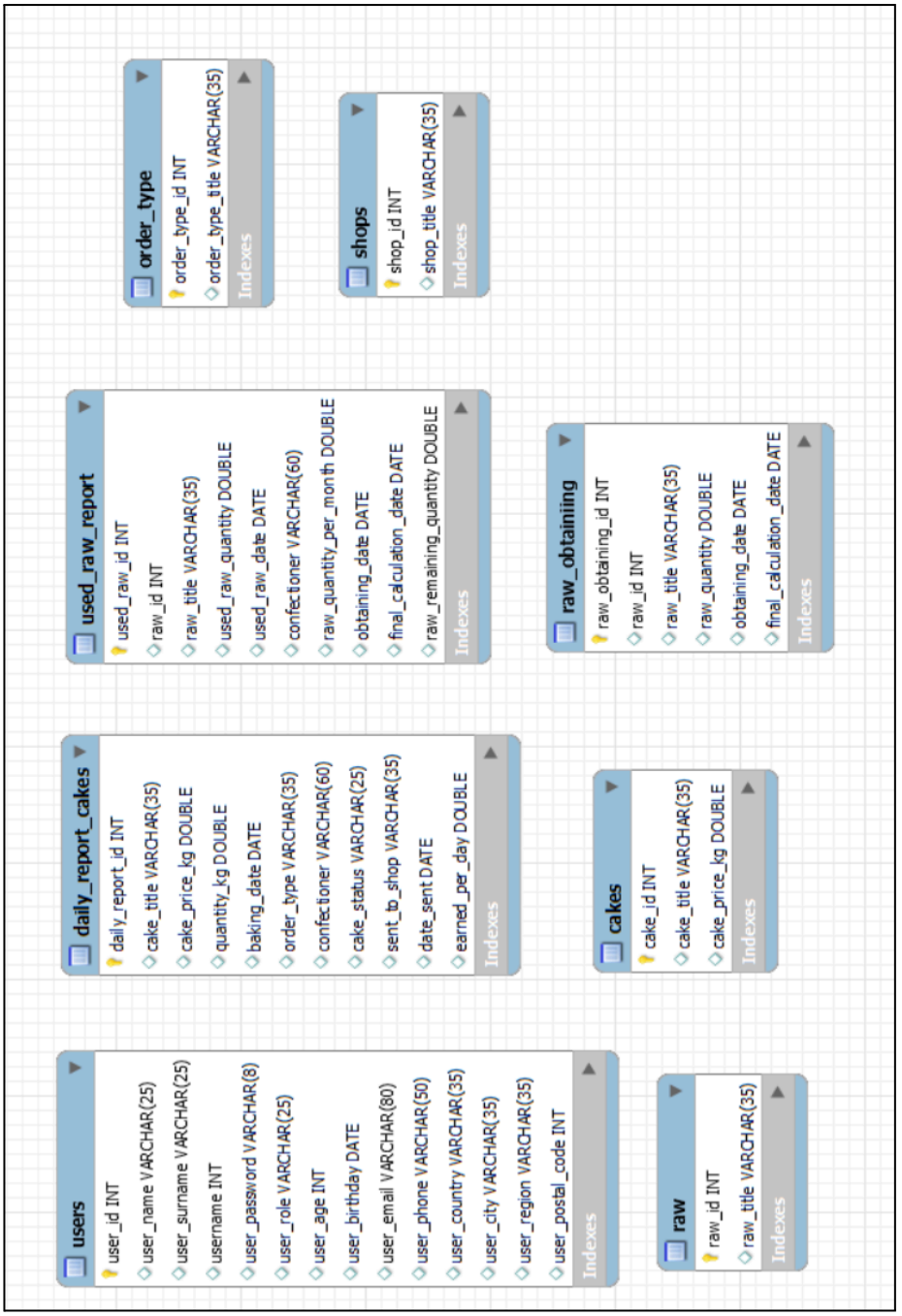


Рисунок Г.1 – Фізична модель даних

Додаток Г.2

Таблиця Г.2.1

«Атрибути таблиць бази даних застосунку»

Таблиця	Атрибут	Опис атрибуту
users	user_id	ID користувача
	user_name	Ім’я
	user_surname	Прізвище
	username	Ім’я користувача в системі, його персональний номер.
	user_password	Пароль
	user_role	Роль (адміністратор, головний кондитер, кондитер)
	user_age	Вік
	user_birthday	День народження
	user_email	Електронна пошта
	user_phone	Номер телефону
	user_country	Країна
	user_city	Місто
	user_region	Область
	user_postal_code	Поштовий код
cakes	cake_id	ID торта
	cake_title	Найменування торта
	cake_price_kg	Ціна за кілограм
raw	raw_id	ID сировини
	raw_title	Найменування сировини
shop	shop_id	ID магазину
	shop_title	Найменування магазину
order_type	order_type_id	ID типу замовлення
	order_type_title	Найменування типу замовлення

daily report cakes	daily_report_id	ID
	cake_title	Найменування торта
	cake_price_kg	Ціна за кілограм
	quantity_kg	Вага випеченого торта
	baking_date	День виготовлення
	order_type	Тип замовлення
	confectioner	Кондитер
	cake_status	Статус продукції
	sent_to_shop	Магазин, куди відправлено продукцію
	date_sent	Дата відправки
	earned_per_day	Зароблено за день
raw obtaining	raw_obtaining_id	ID дати поставки сировини
	raw_id	ID сировини
	raw_title	Найменування сировини
	raw_quantity	Кількість використаної сировини
	obtaining_date	Дата поставки
	final_calculation_date	Дата до якої необхідно рахувати залишок сировини
used raw report	used_raw_id	ID використаної сировини
	raw_id	ID сировини
	raw_title	Найменування сировини
	used_raw_quantity	Кількість використаної сировини
	used_raw_date	Дата використаної сировини
	confectioner	Кондитер
	raw_quantity_per_month	Кількість сировини дана на місяць
	obtaining_date	Дата поставки сировини
	final_calculation_date	Дата до якої необхідно рахувати залишок сировини
	raw_remaining_quantity	Кількість залишку сировини

Додаток Д

Розробка діаграми класів застосунку для підрозділу 2.4 «Проєктування діаграми класів».

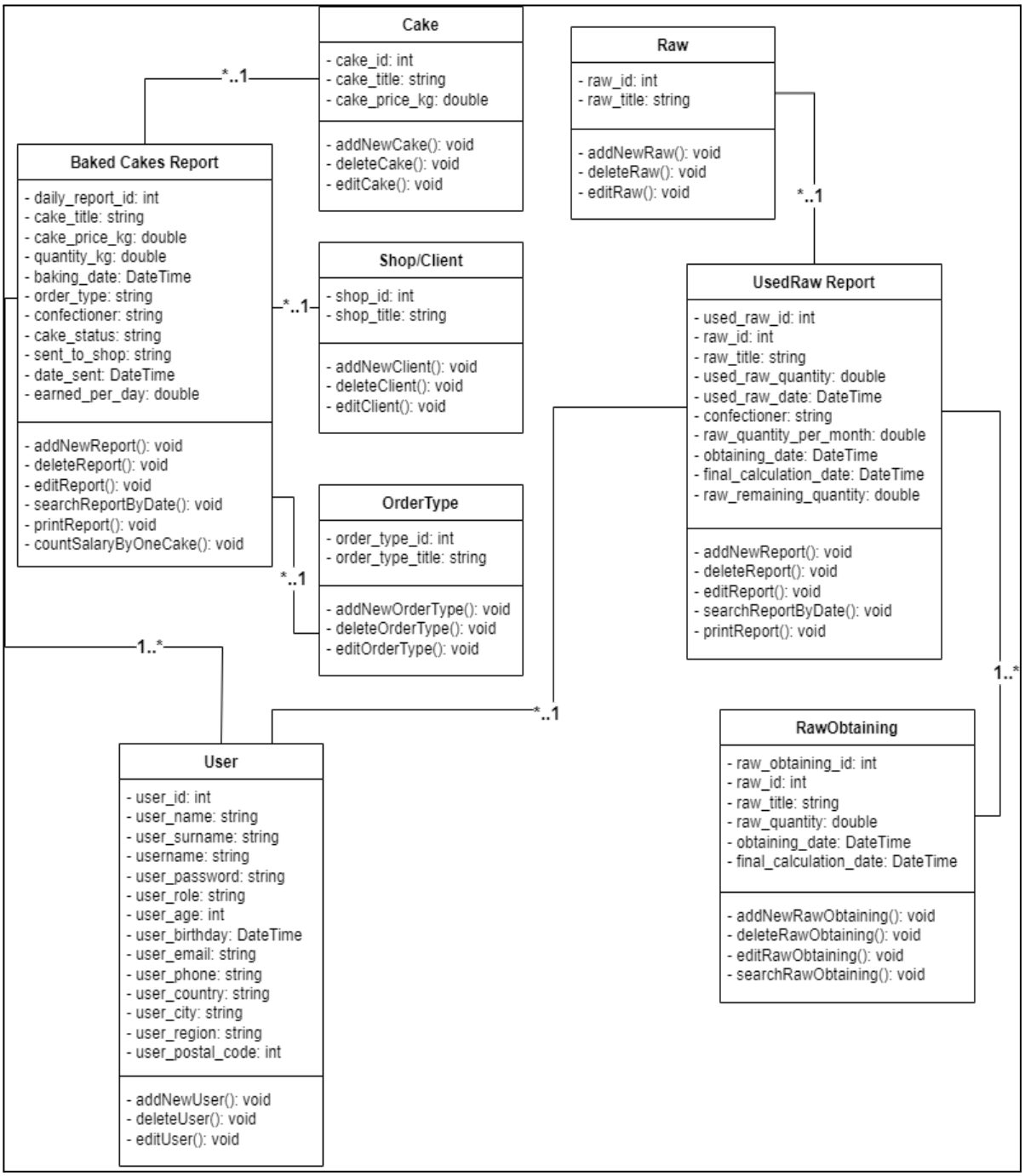


Рисунок Д.1 – Діаграма класів

Додаток Е

Розробка каркасів для розроблюваного застосунку для підрозділу 2.6 «Розробка UI–дизайну в Figma».

Рисунок Е.1 – Wireframes (каркаси) застосунку

Додаток Є

Метод для генерації звіту випеченої продукції для підрозділу 3.3:

```
        private void btnPrintData_Click(object sender, RoutedEventArgs e){
        bindDailyReportTableForPrinting();
        if (bakedCakesReport_DailyReportForm.Items.Count == 0){
        MessageBox.Show("Немає даних для друку"); return;}
        string confectioner = LoginViewModel.UserInfo;
        string _userRole = LoginViewModel.UserPosition;
        //Creating of new FlowDocument
        FlowDocument doc = new FlowDocument();
        doc.PageWidth = 842; doc.PageHeight = 595;
        doc.PagePadding = new Thickness(30);
        doc.ColumnWidth = Double.PositiveInfinity;
        doc.FontFamily = new System.Windows.Media.FontFamily("Arial");
        // Adding of current date and time
        Paragraph currentTime = new Paragraph(new Run("Дата та час
генерації звіту: " + DateTime.Now.ToString()));
        currentTime.FontSize = 9;
        currentTime.TextAlignment = TextAlignment.Right;
        doc.Blocks.Add(currentTime);
        //Adding header
        Paragraph title = new Paragraph(new Run("Звіт випеченої продукції
 за " + dateOfBaking.SelectedDate.Value.ToString("dd MMMM yyyy";
        title.FontSize = 14;
        title.TextAlignment = TextAlignment.Left;
        doc.Blocks.Add(title);
        //Add the confectioner's name
        Paragraph confectionerName = new Paragraph(new Run("Кондитер: " +
 confectioner));
        confectionerName.FontSize = 12;
        confectionerName.TextAlignment = TextAlignment.Right;
        doc.Blocks.Add(confectionerName);
        //Add the confectioner's role
        Paragraph confectionerRole = new Paragraph(new Run("Посада: " +
 _userRole));
        confectionerRole.FontSize = 12;
        confectionerRole.TextAlignment = TextAlignment.Right;
```

```
doc.Blocks.Add(confectionerRole);
//Creating of new table
Table table = new Table();
table.CellSpacing = 0; table.FontSize = 12;
table.BorderBrush = System.Windows.Media.Brushes.Black;
table.BorderThickness = new Thickness(0.5);
// Set columns' width
int numOfColumns = 8;
for (int i = 0; i < numOfColumns; i++)
{ TableColumn column = new TableColumn();
column.Width = new GridLength(1, GridUnitType.Auto);
table.Columns.Add(column);}
// Creating and adding header of table
TableRowGroup headerGroup = new TableRowGroup();
TableRow headerRow = new TableRow();
headerRow.Cells.Add(CreateCell("Код"));
headerRow.Cells.Add(CreateCell("Випічка"));
headerRow.Cells.Add(CreateCell("Всього випечено"));
headerRow.Cells.Add(CreateCell("Ціна за кг"));
headerRow.Cells.Add(CreateCell("Дата випікання"));
headerGroup.Rows.Add(headerRow);
table.RowGroups.Add(headerGroup);
// Add data from Datagrid to the table
TableRowGroup itemGroup = new TableRowGroup();
foreach (DailyReport item in
bakedCakesReport_DailyReportForm.Items){
TableRow row = new TableRow();
row.Cells.Add(CreateCell(item.DailyReportId.ToString()));
row.Cells.Add(CreateCell(item.CakeTitle.ToString()));
row.Cells.Add(CreateCell(item.QuantityKg.ToString()));
row.Cells.Add(CreateCell(item.CakePriceKg.ToString()));
row.Cells.Add(CreateCell(item.BakingDate.ToString("dd.MM.yyyy")));
itemGroup.Rows.Add(row);}
table.RowGroups.Add(itemGroup);
//Add table to FlowDocument
doc.Blocks.Add(table);
// Creating of new PrintDialog
PrintDialog printDialog = new PrintDialog();
```

Додаток Ж

Таблиця Ж.1

«Сценарій ручного тестування застосунку»

№		
1	**Заголовок**	**Авторизація користувача**
	Крок	**Очікуваний результат**
	Ввід значення в поле «Персональний номер»	Введене значення відобразиться в полі «Персональний номер».
	Ввід значення в поле «Пароль»	Введене значення відобразиться в полі «Пароль», приховуючи пароль користувача під символами.
	Натиснення кнопки «Увійти», коли дані введено неправильно	На екрані користувача з’являється повідомлення про те що дані введено неправильно.
	Натиснення кнопки «Увійти», коли дані введено правильно	Відкриття головної сторінки працівника.
2	**Заголовок**	**«Редагувати дані – Додавання нової випічки»**
	Крок	**Очікуваний результат**
	Введення в поле «Код»	Введене значення відобразиться в полі «Код»
	Введення в поле «Назва випічки»	Введене значення відобразиться в полі «Назва випічки»
	Введення в поле «Ціна за кг»	Введене значення відобразиться в полі «Ціна за кг»
	Натиснення кнопки «Додати запис», якщо всі поля заповнені.	На екрані користувача з’являється повідомлення про те що нову випічку успішно додано. Нова випічка добавляється до таблиці.
	Натиснення кнопки «Додати запис», якщо поля (чи поле) не заповнено.	На екрані користувача з’являється повідомлення про те що виникла помилка.
3	**Заголовок**	**Редагувати дані – Видалення випічки**
	Крок	**Очікуваний результат**
	Натиснення кнопки «Видалити запис», коли отримані дані отримано коректно.	На екрані користувача з’являється про успішне видалення. Успішне видалення даних з таблиці, бази даних.
	Натиснення кнопки «Видалити запис», коли з рядка таблиці неможливо отримати дані.	На екрані користувача з’являється повідомлення про помилку.
4	**Заголовок**	**Редагувати дані – Змінення найменування випічки**
	Крок	**Очікуваний результат**
	Натиснення кнопки «Змінити запис», коли дані з рядка успішно отримано.	На екрані користувача з’являється нове вікно для редагування даних.
	Натиснення кнопки «Змінити запис», коли дані з рядка не отримано.	Вікно редагування даних не відкриється, отримання повідомлення про помилку
5	**Заголовок**	**Редагувати дані – Додавання нової сировини**

	Крок	**Очікуваний результат**
	Введення в поле «Код»	Введене значення відобразиться в полі «Код»
	Введення в поле «Найменування сировини»	Введене значення відобразиться в полі «Найменування сировини»
	Натиснення кнопки «Додати запис», якщо всі поля заповнені.	На екрані користувача з’являється повідомлення про те що нову сировину успішно додано. Нова сировина добавляється до таблиці.
	Натиснення кнопки «Додати запис», якщо поля (чи поле) не заповнено.	На екрані користувача з’являється повідомлення про те, що виникла помилка.
6	**Заголовок**	**Редагувати дані – Видалення сировини**
	Крок	**Очікуваний результат**
	Натиснення кнопки «Видалити запис», коли дані з рядка отримано успішно.	На екрані користувача з’являється про успішне видалення. Успішне видалення даних з таблиці, бази даних.
	Натиснення кнопки «Видалити запис», дані з рядка таблиці не отримано.	На екрані користувача з’являється повідомлення про помилку.
7	**Заголовок**	**Редагувати дані – Змінення найменування сировини**
	Крок	**Очікуваний результат**
	Натиснення кнопки «Змінити запис», коли дані з рядка не отримано.	Вікно для редагування відкрито не буде.
	Натиснення кнопки «Змінити запис», коли дані з рядка отримано	На екрані користувача з’являється нове вікно для редагування даних.
8	**Заголовок**	**Редагувати дані – Додавання нового магазину**
	Крок	**Очікуваний результат**
	Введення в поле «Код»	Введене значення відобразиться в полі «Код»
	Введення в поле «Найменування магазину»	Введене значення відобразиться в полі «Найменування магазину»
	Натиснення кнопки «Додати запис», якщо всі поля заповнені.	На екрані користувача з’являється повідомлення про те що новий магазин успішно додано. Новий магазин додається до таблиці.
	Натиснення кнопки «Додати запис», якщо поля не заповнено.	На екрані користувача з’являється повідомлення про те що виникла помилка.
9	**Заголовок**	**Редагувати дані – Видалення магазину**
	Крок	**Очікуваний результат**
	Натиснення кнопки «Видалити запис», коли дані з рядка отримано.	На екрані користувача з’являється про успішне видалення. Успішне видалення даних з таблиці, бази даних.
	Натиснення кнопки «Видалити запис», коли дані з рядка не отримані.	На екрані користувача з’являється повідомлення про помилку.
10	**Заголовок**	**Редагувати дані – Змінення найменування магазину**
	Крок	**Очікуваний результат**
	Натиснення кнопки «Змінити запис», коли дані з рядка не отримано	Вікно для редагування не буде відкрито

	Натиснення кнопки «Змінити запис», коли дані з рядка отримано	На екрані користувача з’являється нове вікно для редагування даних.
11	**Заголовок**	**Додавання випеченої продукції**
	Крок	**Очікуваний результат**
	Вибрати необхідну випічку з ComboBox	Відображення вибраної продукції користувачем.
	Введення вагу продукції в поле «К–сть (кг)»	Введене значення відобразиться в полі «К–сть (кг)»
	Вибрати необхідну дату випікання.	Відображення вибраної дати користувачем.
	Вибрати тип замовлення з ComboBox	Відображення вибраного типу замовлення.
	Натиснення кнопки «Додати запис», коли всі поля заповнені.	На екрані користувача з’являється повідомлення. Дані додаються в таблицю і базу даних.
12	**Заголовок**	**Видалення випеченої продукції**
	Крок	**Очікуваний результат**
	Натиснення кнопки «Видалити запис», коли дані з рядка отримано	На екрані користувача з’являється про успішне видалення. Успішне видалення даних з таблиці, бази даних.
	Натиснення кнопки «Видалити запис», коли дані з рядка не отримано	На екрані користувача з’являється повідомлення про помилку.
13	**Заголовок**	**Додавання використаної сировини**
	Крок	**Очікуваний результат**
	Вибрати необхідну сировину з ComboBox	Відображення вибраної сировини користувачем.
	Введення вагу сировини в поле «К–сть (кг)»	Введене значення відобразиться в полі «К–сть (кг)»
	Вибрати необхідну дату використання.	Відображення вибраної дати користувачем.
	Натиснення кнопки «Додати запис», коли всі поля заповнені.	На екрані користувача з’являється повідомлення. Дані додаються в таблицю і базу даних.

метадані

Заголовок

Розробка програмного забезпечення для цифрової підтримки діяльності кондитерського підприємства

Автор

Марчук Юлія Науковий керівник / Експерт

підрозділ

King Danylo University

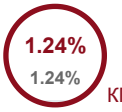
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

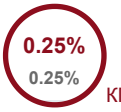
Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	4
Білі знаки	ß	0
Парафрази (SmartMarks)	a	4

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

15083
Кількість слів

112115
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/198/%D0%9E%D0%BD%D1%83%D1%84%D1%80%D0%B0%D0%BA%20%D0%86.%20%D0%86..pdf?sequence=1	37	0.25 %
2	https://dspace.nuft.edu.ua/bitstreams/12558f74-327a-4347-9084-ed509d590d95/download	16	0.11 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/198/%D0%9E%D0%BD%D1%83%D1%84%D1%80%D0%B0%D0%BA%20%D0%86.%20%D0%86..pdf?sequence=1	15	0.10 %
4	http://repository.ukd.edu.ua/bitstream/handle/123456789/391/%D0%9F%D0%B0%D1%85%D0%BE%D0%BB%D1%8C%D1%87%D1%83%D0%BA%20%D0%9E.%D0%A0.%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	14	0.09 %

Microsoft Windows. Саме діалогове вікно для друку дозволить користувачу обрати необхідний йому принтер чи інший вид друку, вказати діапазон сторінок які слід роздрукувати. Якщо користувач погодився з діалоговим вікном друку, документ надсилається на друк.
На рис. 3.1 зображено «Звіт випеченої продукції за день». Звіт для друку має певну різницю від звіту, який бачить користувач в системі. Як було згадано вище, згідно з політикою клієнта на виробництві, працівники-кондитери записують у звіт загальну вагу випеченої продукції одного найменування. Тобто, якщо протягом дня працівник випік 15 разів випічку одного найменування випічки він повинен сам порахувати та записати повну кількість. Саме тому було створено запит, який буде автоматично рахувати загальну вагу випеченої продукції, і кондитеру не доведеться робити це власноруч. Кожен звіт є індивідуальним, тому для кожного звіту включено ім'я працівника та його посада.

Рис. 3. SEQ Рис._3. * ARABIC 1 - Звіт випеченої продукції

«Звіт використаної сировини», який зображено на рис. 3.2, досить схожий до звіту випеченої продукції. Однак у цьому випадку, виконується метод який показує загальну кількість використаної сировини без повторень.

Рис. 3. SEQ Рис._3. * ARABIC 2 - Звіт використаної сировини

«Звіт відправленої продукції», який зображено на рис. 3.3, включає загальну кількість продукції яку магазин замовляв, тип замовлення, самого клієнта та дату відправлення. Згідно з вимогами підприємства, кількість у звіті також записують у кг, а не у штуках. Деталі про кожну випічку яка була відправлена заповнюють у накладній.

Рис. 3. SEQ Рис._3. * ARABIC 3 - Звіт відправленої продукції

У свою чергу звіт «Залишок сировини», що зображено на рис. 3.4, містить більше інформації, а саме: код та найменування сировини, скільки було всього використано за місяць, кількість яка була поставлена на місяць, дата поставки, дата фінального розрахунку, статус, та скільки залишилося сировини згідно з обчисленням. Якщо працівники вказували точну кількість використання сировини протягом місяця, система покаже точні дані, які можна звірити з реальною кількістю залишку сировини.

Рис. 3. SEQ Рис._3. * ARABIC 4- Звіт залишку сировини

3.4 Розробка візуальної частини ПЗ

3.4.1 Бокове меню та UserControls

На рис. 3.5. зображено бокове меню для користувача «Головний кондитер», вміст головної сторінки, аналітики даних для користувача, яка буде відкриватися по замовчуванню, коли користувач буде авторизується в систему.
Для того, щоб уникнути дублювання коду та не прописувати щоразу у кожному вікні, бокове меню було створено в головному файлі, де прописано всі необхідні контейнери, такі як, бокове меню, панель заголовка, а також контейнер для вмісту кожної сторінки.
Для вмісту сторінок було використано тип файлу, як UserControl. UserControl використовується для створення самостійних, багаторазових використовуваних компонентів інтерфейсу користувача для вікна чи сторінки. Він може містити елементи керування, прив'язки даних і обробники подій, що полегшує розробку та підтримку складних інтерфейсів користувача. Також, UserControls можна вкладати один в одного. В головному файлі необхідно створити контейнер, який буде містити всі UserControls. Тому, під час розробки кожного UserControl, увагу зосереджено саме на інформації, яка повинна відобразитись в нашому контейнері.
Для того, щоб переключати сторінки в боковому меню, слід підключити в головному файлі XAML, де знаходиться бокове меню та інші контейнери, які міститимуть UserControls, всі файли інших сторінок типу ViewModel:
17. <Window.DataContext> - це контекст даних для вікна. Він вказує на модель представлення Dashboard_ViewModel_mainCOnfectioner, яка буде використовуватися для прив'язки даних у цьому вікні.
18. <Window.Resources> - це ресурси, які доступні для цього вікна. Вони включають набір шаблонів даних (DataTemplate), які визначають, як саме відобразити різні типи моделей представлення.

Рис. 3. SEQ Рис._3. * ARABIC 5 - Бокове меню та головна сторінка застосунку

Як було згадано вище, DataTemplate визначає, як відобразити певний тип моделі представлення. Наприклад, DasboardDataViewModel буде відображатись за допомогою DashboardDataView. Це означає, що коли користувач використовує DashboardDataViewModel у своєму коді, код автоматично використовує DashboardDataView для відображення цих даних.
Лістинг коду 3.16 - Підключення сторінок для бокового меню застосунку

Для написання логіки переключень між сторінками створюється файл, який знаходиться в папці ViewModel. У файлі знаходиться клас Dashboard_ViewModel_mainConfectioner, який використовується для управління представленнями (views) у застосунку. Файл містить ряд команд (ICommand), які використовуються для переключення між цими представленнями. Кожна команда пов'язана з відповідним методом (ExecuteShow...Command), яка встановлюватиме поточне представлення (CurrentChildView), заголовок (Caption) та іконку (Icon), які відображатимуться в інтерфейсі користувача.
Лістинг коду 3.17 - Підключення сторінок бокового меню у файлі ViewModel

Наприклад, якщо виконується команда ShowDashboardDataViewCommand, вона викликає метод ExecuteShowDashboardDataViewCommand. Цей метод, встановлює CurrentChildView, як новий екземпляр DashboardDataViewModel, Caption як «Головна», а іконка як «IconChar.Microsoft».
Лістинг коду 3.18 - налаштування назви та іконки сторінки у файлі ViewModel

Таким чином, цей клас дозволяє динамічно змінювати представлення, яке відображається користувачу, в залежності від вибраної команди. Це допоможе зберегти чистоту коду та легкість управління представленнями в застосунку.

3.4.2 Розробка статистики для користувача. Діаграми LiveCharts

Сторінка «Статистика користувача» або «Головна» поділена на три секції: дані про зарплату за день, тиждень, місяць та рік без деталей, діаграми та пропонована кількість сировини яку варто замовити на наступний місяць, посилаючись на дані, які користувач вносив протягом місяця про використану сировину.

Для розробки діаграм у застосунку було використано бібліотеку LiveCharts.Wpf. На рис. 3.6 зображено вікно статистики користувача. Дані оновлюються щомісяця. Діаграми відображають дані тільки за поточний місяць, а саме: «Найбільш замовлена продукція місяця», «Найбільш популярні замовлення» та «Найбільш прибуткові партнери». Тому, починаючи з початку місяця користувач отримуватиме нові дані. Щоразу як користувач додаватиме нові дані в системі - дані в діаграмах також будуть оновлюватися.

Рис. 3. SEQ Рис._3. * ARABIC 6 - Сторінка статистики користувача

Щоб відобразити дані у вигляді діаграми було використано два типи діаграм з бібліотеки LiveCharts: стовпчаста (CartessianChart) та кругова (PieChart). Нижче представлений код, який дозволяє згенерувати стовпчасту діаграму. Перш ніж відобразити дані в діаграму, було здійснено підключення до бази даних, після формування SQL-запиту, який знайде дані про зарплату за останні 6 місяців та виконання запиту і заповнення таблиці. Після цих кроків відбувається створення діаграми. Код створює нову діаграму columnChart з використанням бібліотеки LiveCharts.Wpf. Він також створює колекцію даних columnData і список міток labels. Код проходить через кожен рядок в таблиці table, а після додає дані до columnData і Lables, потім додає columnData до columnChart.Series. На останньому кроці діаграма додається до дочірнього елемента salaryForLastSixMonth.

Кольори в діаграмі встановлюються по замовчуванню. Тому при непротребі, це можна проігнорувати. Однак для створення естетичного дизайну діаграм, який буде підходити під стиль застосунку можна задати кольори самостійно, створивши список (List<Brush> colors = new List<Brush>{}) та задати новий колір (new SolidColorBrush((Color)ColorConverter.ConvertFromString(“#”)).

Лістинг коду 3.19 - Генерація діаграми LiveCharts для статистики

Наступний код використовується для отримання даних для третьої секції - «Пропонована кількість сировини для замовлення на наступний місяць». Спершу необхідно визначити період часу, тобто змінні monthStart і monthEnd встановлюють початок і кінець поточного місяця. Після створюється з’єднання з базою даних, формується SQL-запит, а потім виконується SQL-запит і створюється таблиця. На останньому кроці результати прив’язуються до usedRawReport_DailyReportForm.ItemsSource для відображення.

Лістинг коду 3.20 - Обчислення пропонованої кількості сировини для замовлення на наступний місяць

Рис. 3. SEQ Рис._3. * ARABIC 7- Пропонована кількість сировини для замовлення на наступний місяць

3.4.3 Сторінка рецептів. Сторінка окремого рецепту.

Для розробки каталогу рецептів чи будь якого іншого каталогу, XAML файл містить контролер типу ItemControl, який використовується для відображення колекції елементів. Контролер ItemControl відображає колекцію рецептів RecipeCollection. ItemsPanelTemplate відповідає за організацію елементів в ItemsControl. Тут використовується UniformGrid з двома рядками та трьома стовпцями. DataTemplate визначає, як будуть виглядати елементи в ItemsControl. Кожен рецепт представлений як кнопка (Button) з зображенням (Image) та назвою (TextBlock.) Останній крок це Binding. Це механізм, який дозволяє встановити зв’язки між властивостями об’єктів. Source = {Binding Image} означає, що джерело зображення буде взято з властивості Image поточного елемента в RecipeCollection.

Лістинг коду 3.21 - налаштування ItemControl для каталогу рецептів

Після того, як користувач обрав рецепт який хоче подивитися, він повинен отримати його деталі, натискаючи на нього. Насправді файл з окремим рецептом, працює досить подібно, до головного файлу з каталогом рецептів. Однак тепер необхідно отримати правильні дані, які належать конкретному рецепту.

XAML файл описує загальну логіку рецепта, розміщення елементів, стиль тексту, картинки, події. В головному файлі вказуються такі елементи як меню про час приготування торта, інгредієнти, крем та опис приготування. Всі інші дані змінюються залежно від конкретного рецепту, який відкриває користувач.

Рис. 3. SEQ Рис._3. * ARABIC 8 - Сторінка "Рецепти", каталог рецептів

Рис. 3. SEQ Рис._3. * ARABIC 9 - "Сторінка окремого рецепту"

3.4.4 Сторінка «Денний звіт»

Сторінка денний звіт поділена на три секції які також є UserControls. Перша секція «Випечена продукція», яка зображена на рис. 3.10, містить інформацію про випечену продукцію за весь час. Також можна виконати пошук та знайти необхідну інформацію, а після роздрукувати звіт. Доступні дії: редагування, видалення та додавання. Додавання та редагування даних здійснюється в іншому вікні

Друга секція - «Використана сировина». Так само як і секція «Відправлена продукція» секція «Використана сировина» містить інформацію про використану сировину за весь час. Можна виконати пошук за необхідною користувачу датою, а після роздрукувати. Доступні дії: редагування, видалення та додавання. Додавання та редагування відправленої сировини здійснюється в іншому вікні.

Третя секція - «Відправлена продукція». Секція включає дані про всю відправлену випічку за весь час. Серед доступних дій: додавання, видалення, друк звіт та пошук за датою відправлення.

Рис. 3. SEQ Рис._3. * ARABIC 10 - "Сторінка Денний звіт", секція "Випечена продукція"

Після того, як користувач натиснув на кнопку «Додати звіт», відкривається вікно поверх головного, яке менше розміру. Вікно поділене на дві частини: меню, де користувач вводить всі необхідні йому дані про випечену продукцію, і таблиця, в якій можна переглянути дані, які були додані користувачем за поточний день. Це також дозволяє видалити дані, у випадку якщо користувач ввів невірні дані.

Для додавання у меню користувачу необхідно: вибрати з списку назву випічки, ввести кількість у кілограмах, скільки важить торт, обрати дату випікання (по замовчуванню календар показує сьогоднішню дату) та тип замовлення. Попередньо, кондитер має змогу переглянути ціну за кілограм кожного торта. Також в поле «Випечено (кг)» додано перевірку, яка дозволяє вводити тільки десяткові числа. Це забезпечить швидше додавання даних та уникнення небажаних помилок при додавання випеченої продукції.

У розділі 2 та 3 було згадано про те, що такі таблиці як, «Торти», «Сировина», «Магазини» та «Типи замовлення» необхідні для створення списків. Саме завдяки списку у ComboBox (або ж випадаючий список) можна додати список всієї випічки, яку продає підприємство, чи список всієї сировини, яка використовується на підприємстві, чи список всієї клієнтської бази, з якою безпосередньо співпрацює підприємство.

Рис. 3. SEQ Рис._3. * ARABIC 11 - "Вікно додавання даних в звіт випеченої продукції"
Якщо ж користувач, вирішив відредагувати дані з таблиці він отримує вікно, яке зображене на рисунку. Однак для користувача є попередження, що в даному випадку можна редагувати випечену кількість та дату випікання. Такі поля, як «назва випічки», «код» та «тип замовлення» залишаються незмінними.

Рис. 3. SEQ Рис._3. * ARABIC 12 - "Вікно редагування випеченої продукції"
Такі секції як «Використана сировина» та «Відправлена продукція» містять такий самий інтерфейс та логіку як секція «Випечена продукція», однак різниця полягає в заповненні інших даних.
3.4.5 Сторінка Сировина
Сторінка «Сировина» містить дві секції: дані про залишок сировини за місяць, а також секція про поставку сировини.
На рис. 3.13 зображено секцію, що містить дані про залишок сировини, показує стан залишку сировини за поточний місяць. Тобто головний кондитер має змогу переглянути список всієї використаної сировини протягом місяця, а також залишок що залишився і статус сировини. Статус залишку сировини може бути трьох видів:
1. «Добре, залишку не залишилось». Статус цей може бути у тому випадку якщо ми використали повністю нашу сировину і тепер залишок дорівнює 0.
2. «Добре, є залишок». Даний статус буде показано тоді, коли залишок сировини не дорівнює 0 та більше 0.
3. «Недостача». Статус буде показано тоді коли кількість використаної сировини перевищила кількість сировини яку було поставлено на місяць.
Окрім статусу залишку сировини, в таблиці можна переглянути кількість залишеної сировини. Особливо це важливо для випадку 2 та 3. Також користувач має змогу роздрукувати звіт та при потребі звірити з кількістю сировини на виробництві. Якщо користувачі вносили дані правильно протягом місяця та не робили жодних непотрібних дій таблиця покаже точні дані.
Якщо працівнику необхідно знайти дані про залишок сировини за інший місяць, він може здійснити пошук, а після при необхідності роздрукувати.

Рис. 3. SEQ Рис._3. * ARABIC 13 - Сторінка "Сировина", секція "Використана сировина"
Друга секція, що зображена на рис. 3.14 - це «Поставка сировини». Головний кондитер зобов’язаний щомісяця поновлювати дані в системі про поставку сировини. Якщо ж працівник не додав поставку сировини на цей місяць інші працівники не зможуть вносити дані про те що вони використовують цю сировину та отримуватимуть повідомлення про помилку.
Під час додавання даних користувач повинен вказати дату поставки а також дату кінцевого використання. Дата кінцевого використання надзвичайно важлива, оскільки може бути сировина, яку не можна довго тримати на складі, тому її поставляють менше, і відповідно, дата використання менша. Також бувають випадки повторної поставки, якщо сировину передчасно було використано. У такому разі повторної поставки сировини користувач повинен додати новий запис до системи виконуючи ті самі кроки. Після нова кількість буде додана до вже існуючої кількості.

Рис. 3. SEQ Рис._3. * ARABIC 14 - Сторінка "Сировина", секція "Поставка сировини"
3.4.6 Сторінка зарплата
На рис. 3.15 зображено вікно сторінки «Зарплата», яке містить дані про заробітну плату за поточний день, тиждень, місяць та рік. Також користувач має змогу здійснити пошук та знайти заробітну плату за потрібний йому день чи місяць.
Сторінка «Зарплата» розділена на три секції:
1. Поточні дані за день, тиждень, місяць та рік. Дані оновлюються одразу після додавання нових даних користувачем до системи.
2. Дані за поточний день. Дана секція містить більш детальну інформацію про зарплату. Працівник має змогу переглянути всі дані які він додавав протягом дня, та скільки він заробив.
3. Пошук зарплати за заданим днем, місяцем, тижнем чи місяцем. Дана функція допомагає користувачу здійснити пошук необхідних йому даних. Секція розділена на меню з пошуком та результатом даних, а також таблицю де користувач має змогу переглянути всю зароблену зарплату за заданий період.

Раніше, у пункті 3.2.4, було детально описано як відбувається автоматичний розрахунок заробітної плати працівника. Кожен з попередніх пунктів, які було написано вище має ту саму логіку, однак, кожен з них різні типи даних, про які також було згадано у підрозділі 3.1 та пункті 3.2.4.

Рис. 3. SEQ Рис._3. * ARABIC 15 - Сторінка "Зарплата"
3.4.7 Сторінка «Профіль»
Сторінка «Профіль», яку зображено на рис. 3.16, надає користувачам цінну інформацію про їхній обліковий запис. Ця сторінка **розроблена так, щоб бути інтуїтивно зрозумілою та простою у використанні.** Вона призначена тільки для огляду їх персональної інформації, зображення профілю та інших відповідних деталей. Це важлива сторінка, оскільки вона гарантує, що користувачі мають постійний доступ до своєї власної інформації, яка може бути корисна, чи для усунення несправностей, чи інших проблем, пов’язаних з обліковим записом. Користувачі, окрім адміністраторів, не мають змоги редагувати будь-яку інформацію, пов’язану з їхнім профілем. Отримання інформації про користувача відбувається одразу після авторизації в систему. Саме тому користувач отримує точні дані про свою особисту інформацію.

Рис. 3. SEQ Рис._3. * ARABIC 16 - Сторінка "Профіль"
3.4.8 Сторінка «Авторизація»

На рис. 3.17 зображено сторінку авторизації користувача в застосунок - це перший екран, з яким стикаються користувачі під час доступу до програми. Вона розроблена так, щоб вона була зручною та легкою для навігації, із чітким введенням полів імені користувача та пароля, а також кнопкою авторизації.
Кнопка є основною метою сторінки, оскільки вона дозволяє користувача входити в застосунок. Окрім полів введення та кнопки авторизації, сторінка містить логотип застосунку, який знаходиться зверху полів для введення та привітання користувача з поверненням.

Також авторизація включає перевірку правильно введених даних у поля та даними з бази даних. Додатково включені обмеження на кількість символів у полях та типи символів які можна вводити в полях, про які детальніше буде описано у підрозділі 3.5, а саме валідація застосунку.

Рис. 3. SEQ Рис._3. * ARABIC 17 - Сторінка "Авторизація"

3.4.9 Сторінка «Історія дій» та «Користувачі»

Відстеження діяльності користувачів в застосунку вкрай важливо, оскільки це дозволяє перевірити наявність підозрілих дій, про які буде згадано у підрозділі 3.5 - валідація застосунку. Перегляд діяльності користувачів доступна тільки на сторінці адміністратора. Серед дій на даній сторінці, доступний пошук за користувачем та датою, а також видалення даних з таблиці. Таблиця включає назву дії, персональний код користувача, повне ім’я, дату та час виконання дії. Пошук дій за певний день здійснюється за персональним кодом користувача. Вибираючи з випадаючого списку персональний код, адміністратор також поруч в полі «Повне ім’я» може побачити повне ім’я користувача, якому належить персональний код. У списку також доступний вибір як «Всі користувачі», що дозволить здійснити пошук за всіма користувача, які здійснювали будь-які дії протягом заданого дня.

Рис. 3. SEQ Рис._3. * ARABIC 18 - Сторінка "Історія дій"

Окрім відстеження дій користувачів, адміністратор може редагувати дані про існуючих користувачів, видаляти та додавати нових користувачів. Сторінку «Користувачі» зображено на рис. 3.19. Вибираючи з випадаючого списку персональний код, адміністратор отримує дані про конкретного користувача.

Редагування даних відбувається на тій же сторінці. Натискаючи на кнопку редагування, всі необхідні поля будуть активними для редагування. В режимі перегляду поля для редагування неактивні. Для створення паролю створено додаткову функцію, що генерує пароль автоматично.

У випадку видалення користувача з системи, користувач більше не зможе авторизуватися та мати доступ до системи. Додавання нового користувача відбувається в додатковому вікні, яке відкривається поверх головного.

Рис. 3. SEQ Рис._3. * ARABIC 19 - Сторінка "Користувачі"

3.5 Валідація застосунку

Валідація або перевірка є важливим аспектом в розробці застосунку, оскільки вона допомагає переконатися, що введені користувачем дані є точними та відповідають вимогам програми. Без перевірки користувачі можуть ввести неправильні дані або недійсну інформацію, що може призвести до неочікуваних помилок або навіть фатальних збоїв застосунку. Валідація особливо важлива в програмах, які обробляють конфіденційні дані або вимагають точного введення для правильної роботи. Саме тому в застосунку було включено наступні перевірки:

1. Перевірка формату. Під час додавання чи редагування даних є певні поля, в які потрібно вводити інформацію певного типу. Наприклад, найменування випічки повинно містити тільки літери українського алфавіту, кількість випеченої продукції має десяткові числа, а код сировини має включати тільки цілі числа. Нижче представлений код, який відповідає за перевірку формату:

 - Метод numberValidationTextBox. Цей метод перевіряє, чи введене значення є числом від 0 до 9. Якщо введене значення не є числом, то воно не буде оброблено.

 - Метод lettterValidationTextBox. Даний метод перевіряє, чи введене значення є українською літерою. Якщо значення, яке було введене не є українською літерою, то воно не буде оброблено.

 - Метод numberFValidationTextBox. Метод перевіряє, чи введене значення є десятковим числом (наприклад 2.212). Якщо введене значення не є десятковим числом, то воно не буде оброблено. Однак, за винятком випадку, коли введено «.», і воно ще не присутнє в текстовому полі.

Саме для того, щоб перевірити введення символів користувачем, використовується регулярний вираз Regex. Саме він формує шаблон пошуку, і це використовується для знаходження певних рядків або груп рядків у тексті.

Лістинг коду 3.22 - Валідація введених даних

2. Перевірка на діапазон. Під час створення бази даних, кожне поле отримало обмеження на кількість символів. Саме тому, в застосунку необхідно передбачити перевірку на кількість введених символів в поля. Наприклад, код випічки повинен містити тільки 4 цифри, або ім’я користувача при авторизації повинно становити тільки 6 цифр. Тому, щоб уникнути введення зайвих символів в полях використовується властивість в файлі XAML - Max.Length, що відповідає за довжину у полях введення.

3. Перевірка на безпеку. Одна з важливих перевірок, яка включає в себе перевірку на наявність шкідливих або небажаних введень, такі як SQL-ін’єкції. Тому, для того щоб запобігти SQL-ін’єкціям, до кожного SQL-запиту було додано параметри, про які вже було згадано у підрозділі 3.2. Під час використання параметрів, дані, які були введені користувачем ніколи не вставляються безпосередньо в запит, що робить неможливим виконання шкідливого коду. Окрім безпеки, параметризовані запити можуть покращити продуктивність застосунку, а також використання параметрів може зробити SQL код більш читабельним і легких для розуміння. Також, серед заходів безпеки було розглянуто «моніторинг і журналювання». Всі дії які доступні в застосунку, наприклад, додавання чи просто перегляд сторінки, фіксуються в базі даних. Це допоможе перевірити наявність підозрілих дій серед користувачів. Перегляд дій користувачів доступні на сторінці адміністратора, про яку було згадано в підрозділі 3.4.

4. Перевірка на наявність. В застосунку, кожне поле для введення даних є обов’язковим.

В нижче представленому коді, валідація полягає саме у перевірці на наявність підказок для користувача. Якщо користувач не ввів значення в поле «код» («newId») чи назву («newTitle»), підказка залишається незмінною. В цьому випадку, користувач отримує повідомлення про відсутність даних у полях.

Лістинг коду 3.23 - Перевірка на наявність відсутніх даних в полях

5. Перевірка унікальності. В базі даних, є певні поля які вимагають перевірки на унікальність даних, наприклад персональний номер користувача, чи код клієнта. Саме тому, перед додаванням даних включено перевірку, яка буде порівнювати з даними які вже є в системі, і даними, які користувач хоче додати. Нижче представлений код, який виконує перевірку на унікальність даних, на базі таблиці магазини («shops»). Спершу створюється параметризований SQL-запит, який містить команду INSERT для додавання даних. Після

відкривається з’єднання з базою даних та виконується SQL-запит. Після виконується перевірка результату. Якщо count більше 0, то це означає що запис з таким кодом (shop_id) вже існує в базі даних. Користувач отримує повідомлення про помилку, з’єднання з базою даних закривається.
Лістинг коду 3.24 - Перевірка унікальності

6. Підказки для користувачів. Підказки є досить важливими у користуванні, оскільки це дозволить побачити приклад того, які дані необхідно ввести в поле. У наступному блоці коду використовуються події для взаємодії з текстовими полями. Подія this.PreviewMouseDown += IsFieldActive_PreviewMouseDown викликається, коли користувач натискає мишу. Функція IsFieldActive_PreviewMouseDown буде викликана при події. Після встановлюються початкові значення текстових полів назви магазину («txtShopTitle») та коду («txtCode»). Наступна подія .GotFocus викликається, коли текстове поле отримує фокус, тобто коли користувач натискає на нього. При цій події буде викликана функція RemoveText(...). В свою чергу, подія .LostFocus викликається тоді, коли текстове поле втрачає фокус, а саме коли користувач натискає за межами поля. Функція AddText(...) буде викликана при цій події.
Лістинг коду 3.25 - Підказки для користувачів

Як було вище згадано, функція RemoveText(...) викликається при події .GotFocus. Функція перевіряє чи поле містить підказку, в даному випадку «Назва...» для поля назви магазину. Якщо це так, він змінює текст на пустий рядок і повертає фокус на це поле.
Лістинг коду 3.26 - Видалення підказки користувача

Функція AddText(...) викликається коли текстове поле втрачає фокус. Якщо текст у полі пустий або містить лише пробіли, він змінює текст на підказку «Назва...».
Лістинг коду 3.27 - Встановлення підказки для користувача

Код, що подано нижче використовує подію для перевірки, чи є активним текстові поля, і якщо вони активні та пусті, встановлює їх значення на значення за замовчуванням.
Лістинг коду 3.28 - Подія на перевірку активного поля

3.6 Тестування застосунку

Тестування програми - це процес перевірки її відповідності вимогам і очікуваній поведінці. Тестування застосунку має вирішальне значення, оскільки воно допомагає переконатися, що програмне забезпечення вільне від помилок, без помилок і відповідає всім поставленим вимогам. Без тестування майже неможливо гарантувати, що програмне забезпечення працюватиме так, як очікувалося чи призначено, що може призвести до розчарування та зниження довіри користувачів.

У контексті розробки застосунку було обрано ручне тестування. Це рішення було прийнято з урахуванням того, що розробляється невеликий проєкт, і цікавить глибока, і детальна перевірка. Ручне тестування допомагає виявити помилки, які автоматизоване тестування може не виявити, особливо ті, що стосуються взаємодії з користувачем і функціональності. Воно допомагає виявити будь-які проблеми на ранніх стадіях процесу розробки, що може запобігти їх переростанню в більші проблеми пізніше.
Ручне тестування також дозволяє краще зрозуміти, як користувачі повинні взаємодіяти з продуктом, що, в свою чергу допомагає покращити дизайн застосунку і зробити його більш привабливим і зручним для користувачів.
Весь сценарій для ручного тестування подано в додатку Ж.
Висновки до розділу

У третьому розділі ретельно досліджено розробку інформаційної структури даних. Створення таблиць баз даних виконано з урахуванням всіх необхідних вимог до структуризації даних. Необхідні SQL-запити до бази даних розроблені для забезпечення швидкого та ефективного доступу до інформації. Також розглянуто детальну розробку користувацьких звітів. Візуальна частина застосунку розроблена з урахуванням принципів зручності користування. У свою чергу, валідація та тестування застосунку були проведені для забезпечення його надійності та стабільності роботи.

ВИСНОВКИ

В процесі розробки кваліфікаційної роботи було створено програмне забезпечення «Розробка програмного забезпечення для цифрової підтримки діяльності кондитерського підприємства».
В результаті виконаної роботи, було розроблено опис предметної області, методи та засоби спостереження та збирання інформації, що допомогли виявити проблемні точки працівників у роботі на підприємстві. Проаналізовано ринок на наявність аналогічних застосунків, в результаті чого не знайдено жодного програмного забезпечення з такою самою специфікою, як розроблюваний застосунок. Визначено функціональні вимоги, вимоги до дизайну системи, а також вимоги до безпеки та надійності. Спроєктовано діаграму прецедентів, що допомогла зрозуміти як користувач повинен взаємодіяти з системою. В свою чергу, ER-діаграми визначили структуру та організацію даних, а також зв’язки між ними, Діаграма класів дозволила чітко відобразити зв’язки між різними класами та їхніми взаємодіями. Окрім діаграми, розроблено архітектуру та UI-дизайн застосунку в Figma, вибрано технології для розробки. Проєкт також включає детальний опис розробки кожної сторінки застосунку, який був розроблений в Microsoft Visual Studio 2019, автоматичного обчислення зарплати працівника та залишку сировини за місяць. Водночас, розробка застосунку включає валідацію та проведено ручне тестування.
Даний застосунок є цінним для потенційних користувачів, оскільки він вирішує конкретні проблеми, з якими вони стикаються на роботі щодня. Він автоматизує обчислення зарплати, залишку сировини, допомагає підвищити ефективність роботи. В свою чергу, розробка проєкту стала цінним досвідом для мене. Я не тільки удосконалила свої навички з мови програмування C# та MySQL, але і вивчила нові технологію WPF, розробила UI UX дизайн в Figma, навчилася аналізувати потреби користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. UML Use Case Diagram Tutorial. YouTube: веб-сайт. URL: https://youtu.be/zid-MVo7M-E?si=f-_GfZgJZIXPPjL9 (дата звернення: 26.12.2023).
2. Conceptual, Logical & Physical Data Models. YouTube: веб-сайт. URL: https://youtu.be/RJ9TpkWKyU0?si=OAAJceQoUbMh6vqz (дата звернення 28.12.2023).
3. Лавріщева К.М. ПРОГРАМНА ІНЖЕНЕРІЯ: підручник. Київ: Академічна книга, 2008. 319 с. URL:

https://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf (дата звернення 29.12.2023).
4. UML Class Diagrams. YouTube: веб-сайт. URL: https://youtu.be/6XrL5jXmTwM?si=ZDdI1dnlISRqt7kn (дата звернення 01.01.2024).
5. Introducing Figma: A Beginners Tutorial (2023 UI UX Design). YouTube: веб-сайт. URL: https://youtu.be/JGLfyTDgfDc?si=_RKjoDLs1IBUxlbY (дата звернення: 08.01.2024).
6. Figma UX tutorial for beginners - Wireframe. YouTube: веб-сайт. URL: https://youtu.be/D4NyQ5iOMF0?si=tNC5VocDXzVAOcke (дата звернення: 08.01.2024)
7. Figma Learn. Figma Learn: веб-сайт. URL: https://help.figma.com/hc/en-us (дата звернення: 08.01.2024).
8. The Greatest Design System UI Kits for Figma! Full tutorial. YouTube: веб-сайт. URL: https://youtu.be/gnRxVgXsAJM?si=kqNGtVd1wZULqc5g (дата звернення 09.01.2024).
9. MySQL Full Course for free (2023). YouTube: веб-сайт. URL: https://youtu.be/5OdVJbNCSso?si=63k59z78borDZHDD (дата звернення 10.01.2024).
10. MySQL Documentation. Dev MySQL: веб-сайт. URL: https://dev.mysql.com/doc/ (дата звернення 11.01.2024).
11. Мулеба О.Ю. Інформаційні системи та реляційні бази даних: навчальний посібник. Електронне видання, 2018. 118 с. URL: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/19776/1 (дата звернення 13.01.2024).
12. C# WPF Tutorial. YouTube: веб-сайт. URL: https://youtube.com/playlist?list=PLih2KERbY1HHOOJ2C6FOrVXIwg4AZ-hk1&si=a9aKkd1Iqso6nDRV (дата звернення: 20.01.2024).
13. WPF in Visual Studio 2019 | Getting Started. YouTube: веб-сайт. URL: https://youtu.be/73PsdMMINRk?si=pztNhIM37xkzv1cX (дата звернення: 20.01.2024).
14. C# GUI Tutorial using WPF | XAML | - Windows Presentation Foundation. YouTube: веб-сайт. URL: https://youtu.be/oSeYvMEH7jc? (дата звернення 21.01.2024).
15. Connection to a Database C# - WPF and MySQL. YouTube: веб-сайт. URL: https://youtu.be/OPDPI5exPp8?si=xGMJBS7PjNhX2PQa (дата звернення 22.02.2024).
16. Quick start: Install and use a NuGet package in Visual Studio. Learn Microsoft: веб-сайт. URL: https://learn.microsoft.com/uk-ua/nuget/quickstart (дата звернення 25.02.2024).
17. LiveCharts2 Docs. LiveCharts2: веб-сайт. URL: https://livecharts.dev/docs/wpf/2.0.0-rc2/gallery (дата звернення 25.02.2024).
18. C# Language Documentation. Microsoft Learn: веб-сайт. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення 28.02.2024).
19. Система Дистанційної Освіти УКД. СДО УКД: веб-сайт. URL: https://online.ukd.edu.ua/. (дата звернення 24.03.2024).
20. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ щодо виконання кваліфікаційних робіт для здобувачів освітнього рівня бакалавр за освітньою програмою «Розробка та тестування програмного забезпечення спеціальності 121 «Інженерія програмного забезпечення» У ЗВО «Університет Короля Данила». Івано-Франківськ: ЗВО «Університет Короля Данила», 2021. 56 с. (дата звернення 22.03.2024).
21. Windows Presentation Foundation documentation. Microsoft Learn: веб-сайт. URL: https://learn.microsoft.com/en-us/dotne (дата звернення 25.03.2024).
22. XAML Syntax in Detail. Microsoft Learn: веб-сайт. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/advanced/xaml-syntax-in-detail? (дата звернення: 26.03.2024).

ДОДАТКИ

Додаток А

Розробка діаграми прецедентів для підрозділу 2.2.

Рис. А. SEQ Рис._А. * ARABIC 1 - Діаграма прецедентів системи ведення звітів кондитерського підприємства

Додаток Б

Моделювання концептуальної моделі для пункту 2.3.1 «Концептуальна модель даних».

Рис. Б. SEQ Рис._Б. * ARABIC 1 - Концептуальна модель даних

Додаток В

Розробка логічної моделі даних для пункту 2.3.2 «Логічна модель даних».

Рис. В. SEQ Рис._В. * ARABIC 1 - Логічна модель даних

Додаток Г.1

Генерація фізичної моделі даних бази даних в середовищі MySQL Workbench для пункту 2.3.3.

Рис. Г. SEQ Рис._Г. * ARABIC 1 - Фізична модель даних

Додаток Г.2

Таблиця Г.2.1

«Атрибути таблиць бази даних застосунку»

Таблиця	Атрибут	Опис атрибуту
users	user_id	ID користувача
	user_name	Ім’я
	user_surname	Прізвище
	username	Ім’я користувача в системі, його персональний номер.
	user_password	Пароль
	user_role	Роль (адміністратор, головний кондитер, кондитер)

user_age Вік
user_birthday День народження
user_email Електронна пошта
user_phone Номер телефону
user_country Країна
user_city Місто
user_region Область
user_postal_code Поштовий код
cakes cake_id ID торта
cake_title Найменування торта
cake_price_kg Ціна за кілограм

raw raw_id ID сировини
raw_title Найменування сировини

shop shop_id ID магазину
shop_title Найменування магазину
order_type order_type_id ID типу замовлення
order_type_title Найменування типу замовлення
daily report cakes daily_report_id ID
cake_title Найменування торта
cake_price_kg Ціна за кілограм
quantity_kg Вага випеченого торта
baking_date День виготовлення
order_type Тип замовлення
confectioner Кондитер
cake_status Статус продукції
sent_to_shop Магазин, куди відправлено продукцію
date_sent Дата відправки
earned_per_day Зароблено за день
raw obtaining raw_obtaining_id ID дати поставки сировини
raw_id ID сировини
raw_title Найменування сировини
raw_quantity Кількість використаної сировини
obtaining_date Дата поставки
final_calculation_date Дата до якої необхідно рахувати залишок сировини
used raw report used_raw_id ID використаної сировини
raw_id ID сировини
raw_title Найменування сировини
used_raw_quantity Кількість використаної сировини
used_raw_date Дата використаної сировини
confectioner Кондитер
raw_quantity_per_month Кількість сировини дана на місяць
obtaining_date Дата поставки сировини
final_calculation_date Дата до якої необхідно рахувати залишок сировини
raw_remaining_quantity Кількість залишку сировини

Кінець таблиці Г.2.1 - Атрибути таблиць бази даних

Додаток Д

Розробка діаграми класів застосунку для підрозділу 2.4 «Проєктування діаграми класів».

Рис. Д. SEQ Рис._Д. * ARABIC 1 - Діаграма класів

Додаток Е

Розробка каркасів для розроблюваного застосунку для підрозділу 2.6 «Розробка UI-дизайну в Figma».

Рис. Е. SEQ Рис._Е. * ARABIC 1 - Wireframes (Каркаси) застосунку

Додаток Є

Лістинг коду Є.1 - Генерація звіту випеченої продукції кондитера для підрозділу 3.3.

```
private void btnPrintData_Click(object sender, RoutedEventArgs e) {
bindDailyReportTableForPrinting();
if (bakedCakesReport_DailyReportForm.Items.Count == 0)
{
MessageBox.Show("Немає даних для друку");
return;
```

```
}
string confectioner = LoginViewModel.UserInfo;
string _userRole = LoginViewModel.UserPosition;
//Creating of new FlowDocument
FlowDocument doc = new FlowDocument();
doc.PageWidth = 842;
doc.PageHeight = 595;
doc.PagePadding = new Thickness(30);
doc.ColumnWidth = Double.PositiveInfinity;
doc.FontFamily = new System.Windows.Media.FontFamily("Arial");
// Adding of current date and time
Paragraph currentTime = new Paragraph(new Run("Дата та час генерації звіту: " + DateTime.Now.ToString()));
currentTime.FontSize = 9;
currentTime.TextAlignment = TextAlignment.Right;
doc.Blocks.Add(currentTime);
//Adding header
Paragraph title = new Paragraph(new Run("Звіт випеченої продукції за "
  + dateOfBaking.SelectedDate.Value.ToString("dd MMMM yyyy") + " р."));
title.FontSize = 14;
title.TextAlignment = TextAlignment.Left;
doc.Blocks.Add(title);
//Add the confectioner's name
Paragraph confectionerName = new Paragraph(new Run("Кондитер: " + confectioner));
confectionerName.FontSize = 12;
confectionerName.TextAlignment = TextAlignment.Right;
doc.Blocks.Add(confectionerName);
//Add the confectioner's role
Paragraph confectionerRole = new Paragraph(new Run("Посада: " + _userRole));
confectionerRole.FontSize = 12;
confectionerRole.TextAlignment = TextAlignment.Right;
doc.Blocks.Add(confectionerRole);
//Creating of new table
Table table = new Table();
table.CellSpacing = 0;
table.FontSize = 12;
table.BorderBrush = System.Windows.Media.Brushes.Black;
table.BorderThickness = new Thickness(0.5);
// Set columns' width
int numOfColumns = 8;
for (int i = 0; i &lt; numOfColumns; i++)
{
  TableColumn column = new TableColumn();
  column.Width = new GridLength(1, GridUnitType.Auto); // Змінено на Auto
  table.Columns.Add(column);
}
// Creating and adding header of table
TableRowGroup headerGroup = new TableRowGroup();
TableRow headerRow = new TableRow();
headerRow.Cells.Add(CreateCell("Код"));
headerRow.Cells.Add(CreateCell("Випічка"));
headerRow.Cells.Add(CreateCell("Всього випечено"));
headerRow.Cells.Add(CreateCell("Ціна за кг"));
headerRow.Cells.Add(CreateCell("Дата випікання"));
headerGroup.Rows.Add(headerRow);
table.RowGroups.Add(headerGroup);
// Add data from Datagrid to the table
TableRowGroup itemGroup = new TableRowGroup();
foreach (DailyReport item in bakedCakesReport_DailyReportForm.Items)
{
  TableRow row = new TableRow();
  row.Cells.Add(CreateCell(item.DailyReportId.ToString()));
  row.Cells.Add(CreateCell(item.CakeTitle.ToString()));
  row.Cells.Add(CreateCell(item.QuantityKg.ToString()));
  row.Cells.Add(CreateCell(item.CakePriceKg.ToString()));
  row.Cells.Add(CreateCell(item.BakingDate.ToString("dd.MM.yyyy")));
itemGroup.Rows.Add(row);
}
```

```
table.RowGroups.Add(itemGroup);
//Add table to FlowDocument
doc.Blocks.Add(table);
// Creating of new PrintDialog
PrintDialog printDialog = new PrintDialog();
if (printDialog.ShowDialog() == true)
{ //Show print window             printDialog.PrintDocument(((IDocumentPaginatorSource)doc).DocumentPaginator, "Printing Document");
}
```

Додаток Ж

Таблиця Ж.1 для підрозділу 3.6.

«Сценарій ручного тестування застосунку»

No		
1	Заголовок	Авторизація користувача
	Крок	Очікуваний результат
	Ввід значення в поле «Персональний номер»	Введене значення відобразиться в полі «Персональний номер».
	Ввід значення в поле «Пароль»	Введене значення відобразиться в полі «Пароль», приховуючи пароль користувача під символами.
	Натиснення кнопки «Увійти», коли дані введено неправильно	На екрані користувача з'являється повідомлення про те що дані введено неправильно.
	Натиснення кнопки «Увійти», коли дані введено правильно	Відкриття головної сторінки працівника.
2	Заголовок	«Редагувати дані - Додавання нової випічки»
	Крок	Очікуваний результат
	Введення в поле «Код»	Введене значення відобразиться в полі «Код»
	Введення в поле «Назва випічки»	Введене значення відобразиться в полі «Назва випічки»
	Введення в поле «Ціна за кг»	Введене значення відобразиться в полі «Ціна за кг»
	Натиснення кнопки «Додати запис», якщо всі поля заповнені.	На екрані користувача з'являється повідомлення про те що нову випічку успішно додано. Нова випічка добавляється до таблиці.
	Натиснення кнопки «Додати запис», якщо поля (чи поле) не заповнено.	На екрані користувача з'являється повідомлення про те що виникла помилка.
3	Заголовок	Редагувати дані - Видалення випічки
	Крок	Очікуваний результат
	Натиснення кнопки «Видалити запис», коли отримані дані отримано коректно.	На екрані користувача з'являється про успішне видалення. Успішне видалення даних з таблиці, бази даних.
	Натиснення кнопки «Видалити запис», коли з рядка таблиці неможливо отримати дані.	На екрані користувача з'являється повідомлення про помилку.
4	Заголовок	Редагувати дані - Змінення найменування випічки
	Крок	Очікуваний результат
	Натиснення кнопки «Змінити запис», коли дані з рядка успішно отримано.	На екрані користувача з'являється нове вікно для редагування даних.
	Натиснення кнопки «Змінити запис», коли дані з рядка не отримано	Вікно редагування даних не відкриється, отримання повідомлення про помилку
5	Заголовок	Редагувати дані - Додавання нової сировини
	Крок	Очікуваний результат
	Введення в поле «Код»	Введене значення відобразиться в полі «Код»
	Введення в поле «Найменування сировини»	Введене значення відобразиться в полі «Найменування сировини»
	Натиснення кнопки «Додати запис», якщо всі поля заповнені.	На екрані користувача з'являється повідомлення про те що нову сировину успішно додано. Нова сировина добавляється до таблиці.
	Натиснення кнопки «Додати запис», якщо поля (чи поле) не заповнено.	На екрані користувача з'являється повідомлення про те, що виникла помилка.
6	Заголовок	Редагувати дані - Видалення сировини
	Крок	Очікуваний результат
	Натиснення кнопки «Видалити запис», коли дані з рядка отримано успішно.	На екрані користувача з'являється про успішне видалення. Успішне видалення даних з таблиці, бази даних.
	Натиснення кнопки «Видалити запис», дані з рядка таблиці не отримано.	На екрані користувача з'являється повідомлення про помилку.
7	Заголовок	Редагувати дані - Змінення найменування сировини
	Крок	Очікуваний результат
	Натиснення кнопки «Змінити запис», коли дані з рядка не отримано	Вікно для редагування відкрито не буде.
	Натиснення кнопки «Змінити запис», коли дані з рядка отримано	На екрані користувача з'являється нове вікно для редагування даних.
8	Заголовок	Редагувати дані - Додавання нового магазину
	Крок	Очікуваний результат
	Введення в поле «Код»	Введене значення відобразиться в полі «Код»
	Введення в поле «Найменування магазину»	Введене значення відобразиться в полі «Найменування магазину»
	Натиснення кнопки «Додати запис», якщо всі поля заповнені.	На екрані користувача з'являється повідомлення про те що новий магазин успішно додано. Новий магазин додається до таблиці.

Натиснення кнопки «Додати запис», якщо поля не заповнено. На екрані користувача з'являється повідомлення про те що виникла помилка.

9 Заголовок Редагувати дані - Видалення магазину

Крок Очікуваний результат

Натиснення кнопки «Видалити запис», коли дані з рядка отримано. На екрані користувача з'являється про успішне видалення. Успішне видалення даних з таблиці, бази даних.

Натиснення кнопки «Видалити запис», коли дані з рядка не отримані. На екрані користувача з'являється повідомлення про помилку.

10 Заголовок Редагувати дані - Змінення найменування магазину

Крок Очікуваний результат

Натиснення кнопки «Змінити запис», коли дані з рядка не отримано Вікно для редагування не буде відкрито

Натиснення кнопки «Змінити запис», коли дані з рядка отримано На екрані користувача з'являється нове вікно для редагування даних.

11 Заголовок Додавання випеченої продукції

Крок Очікуваний результат

Вибрати необхідну випічку з ComboBox Відображення вибраної продукції користувачем.

Введення вагу продукції в поле «К-сть (кг)» Введене значення відобразиться в полі «К-сть (кг)»

Вибрати необхідну дату випікання. Відображення вибраної дати користувачем.

Вибрати тип замовлення з ComboBox Відображення вибраного типу замовлення.

Натиснення кнопки «Додати запис», коли всі поля заповнені. На екрані користувача з'являється повідомлення. Дані додаються в таблицю і базу даних.

12 Заголовок Видалення випеченої продукції

Крок Очікуваний результат

Натиснення кнопки «Видалити запис», коли дані з рядка отримано На екрані користувача з'являється про успішне видалення. Успішне видалення даних з таблиці, бази даних.

Натиснення кнопки «Видалити запис», коли дані з рядка не отримано На екрані користувача з'являється повідомлення про помилку.

13 Заголовок Додавання використаної сировини

Крок Очікуваний результат

Вибрати необхідну сировину з ComboBox Відображення вибраної сировини користувачем.

Введення вагу сировини в поле «К-сть (кг)» Введене значення відобразиться в полі «К-сть (кг)»

Вибрати необхідну дату використання. Відображення вибраної дати користувачем.

Натиснення кнопки «Додати запис», коли всі поля заповнені. На екрані користувача з'являється повідомлення. Дані додаються в таблицю і базу даних.

Кінець таблиці Ж.1 - «Сценарій ручного тестування застосунку»