АНОТАЦІЯ

Цей проект - це програмне забезпечення для управління рестораном, яке надає зручні інструменти для керування меню, замовленнями та іншими аспектами роботи ресторану. Програма дозволяє додавати, видаляти та редагувати інформацію про продукти, страви та замовлення, а також проводити аналіз статистики роботи ресторану.

Функціонал:

1. Управління продуктами: можливість додавати, видаляти та редагувати інформацію про продукти на складі, включаючи їх назву, вартість, кількість тощо.

2. Управління стравами: можливість створювати нові страви з наявних продуктів, встановлювати їх вартість та вираховувати собівартість.

3. Управління замовленнями: можливість приймати та обробляти замовлення від клієнтів, враховуючи дату, столик, обрані страви тощо.

4. Генерація чеків: можливість формувати чеки для клієнтів на основі їх замовлень, включаючи загальну вартість та іншу інформацію.

5. Аналіз статистики: можливість отримати статистику про роботу ресторану, таку як кількість замовлень, найбільш популярні страви тощо.

Технічні деталі:

- Мова програмування: C++/CLI
- Інтегроване середовище розробки: Visual Studio
- Система управління базами даних: MySQL
- Інтерфейс користувача: Windows Forms

Цей проект допомагає оптимізувати робочі процеси ресторану, полегшує управління ресурсами та дозволяє підвищити ефективність обслуговування клієнтів.

ANNOTATION

This project is restaurant management software that provides convenient tools for managing menu, orders, and other aspects of restaurant operations. The program allows adding, deleting, and editing information about products, dishes, and orders, as well as analyzing restaurant work statistics.

Features:

1. Product Management: Ability to add, delete, and edit information about products in stock, including their name, price, quantity, etc.

2. Dish Management: Ability to create new dishes from available products, set their price, and calculate cost.

3. Order Management: Ability to accept and process orders from customers, considering date, table, selected dishes, etc.

4. Check Generation: Ability to generate checks for customers based on their orders, including total cost and other information.

5. Statistics Analysis: Ability to get statistics about restaurant operation, such as the number of orders, most popular dishes, etc.

Technical Details:

- Programming Language: C++/CLI
- Integrated Development Environment: Visual Studio
- Database Management System: MySQL
- User Interface: Windows Forms

This project helps optimize restaurant workflows, facilitates resource management, and improves customer service efficiency.

ЗМІСТ

Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розроб.		*Гайченя О.О*			***тема***	*Літ.*	*Арк.*	*Аркушів*
Перев.							*5*	*87*
Реценз.					*Створення інформаційної системи обліку та обігу продукції в готельно ресторанному бізнесі*			
Н. контр.						*КНУБА,УП-41*		
Затверд.								

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – Програмне Забезпечення

ОС - Операційна система

БД - база даних

RFC - від англ. Request for Comments Запит коментарів, Прохання прокоментувати

RPC - від англ. Remote Procedure Call віддалений виклик процедур

D-COM - від англ. Distributed Component Object Model модель об'єктів

REST – від англ. Representational state transfer, стиль архітектури програмного забезпечення для розподілених систем

SQL - від англ. Structured Query Language - мова структурованих запитів

GUI – Graphical User Interface

UML – Unified Modelling Language

Звичайно, ось розшифрування цих абревіатур:

MySQL - MySQL - це відкрите вільне програмне забезпечення для керування базами даних, яке зазвичай використовується для зберігання даних в реляційних базах даних.

VS IDE - Visual Studio Integrated Development Environment - це інтегроване середовище розробки, розроблене компанією Microsoft для розробки програмного забезпечення для платформ Microsoft Windows, а також для веб-сайтів, веб-додатків та мобільних додатків.

C++ - Це програмна мова програмування загального призначення.

CLI - Common Language Infrastructure - це специфікація, розроблена Microsoft, яка описує середовище виконання програм.

CRUD - Create, Read, Update, Delete - це набір основних операцій у роботі з базами даних. Вони використовуються для керування даними: додавання нових записів (Create), отримання інформації про існуючі записи (Read), оновлення існуючих записів (Update) та видалення записів (Delete).

ВСТУП

Швидкий розвиток інформаційних технологій та вдосконалення комп'ютерних технологій призвели до їх глобальної інтеграції у всіх сферах людської діяльності. Сфера мереж ресторанів не є винятком.

Зміни організації та технології роботи підприємств сфери обслуговування, насамперед ресторанів та великих розважальних центрів, викликані необхідністю адаптації до ринкових умов економіки та жорсткої конкуренції, потребують відповідного перебудови інформаційного та документаційного забезпечення управління. В основі сучасних технологій інформаційного забезпечення галузі в цілому та суб'єктів господарювання лежить розробка сучасних економічних інформаційних систем. Сучасні інформаційні технології пропонують широкий набір способів реалізації таких програмних комплексів.

Проблема розробки автоматизованих систем управління для таких організацій, як ресторани та кафе, досі не була предметом спеціального загального дослідження. Здебільшого у літературі розглядаються загальні питання організації інформаційних систем та систем документообігу на підприємствах такого профілю, а також проводиться аналіз окремих технологічних розробок. З широкого кола різних публікацій з проблем автоматизації процесів управління та документообігу найбільш привабливими для даної теми видаються останні науково-методичні розробки з питань проектування систем корпоративного електронного документообігу, серед яких виділяється робота М. В. Бобилєвої [6], колективна робота Д А. Романова, Т. Н. Ільїної, А. Ю. Логінової [31], нещодавно вийшли з друку книги А. В. Даниліна [10] і робота В. А. Конявського та В. А. Гадасіна [20], наукова доповідь Н. Н. Куняєва [21], перекладне видання книги американського фахівця Майкла Дж. Саттона [22]. У цих роботах містяться погляди на електронний документообіг та різні елементи системи електронного обміну інформацією, засновані на використанні комп'ютерних інформаційних технологій, викладаються методичні основи їх впровадження, узагальнений

вітчизняний та зарубіжний досвід постановки завдань та розробки рішень у даній галузі, що потребують інтеграції різних інформаційних платформ та апаратно -програмних засобів.

Особливо актуальним питанням для ресторанів і кафе, особливо великих, є завдання автоматизованого формування вичерпної бази даних всіх пунктів меню, що пропонуються фірмою. Важливим завданням, рішення якого здатне забезпечити для ресторану суттєву конкурентну перевагу, є можливість формування меню та замовлення на продукти за попередніми запитами клієнтів. Це може бути забезпечене тільки при роботі з базою даних з технологічних карт ресторану та базою даних, що містить значний статистичний масив. База має бути відкритою всім зацікавлених користувачів, як менеджерів фірми, і фахівців.

Таким чином, для представленої дипломної роботи визначено об'єкт, предмет та мету дослідження.

Об'єктом дипломного дослідження система автоматизації виробничо-технологічних процесів у ресторані.

Предметом дипломного дослідження є методика розробки та технічної реалізації економічної інформаційної системи для меню, замовлення на поставку продуктів на підставі оперативної та аналітичної інформації бази даних, що передбачає зберігання всієї вихідної інформації в структурованій реляційній базі даних.

Мета роботи полягає у розробці системи автоматизації виробничо-технологічних процесів та управління проєктами для ресторану на основі поставленого технічного завдання на розробку та оцінка економічної ефективності розробки та впровадження системи.

Основна перевага автоматизації - це скорочення надмірності даних, що зберігаються, а отже, економія обсягу використовуваної пам'яті, зменшення витрат на багаторазові операції оновлення надлишкових копій і усунення можливості виникнення протиріч через зберігання в різних місцях відомостей про один і той же об'єкт, збільшення ступеня достовірності інформації та

збільшення швидкості обробки інформації; надмірна кількість внутрішніх проміжних документів, різних журналів, папок, заявок і т. д., повторне внесення однієї і тієї ж інформації до різних проміжних документів. Також значно скорочує час автоматичний пошук інформації, що виготовляється зі спеціальних екранних форм, у яких вказуються параметри пошуку об'єкта. Оперативне управління господарськими процесами складає від одного до декількох днів і реалізує реєстрацію подій, наприклад оформлення та моніторинг виконання замовлень, прихід та витрата продуктів тощо. документів відповідно до чітко визначених алгоритмів. Результати виконання господарських операцій реєструються у відповідних журналах. Автоматизація цих процесів дозволить зберігати інформацію в одній базі, інформація в яку вводиться за допомогою зручного інтерфейсу.

Методи дослідження, що застосовуються у роботі:

- Кабінетні дослідження (робота з вторинною інформацією)
- Опитування клієнтів та співробітників ресторану
- Спостереження

Дипломна робота складається з аналітичної, технологічної та проектної частин. У дипломній роботі також розглянуто економічне обґрунтування системи, що розробляється, та питання охорони праці.

1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ІНФОРМАЦІЙНИХ СИСТЕМ ОБЛІКУ ТА ОБІГУ ПРОДУКЦІЇ

1.1 Стандарти управління проектами

Стандарти управління проектами містять методологію управління проектами. Національні стандарти застосовуються в межах однієї країни або отримали загальнонаціональний статус у процесі розвитку. Міжнародні стандарти мають міжнародне значення або призначені для використання на міжнародному рівні.

Корпоративні — це ті, які застосовуються в межах однієї компанії або групи родинних компаній; громадські — це знання, які є доступними для приватних осіб, компаній чи установ; і громадські — це знання, які пропонуються для вільного використання.

Міжнародні стандарти представляють собою повну систему, яка включає опис стандартів управління проектами, навчання, тестування, аудит, консалтинг та інші компоненти. Найвідоміші міжнародні стандарти управління проектами не існують.

1. Американський інститут управління проектами (PMI) розробив стандарт, відомий як «Базис знань управління проектами» (PMBOK). Приблизно раз на чотири роки цей стандарт переглядається. Редакція, яка є однією з найпоширеніших, вийшла в 2000 році. Однак четверта версія стандарту, The Guide to the PMBOK, 4th Edition, вийшла наприкінці 2008 року. Американський національний інститут стандартів (ANSI) був першим, хто прийняв стандарт як національний стандарт у Сполучених Штатах. Зараз він набув світового визнання.

Стандарт базується на процесному підході до управління проектами. Представимо тривимірний простір для загальної кількості потенційних процесів. Виміри, які згадуються в рамкових стандартах, розташовані по осях координат. Інші можуть бути запропоновані, такі як рівні управління та періоди. Кожна

точка цього простору містить основні процедури управління. «Планування ризиків на стадії впровадження системи», наприклад.

Процедури управління проектами можна побудувати на основі вибору елементарних процесів.

Стандарт містить загальні принципи та підходи проектного менеджменту, формалізовані та структуровані таким чином, щоб вони могли використовуватися для більшості проектів. Управління інтеграцією проекту (управління інтеграцією проекту); управління змістом проекту (управління змістом проекту); управління термінами проекту (управління термінами проекту); управління вартістю проекту (управління вартістю проекту); управління якістю проекту (управління якістю проекту); управління людськими ресурсами проекту (управління людськими ресурсами проекту).

Кожна область знань включає окремі процедури, які виконують менеджери на тому чи іншому етапі виконання проекту.

Стандартний процесно-орієнтований підхід до управління проектами включає детальний, формальний опис вхідних документів і даних, необхідних менеджеру для завершення процесу, а також методів і ресурсів, які він може використовувати для завершення процесу.

2. Система міжнародних вимог до компетентності менеджерів проектів визначається IPMA Competence Baseline (ICB). Міжнародна асоціація IPMA — некомерційна професійна організація, зареєстрована у Швейцарії — розробила цей стандарт. Основною метою цієї організації є сприяння розвитку та широкому застосуванню культури, технологій, методів, засобів і методів проектного управління в різних країнах світу.

На його основі всі країни-члени IPMA розробляють власні системи вимог до компетентності фахівців. Національні системи стандартів повинні відповідати ICB IPMA та офіційно затверджуватися або ратифікуватися відповідними уповноваженими органами IPMA. Він служить основою для розробки національних склепінь знань для 32 країн-членів IPMA; наразі 16 країн мають затверджені національні склепіння знань, відповідні ICB.

ICB використовує компетентнісний, діяльнісний підхід, а не PMBOK. Це означає, що він визначає галузі кваліфікації та компетентності в управлінні проектами, а також принципи оцінки кандидатів на отримання сертифікату. У ICB є 42 елементи, з яких 28 основних і 14 додаткових, які визначають галузі знань, майстерності та професійного досвіду, необхідних для управління проектами.

У програмі є чотири рівні сертифікації: директор проектів (CPD), керуючий проектом (CPM), професійний керівник проектів (RPMP) і фахівець з управління проектами (PMF).

Французька, німецька та англійська — це три мовні варіанти ICB. Багато національних розробок, таких як Body of Knowledge of APM (Великобританія); Bewertungsstruktur, VZPM (Швейцарія); PM-Kanon, PM-ZERT/GPM (Німеччина); Criteres d'analyse, AFITEP (Франція), послужили основою.

Кожна національна асоціація, що входить до IPMA, відповідає за створення та затвердження власних Національних вимог з компетентності (National Competence Baseline, або NCB), з посиланням на ICB та відповідно до них, з урахуванням культури та особливостей країни. Спеціальний комітет IPMA перевіряє національні вимоги на відповідність ICB та основним критеріям сертифікації відповідно до стандарту EN 450131.

Міжнародні організації займаються розробкою стандартів проектного менеджменту, щоб допомогти менеджерам проектів у всьому світі розвивати навички, компетенції, знання, навички та навички. Отже, вони уточнюють, визначають, записують практики управління проектами та створюють єдині стандарти.

1.2 Формування вимог до програмного продукту

База даних (БД в подальшому) розгортається на комп'ютері разом з інформаційною системою. Дана БД буде складатися з декілька стуностей, а саме:

- Продукти;

- Страви;
- Замовлення.

Ці сутності взаємопов’язані між собою, саме від них буде відштовхуватися майбутня система. Зберігатимемо деякі дані, такі як кількість вартість, назва і тп.

З вищесказаного випливає що потрібно відмітити основні вимоги до програмного продукту:

База даних:

- Таблиця "Продукти": містить інформацію про продукти (назва, вартість, кількість на складі тощо).
- Таблиця "Страви": містить інформацію про страви (назва, склад, ціна).
- Таблиця "Замовлення": зберігає дані про замовлення (дата, номер столика тощо).

Проміжні таблиці для зв'язків багато до багатьох між "Продуктами" і "Стравами" та між "Стравами" і "Замовленнями".

Система виведення даних:

- Форма з таблицею, де користувач може переглядати дані про продукти, страви та замовлення.
- Можливість додавання, видалення та редагування записів в таблицях.
- Кнопки для виконання певних запитів, наприклад, виведення інформації про найбільш популярні страви або витрати за певний період.

Форма для оформлення замовлення, де користувач може вибирати страви зі списку, вказувати їх кількість та інші параметри.

Оформлення замовлення:

- Форма для введення даних про замовлення (дата, номер столика).
- Можливість вибору страв зі списку та вказання кількості кожної страви.
- Автоматичний розрахунок вартості замовлення на основі обраних страв.

Управління складом:

- Автоматичне віднімання продуктів зі складу при оформленні замовлення.

- Можливість відслідковувати кількість продуктів на складі та вчасно поповнювати їх запаси.

Калькулятор для створення страв:

- Форма для створення нової страви, де користувач може обирати продукти зі списку, вказувати їх кількість та обчислювати вартість страви.

Це загальний опис функціональності програмного продукту. Детальніше можна розробити кожен пункт, визначивши точні вимоги до інтерфейсу користувача, логіку роботи та алгоритми обробки даних.

З вищесказаного випливає, що потрібно розробити зручний користувацький інтерфейс для осіб різнового вік, а саме можна виділити наступні фактори:

– Читабельність тексту;

– Зрозумілість у використанні;

– Легкий функціонал;

– Радісна гамма кольорів;

– Можливість зберігання списків у зручному форматі.

Відштовхуючись від поставлених завдань потрібно розбити розробку на три стадії, а саме:

1. розробка БД та її наповнення;
2. зв’язування БД та інформаційної системи;
3. розробка функціоналу додатку.

Що спростить розробку та зведе до мінімізування помилок під час проектування, визначено наступні форми:

Форма виводу даних у вигляді таблиці:

- Розробка інтерфейсу користувача, що дозволить виводити дані з бази даних у вигляді таблиці.

- Реалізація функціоналу додавання, видалення та редагування записів у таблиці.

- Забезпечення можливості сортування та фільтрації даних за різними параметрами.
- Додавання можливості пагінації для великих обсягів даних.
- Вивід звітів та статистики на основі даних з бази.

Форма оформлення замовлення:

- Розробка інтерфейсу для оформлення нового замовлення, включаючи вибір страв і їх кількості, обрання столика та офіціанта.
- Обчислення загальної вартості замовлення на основі обраних страв та їх кількості.
- Забезпечення автоматичного оновлення залишків продуктів на складі після підтвердження замовлення.
- Можливість додавання знижок або акцій до замовлення.

Форма калькулятора для створення страв:

- Розробка інтерфейсу, що дозволить користувачам створювати нові страви шляхом вибору необхідних продуктів та введення їх кількості.
- Обчислення вартості страви на основі вартості продуктів та їх кількості.
- Можливість зберігання створених рецептів та використання їх у майбутньому для оформлення замовлень.

Ці елементи допоможуть створити повноцінну систему обліку та управління замовленнями в ресторані, що буде забезпечувати зручну роботу персоналу та задоволення потреб клієнтів.

1.3 Завдання і цілі сучасних ПЗ

З кожним роком сучасні технології розвиваються, розробка інформаційних систем також не стоїть на місці. Їх реалізація сприяє покращенню комунікативності, якості спілкування, вирішення різноманітних складних завдань. Хорошим професіоналам це приносить досить високий прибуток.

Розробка таких систем є досить непростим та довгим процесом. Додатки, аплікації, інформаційні ситсеми або програмне забезпечення розробляються для

всіх платформ, ОС, окремих пристроїв і тд. Існує декілька основних етапів розробки, без яких процес буде неможливим.

По-перше, потрібно вибрати платформу (Windows, Linux, MacOs, Android, DEX). Можна вибирати відразу декілька платформ, але для цього необхідно збільшувати бюджет.

По-друге, треба визначитися з основною метою створення додатку та його актуальністю.

Наступним етапом є створення макету додатку. Кожна деталь є досить важливою, адже упустивши одну, на перший погляд незначну, доведеться переробляти все.

Далі розробляється дизайн додатку. Він повинен враховувати, насамперед, головну ціль створення, цільову аудиторію. Якщо для бізнес-цілей, то має бути більш стриманим, з помірними кольорами, якщо ж ігровий додаток – навпаки – яскравим, щоб привертати увагу. Спочатку розробляється дизайн перших трьох головних сторінок, які є основою для наступних. При цьому потрібно опрацьовувати кожну деталь.

Наступний етап – програмна розробка. Щоб додаток функціонував повноцінно, поєднуються між собою усі елементи: кнопки, екрани, іконки.

Тестування є не менш важливим етапом розробки. Проводиться з метою усунення помилок, які формуються в певну таблицю. Кожен додаток є унікальним, тому передбачити і уникнути багів досить важко. Необхідно зробити усе для того, аби на кінцевому етапі споживач отримав ідеальний продукт.

Завершальним етапом є розміщення додатку. Існують спеціальні магазини для розміщення. Вибір потрібно здійснювати, враховуючи цільову аудиторію та техніку, якою вона користується. Існують різні типи додатків:

- контентні – створюються для поширення певної інформації;
- корпоративні – для вирішення різноманітних бізнес-цілей;
- ігрові – створюються з розважальною метою;
- сервісні – надання сервісних послуг (будильник та багато інших).

Також не потрібно забувати про те що потрібна підтримка додатку.

Тобто централізоване управління доступом до інформації. За допомогою таких широко відомих технологій, як обміну інформації по мережі, допомагає структурувати систему зберігання даних за рівнями доступу, в результаті розробляється система, де: кожен користувач має доступ тільки до певної інформації; на кожному ресурсі можливо збереженні тільки тих даних, які визначені політикою безпеки; є можливість протоколювання будь-яких подій доступу до інформаційних ресурсів; вся збережена інформація чітко впорядкована. Зручно всі дані зберігаються на сервері. Доступ до них для різних користувачів і груп користувачів можна обмежити, що зменшить ризик втрати критичних даних.

Для полегшення заповнення інформацією є розробка БД. Адже збір і аналіз інформації займає багато часу, сил і ресурсів. Автоматизована програма справляється з таким завданням швидше і легше.

Відповідно, за допомогою програми- можна знаходити інформацію швидше для наповнення і для пошуку та витягування інформації з них БД та відправки клієнтам які в свою чергу будуть зберігатися та оброблятися в додатку, щоб користувач зміг швидко їх витягувати та оброблювати.

1.4 Міжнародний стандарт управління проектами та архітетура ПЗ

Всесвітня федерація національних органів стандартизації (ISO) створила технічний комітет ISO/TC 176 «Управління якістю та забезпечення якості», і стандарт ISO 10006 є основним документом із серії стандартів профілю.

Принцип ефективності проектування оптимального процесу контролю процесу, а не контролю кінцевого результату, був головним приводом.

Цей набір стандартів поділяє процеси на дві категорії. До першої категорії віднесено процеси, пов’язані з забезпеченням продукту проекту (проектування, виробництво, перевірка). Останній опис відповідає стандарту ISO 9004--1. Стандарт ISO 10006 класифікує процеси управління проектом у другій категорії. Цей стандарт містить десять класифікацій процесів управління проектом.

Розробка стратегії – це процес, який починається в першій частині, яка зосереджується на тому, щоб проект відповідав вимогам замовника та визначає поточний курс роботи. Завданням другої групи є управління взаємозв'язками процесів. Інші вісім груп включають процеси, пов'язані з проектними завданнями, термінами, витратами, ресурсами, кадрами, інформаційними потоками, ризиками та закупівлями матеріалів.

Міжнародний стандарт ISO 10006 спрямований на проекти різних типів для різних умов навколишнього середовища, будь то малі чи великі, короткострокові чи довгострокові. Він не залежить від типу продукту, який планується створити; це може бути програмне забезпечення, технічні засоби, напівфабрикати, послуги або поєднання цих типів. Це означає, що певне керівництво потрібно адаптувати до специфічних умов розробки та реалізації кожного проекту відповідно до закладених у ньому рамкових вимог.

Терміни, такі як проект, продукт проекту, план проекту, учасник проекту, процес і оцінка ходу робіт, використовуються в таблиці за ISO 8402. Процеси та завдання управління якістю застосовуються до всіх процесів управління проектом, включаючи планування, організацію, моніторинг і контроль.

Національні стандарти управління проектами також розробляються на основі міжнародних стандартів. У 2001 році Асоціація з управління проектами Росії (SOVNET) розробила стандарт IPMA «Основи професійних знань». Національні стандарти компетентності експертів. В Україні переклад стандарту ISO 10006:2003 зареєстровано, і він є приватним стандартом, який часто використовується як основа корпоративних стандартів.

Керівні принципи щодо елементів системи якості, ідей і практичних процедур, реалізація яких важлива для якості та впливають на її досягнення в управлінні проектами, визначені стандартом 100006.

Оперуючи цими поняттями розберемо архітектури котрі використовуються при розробці даної системи.

Поняття сервер і клієнт і закріплені за ними ролі утворюють програмну концепцію «клієнт-сервер». Модель клієнт-сервер - це ще один підхід до

структурування ОС. У широкому сенсі модель клієнт-сервер передбачає наявність програмного компонента - споживача будь-якого сервісу - клієнта, і програмного компонента - постачальника цього сервісу - сервера. Взаємодія між клієнтом і сервером стандартизуется, так що сервер може обслуговувати клієнтів, реалізованих різними способами і, може бути, різними виробниками. При цьому головною вимогою є те, щоб вони запитували послуги сервера зрозумілим йому способом. Ініціатором обміну зазвичай є клієнт, який надсилає запит на обслуговування сервера, що знаходиться в стані очікування запиту. Один і той же програмний компонент може бути клієнтом по відношенню до одного виду послуг, і сервером для іншого виду послуг. Модель клієнт-сервер є скоріше зручним концептуальним засобом чіткого уявлення функцій того чи іншого програмного елемента в тій чи іншій ситуації, ніж технологією. Ця модель успішно застосовується не тільки при побудові ОС, але і на всіх рівнях програмного забезпечення, і має в деяких випадках більш вузький, специфічний сенс, зберігаючи, природно, при цьому всі свої загальні риси. Стосовно до структурування ОС ідея полягає в розбитті її на кілька процесів - серверів, кожен з яких виконує окремий набір сервісних функцій - наприклад, управління пам'яттю, створення або планування процесів. Кожен сервер виконується в режимі користувача. клієнт, яким може бути або інший компонент ОС, або прикладна програма, запитує сервіс, посилаючи повідомлення на сервер. Ядро ОС (зване тут мікроядром), працюючи в привілейованому режимі, доставляє повідомлення потрібного сервера, сервер виконує операцію, після чого ядро повертає результати клієнту за допомогою іншого повідомлення. Для взаємодії з клієнтом (або клієнтами, якщо підтримується одночасна робота з декількома клієнтами) сервер виділяє необхідні ресурси між процесами взаємодії (колективна пам'ять, пайп, сокет, і т. П.) І очікує запити на відкриття з'єднання (або, власне, запити на наданий сервіс). Залежно від типу такого ресурсу, сервер може обслуговувати процеси в межах однієї комп'ютерної системи або процеси на інших машинах через канали передачі даних (наприклад,

Формат запитів клієнта і відповідей сервера визначається протоколом. Специфікації відкритих протоколів описуються відкритими стандартами, наприклад, протоколи Інтернету визначаються в документах RFC.

Залежно від виконуваних завдань одні сервери, при відсутності запитів на обслуговування, можуть простоювати в очікуванні. Інші можуть виконувати якусь роботу (наприклад, роботу зі збору інформації), у таких серверів робота з клієнтами може бути другорядною завданням.

Класифікація стандартних серверів

Як правило, кожен сервер обслуговує один (або кілька схожих) протоколів і сервери можна класифікувати за типом послуг, які вони надають.

Універсальні сервери - особливий вид серверної програми, який не надає жодних послуг самостійно. Замість цього універсальні сервери надають серверів послуг спрощений інтерфейс до ресурсів між процесами взаємодії і / або уніфікований доступ клієнтів до різноманітних послуг. Існують кілька видів таких серверів:

Inetd від англ. internet super-server daemon демон сервісів IP - стандартна програма UNIX-систем - програма, що дозволяє писати сервери TCP / IP (і мережевих протоколів інших родин), що працюють з клієнтом через переслані inetd потоки стандартного вводу і виводу (stdin і stdout).

RPC - система інтеграції серверів у вигляді процедур доступних для виклику віддаленим користувачем через уніфікований інтерфейс. Інтерфейс винайдений Sun Microsystems для своєї операційної системи (SunOS, Solaris; Unix-система), в даний час іспользуетстся як в більшості Unix-систем, так і в Windows.

Прикладні клієнт-серверні технології Windows:

(D-) COM - дозволяє одним програмами виконувати операції над об'єктами даних використовуючи процедури інших програм. Спочатку дана технологія призначена для їх «впровадження і зв'язування об'єктів» (OLE англ. Object Linking and Embedding), але, в загальному, дозволяє писати широкий

спектр різних прикладних серверів. COM працює тільки в межах одного комп'ютера, DCOM доступна віддалено через RPC.

Active-X - Розширення COM і DCOM для створення мультимедіа-систем.

Універсальні сервери часто використовуються для написання всіляких інформаційних серверів, серверів, яким не потрібна якась специфічна робота з мережею, серверів, які не мають ніяких завдань, окрім обслуговування клієнтів. Наприклад, в ролі серверів для inetd можуть виступати звичайні консольні програми та скрипти.

Більшість внутрішніх і мережевих специфічних серверів Windows працюють через універсальні сервери (RPC, (D-) COM).

Мережеві служби забезпечують функціонування мережі, наприклад сервери DHCP і BOOTP забезпечують стартову ініціалізацію серверів і робочих станцій, DNS - трансляцію імен в адреси і навпаки.

Сервери тунелювання (наприклад, різні VPN-сервери) і проксі-сервери забезпечують зв'язок з мережею, недоступною роутингом.

Сервери AAA і Radius забезпечують в мережі єдину аутентифікацію, авторизацію і ведення логів доступу.

Інформаційні служби. До інформаційних служб можна віднести як найпростіші сервери що повідомляють інформацію про хості (time, daytime, motd), користувачів (finger, ident), так і сервери для моніторингу, наприклад SNMP. Більшість інформаційних служб працюють через універсальні сервери.

Особливим видом інформаційних служб є сервери синхронізації часу - NTP крім інформуванні клієнта про точний час NTP-сервер періодично опитує кілька інших серверів на предмет корекції власного часу. Крім корекції часу аналізується і коректується швидкість ходу внутрішнього годинника. Корекція часу здійснюється прискоренням або уповільненням ходу внутрішнього годинника (в залежності від напрямку корекції), щоб уникнути можливих проблем при звичайну перестановку часу.

Файл-сервери є сервери для забезпечення доступу до файлів на диску сервера.

Перш за все, це сервери передачі файлів на замовлення, по протоколах FTP, TFTP, SFTP і HTTP. Протокол HTTP орієнтований на передачу текстових файлів, але сервери можуть віддавати в якості запитаних файлів і довільні дані, наприклад, динамічно створені веб-сторінки, картинки, музику і т. П.

Інші сервери дозволяють монтувати дискові розділи сервера в дисковий простір клієнта і повноцінно працювати з файлами на них. Це дозволяють сервери протоколів NFS і SMB. Сервери NFS і SMB працюють через інтерфейс RPC.

Недоліки файл-серверної системи:

Дуже велике навантаження на мережу, підвищені вимоги до пропускної здатності. На практиці це робить практично неможливою одночасну роботу великої кількості користувачів з великими обсягами даних.

Обробка даних здійснюється на комп'ютері користувачів. Це тягне підвищені вимоги до апаратного забезпечення кожного користувача. Чим більше користувачів, тим більше грошей доведеться витратити на оснащення їх комп'ютерів.

Блокування даних при редагуванні одним користувачем робить неможливою роботу з цими даними інших користувачів.

Безпека. Для забезпечення можливості роботи з такою системою Вам буде необхідно дати кожному користувачеві повний доступ до цілого файлу, в якому його може цікавити тільки одне поле.

Сервери доступу до даних обслуговують базу даних і віддають дані на запит. Один з найпростіших серверів подібного типу - LDAP.

Для доступу до серверів баз даних єдиного протоколу не існує, проте всі сервери баз даних об'єднує використання єдиних правил формування запитів - мова SQL.

Служби обміну повідомленнями дозволяють користувачеві передавати і отримувати повідомлення (зазвичай - текстові).

В першу чергу це сервери електронної пошти, що працюють по протоколу SMTP. SMTP-сервер приймає повідомлення і доставляє його в локальний

поштову скриньку користувача або на інший SMTP-сервер (сервер призначення або проміжний). На багатокористувацьких комп'ютерах, користувачі працюють з поштою прямо на терміналі (або веб-інтерфейсі). Для роботи з поштою на персональному комп'ютері, пошта забирається з поштової скриньки через сервери, що працюють по протоколах POP3 або IMAP.

Для організації конференцій існує сервери новин, що працюють по протоколу NNTP.

Для обміну повідомленнями в реальному часі існують сервери чатів, стандартний чат-сервер працює по протоколу IRC - розподілений чат для інтернету. Існує велика кількість інших чат-протоколів, наприклад ICQ або Jabber.

Сервери віддаленого доступу

Сервери віддаленого доступу, через відповідну клієнтську програму, забезпечують користувача консольним доступом до віддаленої системи.

Для забезпечення доступу до командного рядка служать сервери telnet, RSH, SSH.

Графічний інтерфейс для Unix-систем - X Window System, має вбудований сервер віддаленого доступу, так як з такою можливістю розроблявся спочатку. Іноді можливість віддаленого доступу до інтерфейсу X-Window неправильно називають «X-Server» (цим терміном в X-Window називається відеодрайвер).

Стандартний сервер віддаленого доступу до графічного інтерфейсу Microsoft Windows називається термінальний сервер.

Деяку різновид управління (точніше моніторингу і конфігурації), також, надає протокол SNMP. Комп'ютер або апаратний пристрій для цього повинно мати SNMP-сервер.

Ігрові сервери, служать для одночасної гри декількох користувачів в єдиній ігрової ситуації. Деякі ігри мають сервер в основний постачання і дозволяють запускати його у невиділеному режимі (тобто дозволяють грати на машині, на якій запущений сервер).

Серверні рішення - операційні системи та / або пакети програм, оптимізовані під виконання комп'ютером функцій сервера і / або містять в своєму складі комплект програм для реалізації типового сервісів.

Прикладом серверних рішень можна привести Unix-системи, спочатку призначені для реалізації серверної інфраструктури, або серверні модифікації платформи Microsoft Windows.

Також необхідно виділити пакети серверів і супутніх програм (наприклад, комплект веб-сервер JavaScript MySQL для швидкої розгортки хостингу) для установки під Windows (для Unix властива модульна або «пакетна» установка кожного компонента, тому такі рішення рідкісні).

В інтегрованих серверних рішеннях установка всіх компонентів виконується одноразово, всі компоненти в тій чи іншій мірі тісно інтегровані і попередньо налаштовані один на одного. Однак в цьому випадку, заміна одного з серверів або вторинних систем (якщо їх можливості не задовольняють потреб) може представляти проблему.

Серверні рішення служать для спрощення організації базової IT-інфраструктури компаній, тобто для оперативної побудови повноцінної мережі в компанії, в тому числі і «з нуля». Компонування окремих серверних систем в рішення має на увазі, що рішення призначене для виконання більшості типових завдань; при цьому значно знижується складність розгортання і загальна вартість володіння IT-інфраструктурою, побудованою на таких рішеннях.

Проксі-сервер служба в комп'ютерних мережах, що дозволяє клієнтам виконувати непрямі запити до інших мережних служб. Спочатку клієнт підключається до проксі-сервера і запитує який-небудь ресурс (наприклад, e-mail), розташований на іншому сервері. Потім проксі-сервер або підключається до вказаного серверу і отримує ресурс у нього, або повертає ресурс із власного кеша (у випадках, якщо проксі має свій кеш). У деяких випадках запит клієнта або відповідь сервера може бути змінений проксі-сервером в певних цілях. Також проксі-сервер дозволяє захищати клієнтський комп'ютер від деяких мережевих атак.

1.5 Висновки до розділу 1

В даному розділі розглянуто та проаналізовано ринок систем можливих конкурентів, як було вищесказано.

Також сформовано вимоги до програмного продукту, розглянуто можливі функції для зручності користувачів.

Розглянуто можливі архітектури програмного забезпечення. Освоєно основні цілі та завдання сучасних програмних продуктів. Перелічино технології розробки, обрано можливу структуру проекту, переваги та недоліки даної структури.

Також аналізовано технології розробки та фреймворки для подальшої розробки інформаційної системи.

2 АЛГОРИТМІЧНІ ТА МАТЕМАТИЧНІ МОДЕЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТА ОБІГУ ПРОДУКЦІЇ

2.1 Ергономіка системи, виділення стадій розробки

Програмування - це акт інструктування комп'ютерів для виконання завдань». Ще його називають розробкою або кодінгом .

ПЗ являє собою послідовність інструкцій, які виконуються ПК. Комп'ютер же - це будь-який пристрій, здатне обробляти код.

Під технологією розробки ПЗ розуміють оптимальний спосіб ведення розробки, який за певних умов забезпечить отримання кінцевого продукту з наперед заданими властивостями.

Ергономіка програмного забезпечення (ПЗ) — це галузь, яка займається розробкою та оптимізацією програмного забезпечення, щоб зробити програму зручною, ефективною та задоволеною для користувача. Ергономіка програмного забезпечення полягає в тому, щоб програмне забезпечення було максимально простим у використанні, зручним у використанні та відповідало потребам і очікуванням користувачів. Це включає розгляд фізичних, когнітивних і емоційних елементів взаємодії людини з комп’ютером, а також аналіз, проектування та тестування користувацьких інтерфейсів.

Ергономіка програмного забезпечення включає такі елементи, як простота використання (usability):

Програма повинна бути ефективною для досвідчених користувачів і легкою для нових користувачів.

Інтерфейс повинен бути простим для розуміння, а всі функції повинні бути доступними.

Ефективність і продуктивність: Програмне забезпечення повинно дозволяти користувачам виконувати свої завдання швидко та ефективно.

Це включає скорочення кількості дій, необхідних для досягнення результату.

Задоволення клієнта:

Користувачі повинні відчувати задоволення від того, як вони використовують програму, що може включати приємний вигляд, відсутність стресу та впевненість у своїх діях.

Доступність: Програмне забезпечення має бути доступним для широкого кола користувачів, включаючи людей з фізичними або когнітивними обмеженнями.

Підтримка спеціальних пристроїв введення, можливість збільшення шрифту, екранні читачі тощо може бути частиною цього.

Гнучкість і адаптивність: Програмне забезпечення повинно дозволяти користувачам змінювати інтерфейс залежно від їхніх уподобань.

Це може включати зміни дизайну інтерфейсу, налаштувань функціональності та поведінки системи.

Зменшення помилок: Системи повинні бути розроблені таким чином, щоб вони були менш схильні до помилок користувача.

Це включає використання захисних систем, зворотного зв’язку та явних повідомлень про помилки.

Приклади застосування ергономіки ПЗ: мобільні додатки: ергономіка є життєво важливою для створення зручних інтерфейсів, де кожен елемент екрана має бути добре помітним і простим у використанні, оскільки екран смартфонів обмежений.

Сайти та веб-додатки: веб-сторінки повинні бути максимально зручними та простими у використанні, коли справа доходить до організації контенту та взаємодії з елементами.

Професійні інструменти: Програмне забезпечення, яке використовується для професійного використання, як-от програмні засоби для розробників або CAD-системи, повинно бути максимально ефективним і адаптивним, щоб користувачі могли максимізувати продуктивність.

В сучасному світі ергономіка програмного забезпечення є важливою, оскільки якість взаємодії з програмними продуктами може суттєво впливати на успіх продукту та конкурентоспроможність.

Всі етапи розробки програмного забезпечення повинні включати ергономіку програмного забезпечення, але найбільше її враховують на початкових стадіях. Це дозволяє встановити зручність використання та ефективну взаємодію з програмою ще на етапі проектування, що значно підвищує ймовірність того, що продукт вийде в результаті. Розглянемо, як ергономіка програмного забезпечення впливає на різні стадії розробки: 1. На цьому етапі необхідно визначити потреби та бажання користувачів щодо ергономіки.

Дослідження користувачів за допомогою інтерв'ю, опитувань і спостереження

аналізуйте конкурентів і поточні рішення, щоб дізнатися про кращі практики та виявити недоліки.

створити персону користувача, або персону користувача, для покращення розуміння цільової аудиторії

Завдання: переконатися, що функціональні можливості програмного забезпечення задовольняють потреби користувачів.

Визначте важливі завдання, які користувачі хочуть вирішити за допомогою програмного забезпечення.

2: На цьому етапі планування та проектування ергономіки є розробка ідеї для користувацького інтерфейсу (UI).

створювати концептуальні візуальні каркаси та прототипи інтерфейсів.

Тестування прототипів з користувачами для оцінки їх зручності використання.

визначити та дотримуватися стандартів дизайну, таких як консистентність, зворотний зв'язок, запобігання помилкам тощо.

Завдання: зробити використання ПЗ простим і простим для користувачів.

Зменшіть кількість кроків, необхідних для виконання основних завдань користувачів.

3: На цьому етапі розробки ергономіка інтегрується в процес розробки.

тісна співпраця розробників і дизайнерів UI/UX для забезпечення того, що технічна реалізація відповідає проектним рішенням.

використання сучасних технологій для забезпечення зручності та ефективності інтерфейсу (наприклад, адаптивний дизайн, швидке реагування системи).

Завдання: Переконатися, що реалізація відповідає проектним вимогам і стандартам ергономіки.

Враховуйте результати тестування та внесіть необхідні корективи.

:4. Наразі проводяться тести ергономіки з використанням реальних користувачів для визначення ефективності та зручності інтерфейсу.

використання A/B-тестування, тестування юзабіліті та дослідження зворотного зв’язку від користувачів.

Аналіз даних, зібраних для пошуку проблем з ергономікою та вирішення їх.

Завдання: знайти та виправити проблеми, які можуть погіршити користувацький досвід.

Переконайтеся, що програмне забезпечення відповідає очікуванням користувачів щодо ефективності та зручності.

5. Впровадження та підтримка ергономіки на цьому етапі включає збір відгуків користувачів після випуску продукту для виявлення проблем або потенційних покращення.

регулярне оновлення інтерфейсу відповідно до змін у поведінці користувачів або появи нових технологічних можливостей.

підтримувати та навчати користувачів, щоб вони могли легко використовувати ПЗ.

Підтримка та покращення ергономіки програмного забезпечення протягом життєвого циклу продукту

Залишатися гнучким до змін, викликаних користувачами та технологіями.

Всі етапи розробки програмного забезпечення включають ергономіку. Вимоги формуються на початку, а потім проходять всі наступні етапи, такі як

планування, проектування, розробка, тестування та підтримка. Такий підхід дозволяє створювати товари, які не тільки відповідають технічним вимогам, але й забезпечують користувачам приємний і продуктивний досвід.

В ході дослідження був структурований існуючий матеріал і на його основі створена технологія розробки клієнт-серверних систем.

Розробка здійснюється за наступним алгоритмом:

1. Вибір архітектури клієнт-сервера (технічна частина). На початку розробки необхідно вибрати між дворівневою і багаторівневої архітектурою. Вибір в даному випадку залежить від кількості користувачів, які будуть працювати в даній системі, вартості обладнання та подальше обслуговування системи (наприклад, доробка функціоналу).

Детальний опис цих архітектур описані в першому розділі.

2.Вибор мови програмування (програмна частина). Клієнт-сервер можна написати на самих різних мовах. У різних мов присутні свої особливості в реалізації мережевих завдань. Нижче наведено (рисунок 2.1) список десяти мов програмування за індексом TIOBE станом на січень 2020 року ("TIOBE programming community index"- індекс, що оцінює популярність мов програмування, на основі підрахунку результатів пошукових запитів [12]. Для формування індексу використовується пошук в декількох найбільш відвідуваних (за даними Alexa) порталах: Google, Blogger, Wikipedia, YouTube, Baidu, Yahoo !, Bing, Amazon.

Jan 2020	Jan 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.896%	-0.01%
2	2		C	15.773%	+2.44%
3	3		Python	9.704%	+1.41%
4	4		C++	5.574%	-2.58%
5	7	^	C#	5.349%	+2.07%
6	5	v	Visual Basic .NET	5.287%	-1.17%
7	6	v	JavaScript	2.451%	-0.85%
8	8		PHP	2.405%	-0.28%
9	15	^^	Swift	1.795%	+0.61%
10	9	v	SQL	1.504%	-0.77%

Рисунок 2.1 – Рейтинг мов програмування

Варто враховувати, що Веб-програмування - це окремий випадок програмування клієнт-серверного додатка. Клієнтом в веб додатків виступає браузер, а сервером - веб-сервер що дає ряд плюсів:

– Низька вартість впровадження;

– Дуже проста підтримка;

– Незалежність від Операційної системи;

– Доступність з будь-якої точки світу.

У веб-програмуванні зазвичай використовують JavaScript на стороні клієнта, а на стороні сервера можуть бути як один або кілька мов програмування: PHP, Java, C ++, C, JavaScript, ASP.NET, Python і інші.

3. Створення робочого прототипу.

Процес створення прототипу зазвичай складається з кроків:

1. Визначення початкових вимог;

2. Розробка першого варіанту прототипу, який містить тільки призначений для користувача інтерфейс системи;

3. Вивчення прототипу замовником і кінцевими користувачами, отримання зворотного зв'язку про необхідні зміни та доповнення;

4. Переробка та поліпшення прототипу: з урахуванням отриманих зауважень і пропозицій змінюються як специфікації, так і прототип, після цього кроки 3 і 4 можуть повторюватися.

Прототипи дають можливість глибше вникнути в проблему і вжити всіх необхідні проектні рішення ще на ранніх етапах проектування.

4. Тестування. Для клієнт-серверних систем тестування можна

умовно розділити на два рівні:

– Серверна;

– Клієнтська.

На першому (серверному) рівні, тестується взаємодія

випускається програмного забезпечення з оточенням, в яке воно буде

встановлено:

1. Апаратні засоби (тип і кількість процесорів, обсяг пам'яті, характеристики мережі / мережевих адаптерів і т.д.);

2. Програмні засоби (ОС, драйвера і бібліотеки, стороннє ПЗ, впливає на роботу програми і т.д.);

3. Основний упор тут робиться на тестування з метою визначення оптимальної конфігурації обладнання, що задовольняє необхідним характеристикам якості (ефективність, портативність, зручність супроводу, надійність).

На клієнтському рівні, програмне забезпечення тестується з позиції його кінцевого користувача і конфігурації його робочої станції. На цьому етапі будуть протестовані наступні характеристики: зручність використання, функціональність. Для цього необхідно буде провести ряд тестів з різними конфігураціями робочих станцій:

1. Тип, версія і бітность операційної системи (подібний вид тестування називається кроссплатформне тестування);

(Подібний вид тестування називається крос-платформерне тестування);

3. Тип і модель відео адаптера (при тестуванні ігор це дуже важливо);

4. Робота програми при різних дозволах екрану;

5. Версії драйверів, бібліотек і т.д. (Для JAVA систем версія JAVA машини дуже важлива, теж можна сказати і для .NET систем щодо версії .NET бібліотеки);

Вже на початковому етапі стає очевидно, що чим більше вимог до роботи програми при різних конфігураціях робочих станцій, тим більше тестів нам необхідно буде провести. У зв'язку з цим, рекомендуємо, по можливості, автоматизувати цей процес, так як саме при конфігураційному тестуванні автоматизація реально допомагає заощадити час і ресурси. Звичайно ж автоматизоване тестування не є панацеєю, але в даному випадку воно виявиться дуже ефективним помічником.

5. Реліз і подальша підтримка. Стабільна версія програми, готова до використання за її призначенням, що пройшла тестування, в яких виправлені основні помилки, але існує ймовірність появи нових, раніше не помічених, помилок.

Для створення використовувалося Visual Studio 2012 (VS2012) та мова програмування C++ є потужними інструментами для розробки програмного забезпечення для платформи Windows. Ось деякі переваги використання цих інструментів:

1. Інтегроване середовище розробки (IDE): Visual Studio 2012 має потужне та зручне середовище розробки з багатьма корисними функціями, такими як автодоповнення коду, відлагодження, рефакторинг, керування версіями та багато іншого. Це полегшує розробку та підвищує продуктивність розробника.

2. C++ є потужною мовою програмування, яка надає велику швидкість виконання, ефективне використання пам'яті та можливість працювати на рівні машинного коду, що робить її ідеальним вибором для розробки швидких та ефективних додатків.

3. Підтримка платформи Windows: VS2012 та мова C++ надають широкі можливості для розробки додатків для платформи Windows. Це включає створення класичних десктопних програм, універсальних програм для Windows, додатків для Windows Store, служб та інших типів додатків.

4. Бібліотеки та інструменти: Visual Studio постачається з багатьма корисними бібліотеками та інструментами, які спрощують розробку. Наприклад, Microsoft Foundation Classes (MFC) та Windows Template Library (WTL) допомагають у створенні десктопних додатків, а Windows Runtime (WinRT) API надає доступ до функціональності платформи Windows для створення універсальних додатків.

5. Підтримка стандартів: C++ підтримує стандарти мови, такі як C++98, C++11, C++14, C++17 та інші. Це дозволяє розробникам використовувати сучасні функції та можливості мови для покращення продуктивності та зручності використання.

Узагальнюючи, використання Visual Studio 2012 та мови C++ для розробки додатків на платформі Windows дозволяє розробникам швидко створювати широкий спектр програмного забезпечення з високою продуктивністю та ефективністю.

Структурна модель будувалася так що навіть після релізу можна відносно легко додавати новий функціонал.

2.2 Алгоритмічне та матиматечне рішення планування БД

Створення бази даних у Visual Studio 2012 за допомогою C++ можна виконати, наприклад, за допомогою вбудованої підтримки баз даних SQL Server LocalDB або SQLite. Нижче наведено загальний опис кожного з цих методів:

Використання SQL Server LocalDB:

- Відкрийте Visual Studio і створіть новий проект C++.
- У рішенні проекту правий клік і оберіть "Додати" -> "Новий елемент".
- У вікні "Новий елемент" виберіть "База даних" і оберіть "База даних SQL Server".
- Далі слідуйте майстру створення бази даних для налаштування з'єднання з SQL Server LocalDB, створення таблиць та визначення їх структури.

Після створення бази даних ви зможете додавати таблиці, визначати їх структуру та взаємозв'язки, а також виконувати SQL-запити для маніпулювання даними.

Ці методи можуть відрізнятися залежно від версії Visual Studio або від того, який плагін або бібліотеку ви використовуєте для роботи з базами даних.

База даних:

Таблиця "Продукти":

Поля:

id_product: унікальний ідентифікатор продукту (може бути автоматично згенерованим).

name_product: назва продукту.

cost_product: вартість одиниці продукту.

numb_product: кількість одиниць продукту на складі.

unit_product: одиниця виміру продукту (кг, шт, літр і т.д.).

Таблиця "Страви":

Поля:

id_dish: унікальний ідентифікатор страви.

name_dish: назва страви.

dish_price: вартість страви.

category_dish: категорія страви (наприклад, супи, салати, основні страви тощо).

Таблиця "Замовлення":

Поля:

id_order: унікальний ідентифікатор замовлення.

order_date: дата замовлення.

table_number: номер столика.

waiter_name: ім'я офіціанта.

Проміжні таблиці:

dish_product: зв'язок багато до багатьох між стравами і продуктами. Містить ідентифікатор страви та ідентифікатор продукту, які входять до складу цієї страви.

order_dish: зв'язок багато до багатьох між замовленнями і стравами. Містить ідентифікатор замовлення та ідентифікатор страви, які були замовлені в цьому замовленні.

Така структура бази даних дозволить зберігати і управляти інформацією про продукти, страви та замовлення, а також встановлювати зв'язки між ними для зручного управління і відображення даних.

2.3 Алгоритмічне та матиматичне рішення провайдера з БД

Використовуючи інтерфейс OLE DB, PowerBuilder може з’єднувати, зберігати та отримувати дані в базах даних ANSI/DBCS та Unicode, але не перетворює дані між Unicode та ANSI/DBCS. Коли символьні дані або текст команди надсилаються до бази даних, PowerBuilder надсилає рядок Unicode. Постачальник даних повинен гарантувати, що дані правильно збережено як дані Unicode. Коли PowerBuilder отримує символьні дані, він припускає, що ці дані є Unicode.

База даних Unicode — це база даних, набір символів якої встановлено у форматі Unicode, наприклад UTF-8, UTF-16, UCS-2 або UCS-4. Усі дані мають бути у форматі Unicode, і будь-які дані, збережені в базі даних, мають бути перетворені в дані Unicode неявно чи явно.

База даних, яка використовує ANSI (або DBCS) як набір символів, може використовувати спеціальні типи даних для зберігання даних Unicode. Стовпці з цими типами даних можуть зберігати лише дані Unicode. Будь-які дані, збережені в такому стовпці, мають бути явно перетворені в Юнікод. Це перетворення має виконуватися сервером бази даних або клієнтом.

Компоненти підключення OLE DB

Коли ви отримуєте доступ до постачальника даних OLE DB у PowerBuilder, ваше підключення проходить кілька рівнів, перш ніж досягти постачальника даних. Важливо розуміти, що кожен рівень представляє окремий компонент з'єднання, і кожен компонент може надходити від іншого постачальника.

Оскільки OLE DB є стандартним API, PowerBuilder використовує той самий інтерфейс для доступу до кожного постачальника даних OLE DB. Поки постачальник даних OLE DB підтримує інтерфейси об'єктів, необхідні для PowerBuilder, PowerBuilder може отримати до нього доступ через інтерфейс OLE DB.

Діаграма представлена загальні компоненти підключення OLE DB на рисунку 2.2.

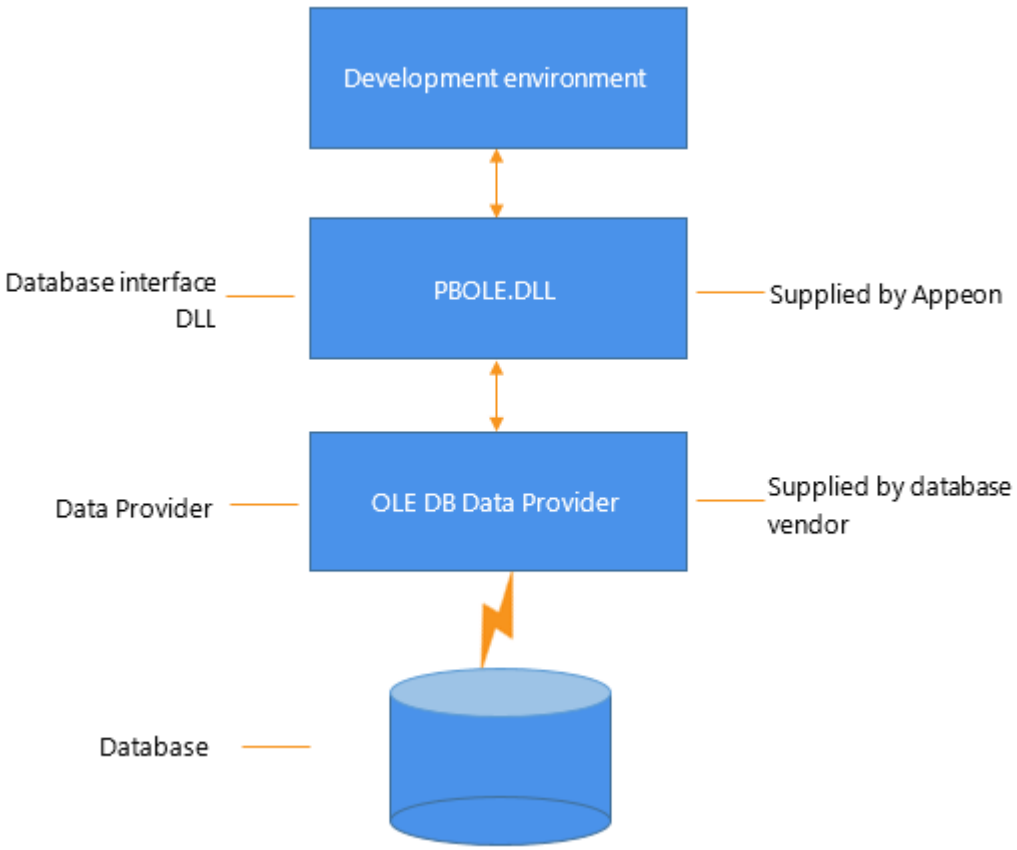


Рисунок 2.2 – Діаграма функціонування клієнта та сервера

З програмної позиції дану архітектуру реалізовують:

БД для зберігання;

Провайдер обробки;

Додаток.

Якщо правильно розставити приорітети можна розглянути концептуальну модель на рисунку 2.3.

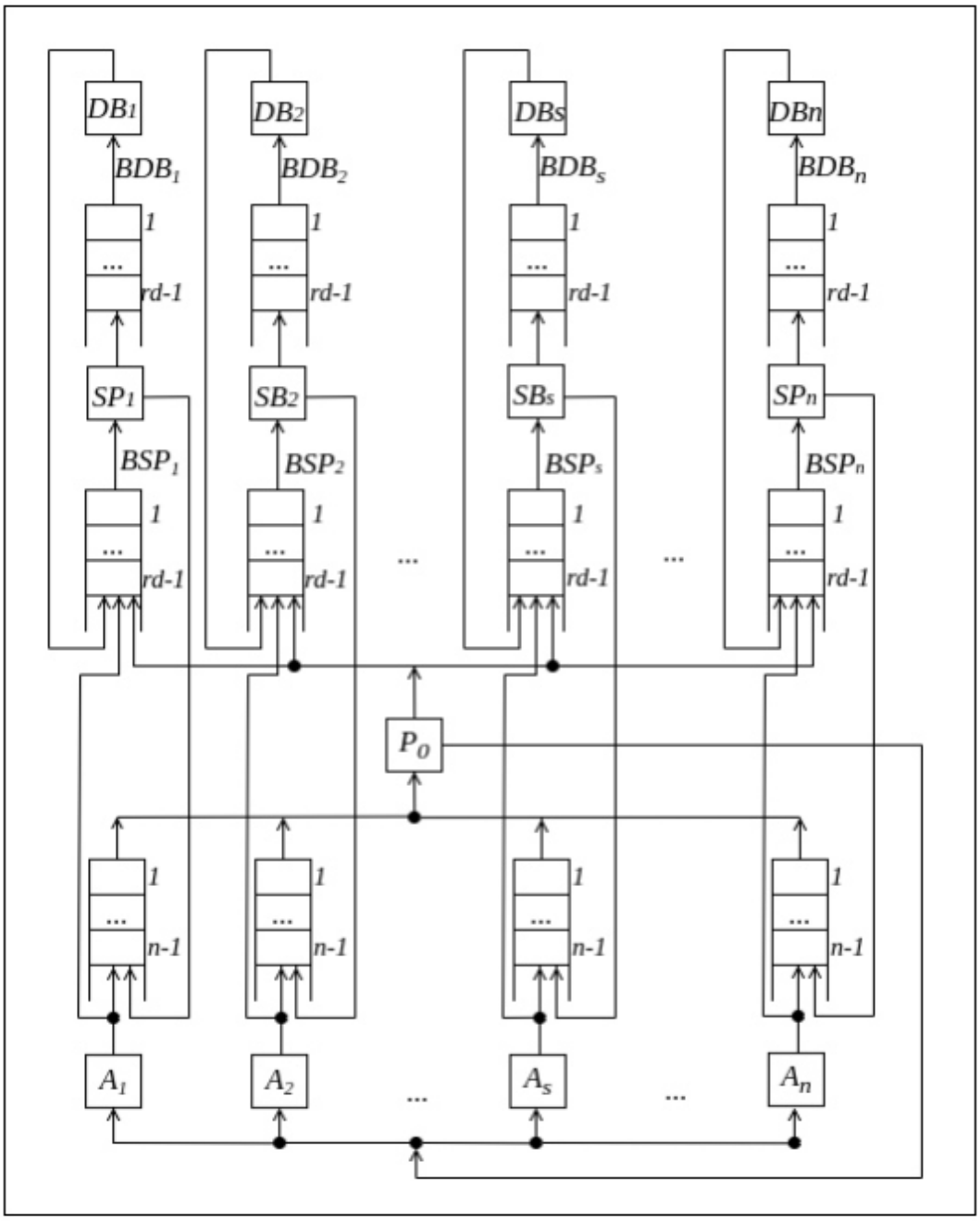


Рисунок 2.3 – Концептуальна схема обміну даних системи

Отже можна розглянути наступним чином як концептуальну модель:

$DB_1, \dots, DB_s, \dots, DB_n,$ - потоки сервра з база даних, які моделюють багато поточність;

$BDB_1, \dots, BDB_s, \dots, BDB_n,$ - буфери пам'яті серверу, які відповідають за дані;

$SP_1, \dots, SP_s, \dots, SP_n,$ - так званні тонкі клієнти, які моделюють роботу клієнт систем;

$SP_1, \dots, SP_s, \dots, SP_n,$ - буфери пам'яті клієнтів, які відповідають за отримання даних від клієнтів.

Якщо дану архітектуру розглядати як математичну модель то отримаємо потоки даних від клієнтів, які потрібно буде обробляти. Для ідентифікації статусу мережі введено векторний простір , який виглядатиме наступним чином (ф. 2.1):

$$\bar{E} = \left\{ E_i \left(\begin{array}{c} i_{11}, \dots, i_{1s}, \dots, i_{1n}; i_{21}, \dots, i_{2s}, \dots, i_{2n}; \dots \\ \dots; i_{s+1,1}, \dots, i_{s+1,n}, \dots, i_{n+1,1}; \dots \\ \dots; i_{n+1,s}, \dots, i_{n+1,n}, \dots, i_{n+2,1}, \dots \\ \dots, i_{n+2,s}, \dots, i_{n+2,s}, \dots, i_{n+2,n}; \dots \\ \dots; i_{n+s+1,1}, \dots, i_{n+s+1,s}, \dots, i_{n+s+1,n}; \dots \\ \dots; i_{2n+1,1}, \dots, i_{2n+1,s}, \dots, i_{2n+1,n}; \dots \\ \dots; i_{2n+2,1}, \dots, i_{2n+s+1,s}, \dots, i_{2n+2,n}; \dots \\ \dots; i_{2n+s+1,1}, \dots, i_{2n+s+1,s}, \dots, i_{2n+s+1,n}; \dots \\ \dots; i_{3n+1,1}, \dots, i_{3n+1,s}, \dots, i_{3n+1,n} \end{array} \right) \right\}, i = \overline{1,s} \qquad 2.1$$

Де:

$$i_s = \begin{cases} 1, \text{якщо } s - \text{й користувач (тонкий клієнт}, s = \overline{1,n}) \\ \text{знаходиться в активному статусі} \\ \text{тобто формує запит на ініціалізацію даних системи}), \\ 0 - \text{в протилежному випадку;} \end{cases}$$

($i_{sr}, s = \overline{2, n+1}, r = \overline{1,n}$) – описує чергу до каналу і статус каналу, де i_{sr} – кілікість запитів r-го користувача (тонкого клієнту) в s-й буфера пам'яті каналу і наобслуговування в каналі;

($i_{sr}, s = \overline{n+2, 2n+1}, r = \overline{1,n}$) – описує чергу до серверів систем і статуси серверів систем, де i_{sr} – кількість запитів r-го користувача в буферній пам'яті s-й сервера системи і на обслуговування в s-им сервером системи;

($i_{sr}, s = \overline{2n+2, 3n+1}, s = \overline{1, n}$) – описує черги до серверів з базами даних та статус серверів баз даних, де i_{sr} – кількість запитів r-го користувача в буферній пам'яті s-й сервера системи і на обслуговування в s-им сервером системи;

При цьому є деякі обмеження.

Представлення інтересів характеристики відображають ймовірності статусу мереж. Нехай стаціонарна ймовірність того що вже знаходиться в статусі де процес потоку змінюється статусом мережі та описується поток представленим вище.

2.4 Алгоритмічне та матиматичне рішення системи

Перш ніж почати написання системи потрібно зрозуміти цілі, покладені на проект. Які потреби користувачів він повинен задовольняти? Яка його цінність для цільової аудиторії? Відповівши на ці питання, можна чітко зрозуміти, який саме продукт потрібний. Саме тому створення системи має включати в себе попередній аналіз ринку клієнта, активності конкурентів та переваг цільової аудиторії.

Створення системи відрізняється від розробки сайту, але і тут важливо дотримуватись законів розробки. Адже клієнти будуть їх використовувати з певними цілями, а це означає, що важливо зробити додаток інтуїтивно зрозумілим, його структуру – простою і оформлення – приємним для очей. Щоб хотілось використовувати його щодня. Розроблено прототип ключових розділів системи, після чого створено макети. Розробка систем, в тому числі і етап дизайну, – складний процес, який вимагає специфічних знань і глибокого розуміння особливостей розробки та принципів дизайну. Дизайн системи готовий тепер потрібно зробити так, щоб усе, що намальовано і придумано, працювало. Тут у справу вступають впровадження технологій, що є одним з найважливіших етапів, які включає розробка систем.

Аналізувавши багато систем було вирішено додати декілька активностей, які дозволяють переходити між екранами і взаємодіяти між собою.

Виділено такі активності:

– Форма відображення даних;

– Форма оформлення даних;

– Форма обрахування вартості страв.

Розглянемо детальніше кожну форму та її взаємодію з користувачем.

Алгоритмічне рішення для створення форм для виведення даних у вигляді таблиці, оформлення замовлення та калькулятора страв може включати наступні кроки:

Створення проекту у Visual Studio:

Створення нового проекту у VS2012 з використанням мови програмування C++.

Вибір типу проекту, наприклад, Windows Forms Application.

Створення користувацького інтерфейсу:

Додавання відповідних елементів у інтерфейс, таких як DataGridView для виведення даних у вигляді таблиці, кнопки для додавання, видалення та редагування записів, текстові поля для введення даних.

Розміщення елементів у вікні форми, налаштування їх розміщення та вигляду.

Підключення до бази даних:

Встановлення зв'язку з базою даних у VS2012.

Виконання запитів до бази даних для отримання, додавання, видалення та оновлення даних.

Реалізація функціоналу:

Написання коду для обробки подій, таких як натискання кнопок або вибір елементів у таблиці.

Реалізація логіки для додавання, видалення та редагування записів у базі даних.

Реалізація функціоналу сортування, фільтрації та пагінації даних.

Створення форми оформлення замовлення:

Додавання елементів у інтерфейс для вибору страв, обрання столика та офіціанта, введення кількості страв.

Написання коду для обчислення загальної вартості замовлення та оновлення залишків продуктів на складі.

Реалізація можливості додавання знижок або акцій до замовлення.

Створення калькулятора для створення страв:

Розробка інтерфейсу для вибору продуктів та введення їх кількості.

Написання коду для обчислення вартості страви на основі вартості продуктів та їх кількості.

Реалізація можливості зберігання створених рецептів та використання їх для оформлення замовлень.

Навігація по розділах системи коли користувач відкриває програму, йому повинно бути легко і швидко знайти необхідний контент. Це головний принцип, на базі якого повинна будуватися вся навігація по розділах. Взаємодію між розділами зображено на рисунку 2.4.

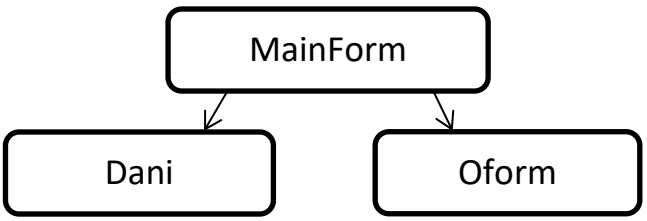


Рисунок 2.4 – Ілюстрація того, як активності утворюють відгалуження у макеті та містять активності

Поставлена математична модель розглянута наступним чином у вигляді векторів даних, які формуються:

$A = (A_1, \dots, A_2, \dots, A_n)$, – декілька клієнтських записів (запити до БД);

$SP = (SP_1, \dots, SP_s, \dots, SP_n)$, – декілька каналів з'єднання з БД;

$DB = (DB_1, \dots, DB_s, \dots, DB_n)$, – кількість потоків з'єднання з базою даних;

$Q = (Q_1, \dots, Q_2, \dots, Q_n)$, – кількість запитів до бази даних які формуються клієнтом;

$r = (r_1, \dots, r_2, \dots, r_q)$, – можлива кількість запитів, які звернуться до БД;

$V = (V_1, \dots, V_j, \dots, V_d)$, – об'єм відношень даних між клієнтом та БД;

$VSP = (VSP_1, \dots, VSP_s, \dots, VSP_n)$, – швидкість зчитування даних клієнтом;

$DSP = (DSP_1, \dots, DSP_s, \dots, DSP_n)$, – швидкість запису оброблюваних даних клієнта;

Взаємовдношення представлено на вищесказаній схемі на рисунку 2.3.

2.5 Висновки до розділу 2

В даному розділі представлено математичні моделі та деяке алгоритмічне рішення щодо полегшення розробки інформаційної системи обліку та обігу продукції в готельно ресторанному бізнесі.

Проаналізовано технології обрано відповідні архітектури для системи. Також виділено стадії розробки окремих модулів, а саме:

– Модуль відображення даних або ж графічний інтерфейс;

– Модуль зберігання даних або ж БД;

– Модуль взаємодії БД та інформаційної системи;

– Модуль взаємодії інформаційної система та графічного інтерфейсу користувача.

Модулі та зв'язки між ними допоможуть реалізувати потрібний функціонал у додатку за допомогою мови програмування C++ та інструментів Visual Studio.

3 АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ РІШЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТА ОБІГУ ПРОДУКЦІЇ

3.1 Розробка та функціонування БД

CREATE DATABASE створює базу даних із заданим іменем. Щоб використовувати цей оператор, вам потрібні CREATEпривілеї для бази даних. CREATE SCHEMAє синонімом CREATE DATABASE. Помилка виникає, якщо база даних існує, а ви не вказали IF NOT EXISTS. CREATE DATABASEне дозволяється в сеансі, який має активний LOCK TABLESоператор. Кожен create_optionвизначає характеристику бази даних. Характеристики бази даних зберігаються в словнику даних.

Параметр CHARACTER SETвизначає стандартний набір символів бази даних. Параметр COLLATE визначає сортування бази даних за замовчуванням.. Параметр ENCRYPTION, представлений у MySQL 8.0.16, визначає стандартне шифрування бази даних, яке успадковується таблицями, створеними в базі даних. Дозволені значення: 'Y'(шифрування ввімкнено) і 'N'(шифрування вимкнено).

Якщо ENCRYPTION параметр не вказано, значення системної default_table_encryption змінної визначає стандартне шифрування бази даних. Якщо table_encryption_privilege_check системну змінну ввімкнено, TABLE_ENCRYPTION_ADMIN потрібен привілей, щоб вказати налаштування шифрування за замовчуванням, яке відрізняється від default_table_encryption налаштування

База даних у MySQL реалізована як каталог, що містить файли, які відповідають таблицям у базі даних. Оскільки під час початкового створення бази даних у базі даних немає таблиць, CREATE DATABASE оператор створює лише каталог у каталозі даних MySQL.

Коли ви створюєте базу даних, дозвольте серверу керувати каталогом і файлами в ньому. Безпосереднє маніпулювання каталогами та файлами бази даних може спричинити невідповідності та несподівані результати. MySQL не

має обмежень щодо кількості баз даних. Базова файлова система може мати обмеження на кількість каталогів.

Один до багатьох (One-to-Many):

Зв'язок між таблицею order і таблицями placing_order, dish, products є відношенням один до багатьох. Одне замовлення може містити багато страв і багато продуктів, але кожна страва і кожен продукт можуть бути включені тільки в одне замовлення. Так само, кожна страва може бути включена тільки в одне замовлення.

Багато до багатьох (Many-to-Many):

Зв'язок між таблицями dish і products є прикладом відношення багато до багатьох. Одна страва може містити декілька продуктів, і навпаки, один продукт може бути використаний для приготування декількох страв.

Зв'язки між таблицями в базі даних визначають спосіб, якими дані взаємодіють та як вони пов'язані між собою. Ось детальний опис зв'язків у вашій базі даних:

Продукти (products) та Склад страв (dish_composition):

Таблиця dish_composition встановлює зв'язок між стравами і продуктами, які використовуються для їх приготування. Кожен запис у таблиці dish_composition містить ідентифікатор страви (comp_id_dish) та ідентифікатор продукту (comp_id_products). Ці поля утворюють зв'язок багато-до-багатьох між таблицями dish і products.

Страви (dish) та Замовлення (order):

Таблиця placing_order встановлює зв'язок між стравами і замовленнями. Кожен запис у таблиці placing_order містить ідентифікатор страви (pl_id_dish) та ідентифікатор замовлення (pl_id_order). Ці поля утворюють зв'язок багато-до-багатьох між таблицями dish і order.

Замовлення (order) та Продукти (products):

Немає прямого зв'язку між цими двома таблицями в базі даних, але обидві таблиці використовуються у контексті замовлення страв. Зазвичай, таблиця products використовується для відстеження наявності та характеристик

продуктів, а таблиця order - для відстеження замовлень і пов'язаних з ними даних.

Ці зв'язки допомагають вам створювати складні запити, які об'єднують дані з різних таблиць для отримання потрібної інформації. Наприклад, ви можете отримати всі продукти, які використовуються у певній страві, або всі страви, які були замовлені на певну дату.

Розпочнемо з створення таблиці "dish" з вказаними полями:

id_dish (Ціле число): Унікальний ідентифікатор страви. Це поле буде використовуватися для ідентифікації кожної конкретної страви у базі даних.

name_dish (Рядок): Назва страви. В цьому полі зберігатиметься інформація про назву конкретної страви.

cost_calculation (Число з рухомою комою): Вартість обчислення. Це поле може використовуватися для зберігання витрат на приготування страви або вартості інгредієнтів.

marup_cost_dish (Число з рухомою комою): Маржинальна вартість страви. Це поле може використовуватися для зберігання додаткових витрат на приготування страви, таких як зарплата персоналу або витрати на упаковку.

category_dish (Рядок): Категорія страви. Це поле дозволяє класифікувати страви за їхнім типом або призначенням, наприклад, супи, основні страви, десерти тощо.

dish_cost (Число з рухомою комою): Вартість страви. Це поле міститиме загальну вартість страви для кінцевого користувача.

Ось SQL-запит для створення таблиці "dish" з вказаними полями:

```
CREATE TABLE dish (
    id_dish INT PRIMARY KEY AUTO_INCREMENT,
    name_dish VARCHAR(255) NOT NULL,
    cost_calculation DECIMAL(10, 2),
    marup_cost_dish DECIMAL(10, 2),
    category_dish VARCHAR(255),
    dish_cost DECIMAL(10, 2) NOT NULL
);
```

Тепер продовжимо з таблицею "order":

id_order (Ціле число): Унікальний ідентифікатор замовлення. Це поле буде використовуватися для ідентифікації кожного окремого замовлення.

order_date (Дата або строка): Дата замовлення. В цьому полі буде зберігатися дата, коли було зроблено замовлення.

number_table_order (Ціле число): Номер столика. Це поле дозволяє вказати, за яким столиком було здійснено замовлення.

waiter_name (Рядок): Ім'я офіціанта. Це поле містить інформацію про офіціанта, який обслуговував замовлення.

SQL-запит для створення таблиці "order":

```
CREATE TABLE order (
    id_order INT PRIMARY KEY AUTO_INCREMENT,
    order_date DATE NOT NULL,
    number_table_order INT NOT NULL,
    waiter_name VARCHAR(255)
);
```

Тепер перейдемо до таблиці "products":

id_product (Ціле число): Унікальний ідентифікатор продукту. Це поле буде використовуватися для ідентифікації кожного конкретного продукту.

name_product (Рядок): Назва продукту. В цьому полі зберігатиметься інформація про назву продукту.

cost_product (Число з рухомою комою): Вартість продукту. Це поле містить ціну продукту за одиницю.

numb_product (Ціле число): Кількість продукту. Це поле вказує кількість одиниць продукту на складі.

unit_product (Рядок): Одиниця виміру продукту. Це поле містить інформацію про одиниці виміру продукту, наприклад, кг, шт, л.

actual_numb_product (Ціле число): Фактична кількість продукту. Це поле може використовуватися для відображення кількості продукту після замовлення.

SQL-запит для створення таблиці "products":

```
CREATE TABLE products (
    id_product INT PRIMARY KEY AUTO_INCREMENT,
```

```
    name_product VARCHAR(255) NOT NULL,
    cost_product DECIMAL(10, 2) NOT NULL,
    numb_product INT NOT NULL,
    unit_product VARCHAR(50),
    actual_numb_product INT
);
```

Продовжимо з таблицею "placing_order":

pl_id_order (Ціле число): Унікальний ідентифікатор замовлення. Це поле вказує на конкретне замовлення, до якого відноситься певна страва.

pl_id_dish (Ціле число): Унікальний ідентифікатор страви. Це поле вказує на конкретну страву, яка була замовлена.

numb (Ціле число): Кількість страв. Це поле вказує кількість замовлених страв.

SQL-запит для створення таблиці "placing_order":

```
CREATE TABLE placing_order (
    pl_id_order INT,
    pl_id_dish INT,
    numb INT,
    FOREIGN KEY (pl_id_order) REFERENCES order(id_order),
    FOREIGN KEY (pl_id_dish) REFERENCES dish(id_dish)
);
```

Наостанок, таблиця "dish_composition":

comp_id_dish (Ціле число): Унікальний ідентифікатор страви. Це поле вказує на конкретну страву, до якої входять інгредієнти.

comp_id_products (Ціле число): Унікальний ідентифікатор продукту. Це поле вказує на конкретний продукт, який входить до складу страви.

count_product (Ціле число): Кількість продукту. Це поле вказує кількість одиниць продукту, що використовуються для приготування страви.

SQL-запит для створення таблиці "dish_composition":

```
CREATE TABLE dish_composition (
    comp_id_dish INT,
    comp_id_products INT,
    count_product INT,
    FOREIGN KEY (comp_id_dish) REFERENCES dish(id_dish),
    FOREIGN KEY (comp_id_products) REFERENCES products(id_product)
);
```

Ці SQL-запити створять всі необхідні таблиці з вказаними полями та зв'язками між ними.

Вид схеми даних створеної БД представлено на рисунку 3.1.

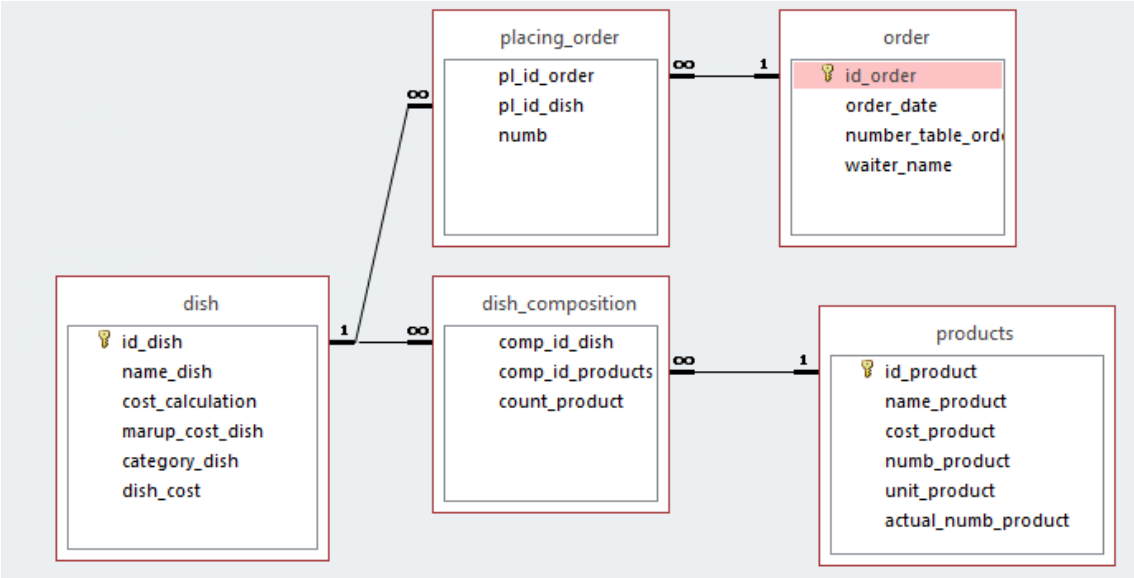


Рисунок 3.1 – Ілюстрація того, як активності утворюють відгалуження у макеті та містять активності

3.2 Розробка та функціонування системи обліку та обігу продукції в готельно ресторанному бізнесі.

Згідно структури, яка зображена на рисунку 2.4 розроблено головне вікно програми, яке відкривається при запуску програми.

Для розробки використовувалося VS кроки створення форми Welcome в Visual Studio з використанням C++/CLI:

Оберіть "Visual C++" -> "Windows Desktop" -> "Windows Forms App (.NET Framework)".

Введіть ім'я проекту, наприклад, "WelcomeFormApp".

Натисніть кнопку "Створити".

Створення форми Welcome:

У Solution Explorer виберіть ваш проект.

Натисніть правою кнопкою миші на папці "Source Files".

Оберіть "Додати" -> "Новий елемент".

Оберіть "Windows Form" і дайте йому ім'я, наприклад, "WelcomeForm.h".

Натисніть "Додати".

Додавання елементів на форму:

Відкрийте WelcomeForm.h.

Відкрийте вікно конструктора форми, натиснувши на неї двічі в редакторі коду.

Перетягніть кнопки (Button) з панелі інструментів на форму.

Змініть властивості кнопок (Text, Name, Location, Size тощо) за допомогою вікна властивостей внизу.

Створення обробників подій для кнопок:

Перейдіть до вікна конструктора форми.

Виберіть кнопку, для якої потрібно додати обробник подій.

Двічі клацніть на кнопці, щоб створити обробник подій.

Введіть код обробника подій у вікні коду, що відкрилося.

Отже, ось приклад коду для створення обробників подій для кнопок:

```
// Обробник події для кнопки, яка відкриває форму PlaceOrder
private:    System::Void    btnPlaceOrder_Click(System::Object^    sender,
System::EventArgs^ e) {
    PlaceOrderForm^ placeOrderForm = gcnew PlaceOrderForm();
    placeOrderForm->Show();
}

// Обробник події для кнопки, яка відкриває форму Dani
private:    System::Void    btnOpenDani_Click(System::Object^    sender,
System::EventArgs^ e) {
    DaniForm^ daniForm = gcnew DaniForm();
    daniForm->Show();
}
```

Lодати #include для форм PlaceOrderForm.h і DaniForm.h вверху файлу форми WelcomeForm.h.

Цей код створить обробники подій для кнопок, які відкриватимуть форми PlaceOrderForm і DaniForm, відповідно, коли користувач натисне на них.

Вигляд форми в режимі конструкто представлено на рисунку 3.2.

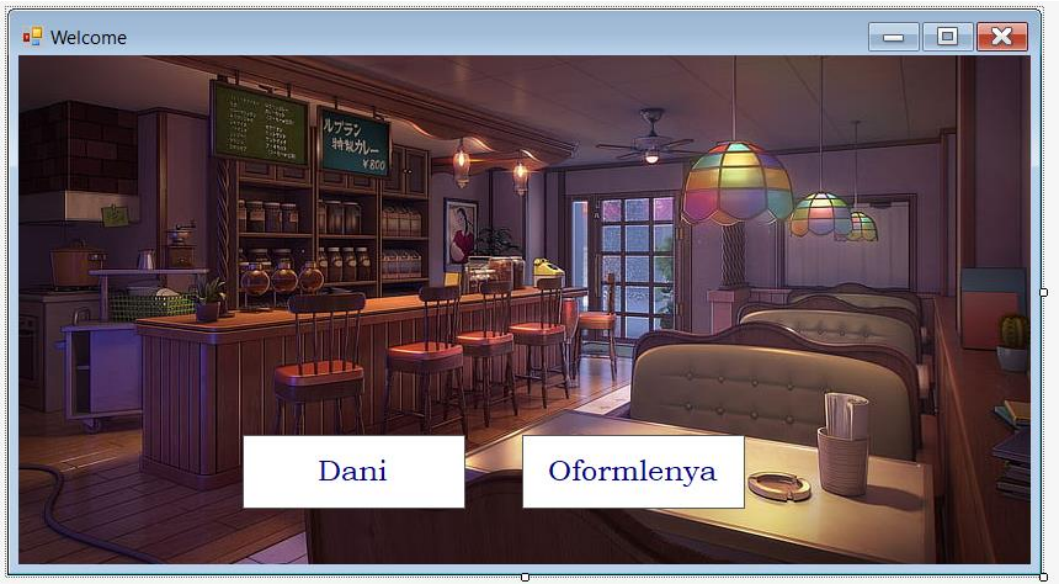


Рисунок 3.1 – Ілюстрація форми Welcome в режимі конструктура

Також було розроблено форму Dani і методи вкладки та меню.

Ось кроки створення форми "Dani" з використанням Visual Studio та C++/CLI:

Створення форми Dani:

Форма "Dani" була успішно створена в середовищі Visual Studio за допомогою майстра створення нової форми. На формі були додані всі необхідні елементи, такі як MenuStrip, ToolStripMenuItem, DataGridView та TextBox.

Додавання обробників подій для кнопок і меню:

Для кожної кнопки та пункта меню на формі були успішно додані обробники подій, які відповідають за виконання потрібних дій. Наприклад, обробник події для кнопки "Додати" відповідає за додавання нового запису у базу даних.

Також для зменшення навантаженості і надлишковості зроблено вкладки для швидкого переходу між ними, вкладки складаються з таких активностей:

- Продукти;
- Страви;
- Замовлення;

– Вибірка.

Вкладки продемостровано в режимі конструктора на рисунку 3.2.

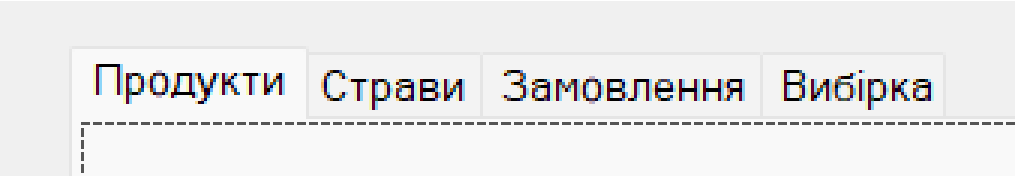


Рисунок 3.2 – Ілюстрація вкладок в режимі коструктора

Налаштування DataGridView:

DataGridView на формі "Dani" був налаштований для відображення даних з бази даних. Всі необхідні колонки були налаштовані, включаючи назву, розмір та тип даних.

Написання коду для роботи з базою даних:

Був написаний код для підключення до бази даних та виконання запитів для оновлення, видалення, додавання даних і т. д. Також був написаний код для пошуку за критеріями.

Розміщення елементів на формі:

Усі елементи були розміщені на формі "Dani" таким чином, щоб вони були зручні для користувача та логічно розташовані.

Відлагодження і тестування:

Після створення форми "Dani" було проведено відлагодження та тестування програми. Впевнилися, що всі кнопки виконують потрібні дії і що дані коректно відображаються в DataGridView.

Отже, форма "Dani" була успішно створена та протестована, і готова до використання в програмі для управління базою даних.

метод Dani_Load викликається при завантаженні форми і виконує наступні дії:

Створення підключення до бази даних:

Спочатку створюється об'єкт OleDbConnection з вказаною рядком підключення до бази даних "restApp.mdb".

Вибірка даних з таблиці "products":

Виконується SQL-запит для вибірки всіх даних з таблиці "products".

Результати запиту зберігаються в об'єкті DataTable під назвою ProductsTable.

Створюються колонки для ProductsTable на основі назв стовпців у вибірці даних.

Кожен рядок результату запиту додається в ProductsTable.

Відображення даних таблиці "products" у DataGridView контролі:

DataSource для DataGridView контролю ProductsDataGridView встановлюється на ProductsTable.

Вибірка даних з таблиці "dish":

Аналогічні операції виконуються для таблиці "dish". Результати зберігаються в DataTable під назвою DishTable.

Відображення даних таблиці "dish" у DataGridView контролі:

DataSource для DataGridView контролю DishDataGridView встановлюється на DishTable.

Вибірка даних з таблиці "order":

Аналогічні операції виконуються для таблиці "order". Результати зберігаються в DataTable під назвою OrderTable.

Відображення даних таблиці "order" у DataGridView контролі:

DataSource для DataGridView контролю OrderDataGridView встановлюється на OrderTable.

Закриття підключення до бази даних:

Після завершення вибірки даних з кожної таблиці підключення до бази даних закривається.

Отже, цей метод завантажує дані з таблиць "products", "dish" та "order" і відображає їх у відповідних DataGridView контролях на вашій формі.

Лістинг коду методу наведено нижче:

```
      private: System::Void Dani_Load(System::Object^  sender,
System::EventArgs^  e) {
```

```
                        auto OleDbConnection1 = gcnew
OleDb::OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\\restApp.mdb");
                    OleDbConnection1->Open();
                    auto comm1 = gcnew OleDb::OleDbCommand("Select *
From [products]", OleDbConnection1);
                    auto ExecuteReader1 = comm1-> ExecuteReader();
                    auto ProductsTable = gcnew DataTable();
                    ProductsTable->Columns->Add(ExecuteReader1-
>GetName(0));
                        ProductsTable->Columns->Add(ExecuteReader1-
>GetName(1));
                        ProductsTable->Columns->Add(ExecuteReader1-
>GetName(2));
                        ProductsTable->Columns->Add(ExecuteReader1-
>GetName(3));
                        ProductsTable->Columns->Add(ExecuteReader1-
>GetName(4));
                        ProductsTable->Columns->Add(ExecuteReader1-
>GetName(5));
                    while (ExecuteReader1->Read() == true)
                        ProductsTable->Rows->Add(ExecuteReader1-
>GetValue(0),
                        ExecuteReader1->GetValue(1), ExecuteReader1-
>GetValue(2),
                                ExecuteReader1->GetValue(3),
ExecuteReader1->GetValue(4),
                                ExecuteReader1->GetValue(5));
                    ExecuteReader1->Close();
                        OleDbConnection1->Close();
                        ProductsDataGridView->DataSource =
ProductsTable;
                        OleDbConnection1->Open();
                    auto comm2 = gcnew OleDb::OleDbCommand("Select *
From [dish]", OleDbConnection1);
                    auto ExecuteReader2 = comm2-> ExecuteReader();
                    auto DishTable = gcnew DataTable();
                    DishTable->Columns->Add(ExecuteReader2->GetName(0));
                        DishTable->Columns->Add(ExecuteReader2-
>GetName(1));
                        DishTable->Columns->Add(ExecuteReader2-
>GetName(2));
                        DishTable->Columns->Add(ExecuteReader2-
>GetName(3));
                        DishTable->Columns->Add(ExecuteReader2-
>GetName(4));
                        DishTable->Columns->Add(ExecuteReader2-
>GetName(5));
                    while (ExecuteReader2->Read() == true)
                        DishTable->Rows->Add(ExecuteReader2-
>GetValue(0),
                        ExecuteReader2->GetValue(1), ExecuteReader2-
>GetValue(2),
                                ExecuteReader2->GetValue(3),
ExecuteReader2->GetValue(4),
                                ExecuteReader2->GetValue(5));
                    ExecuteReader2->Close();
                        OleDbConnection1->Close();
                        DishDataGridView->DataSource = DishTable;
                        OleDbConnection1->Open();
                    auto comm3 = gcnew OleDb::OleDbCommand("Select *
From [order]", OleDbConnection1);
                    auto ExecuteReader3 = comm3-> ExecuteReader();
                    auto OrderTable = gcnew DataTable();
```

```
                    OrderTable->Columns->Add(ExecuteReader3-
>GetName(0));
                        OrderTable->Columns->Add(ExecuteReader3-
>GetName(1));
                        OrderTable->Columns->Add(ExecuteReader3-
>GetName(2));
                        OrderTable->Columns->Add(ExecuteReader3-
>GetName(3));
                    while (ExecuteReader3->Read() == true)
                        OrderTable->Rows->Add(ExecuteReader3-
>GetValue(0),
                        ExecuteReader3->GetValue(1), ExecuteReader3-
>GetValue(2),
                                ExecuteReader3->GetValue(3));
                    ExecuteReader3->Close();
                        OleDbConnection1->Close();
                        OrderDataGridView->DataSource = OrderTable;
                      }
```

Лістинги даного класу представлено у додатку А.

Вигляд форми в режимі конструктора представлено на рисунку 3.3.

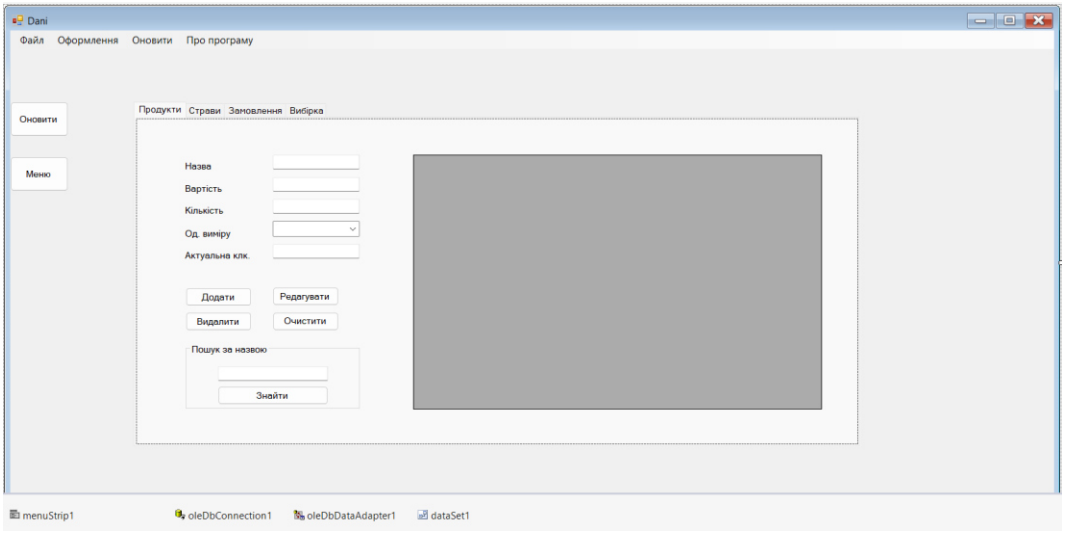


Рисунок 3.3 – Ілюстрація форми Dani в режимі конструктура

Звідси й далі будемо описувати процес створення форми "Place Order" з використанням Visual Studio та C++/CLI.

Створення форми "Place Order":

Користувач відкриває середовище розробки Visual Studio та створює нову форму з назвою "Place Order".

На цій формі відображатимуться доступні страви для замовлення, а також інша інформація, пов'язана з оформленням замовлення.

Створення замовлення:

Користувач запускає форму "Place Order" та переходить до розділу створення нового замовлення.

На формі користувач заповнює необхідну інформацію про замовлення, таку як дата, номер столика та ім'я офіціанта.

Вибір страв для замовлення:

Після створення замовлення, користувач переходить до розділу вибору страв.

На формі відображається список доступних страв з бази даних.

Користувач може вибрати потрібні страви, встановити їх кількість та додати до замовлення.

Формування чеку:

Після додавання усіх необхідних страв до замовлення, користувач переходить до розділу формування чеку.

На формі автоматично обчислюється загальна вартість замовлення на основі вибраних страв та їх кількості.

Користувач може переглянути усі деталі замовлення та впевнитися в їх правильності.

Оформлення замовлення:

Після перевірки та підтвердження всіх деталей замовлення, користувач натискає кнопку "Підтвердити замовлення".

Форма "Place Order" виконує необхідні операції для оформлення замовлення, включаючи збереження даних у базу даних.

Користувач отримує повідомлення про успішне оформлення замовлення та може продовжити роботу з програмою.

Отже, після виконання цих кроків, користувач успішно оформив замовлення на формі "Place Order" та отримав відповідне повідомлення про це.

Лістинги даних класів представлено у додатку Б.

Вигляд форми в режимі конструктор представлено на рисунку 3.4.

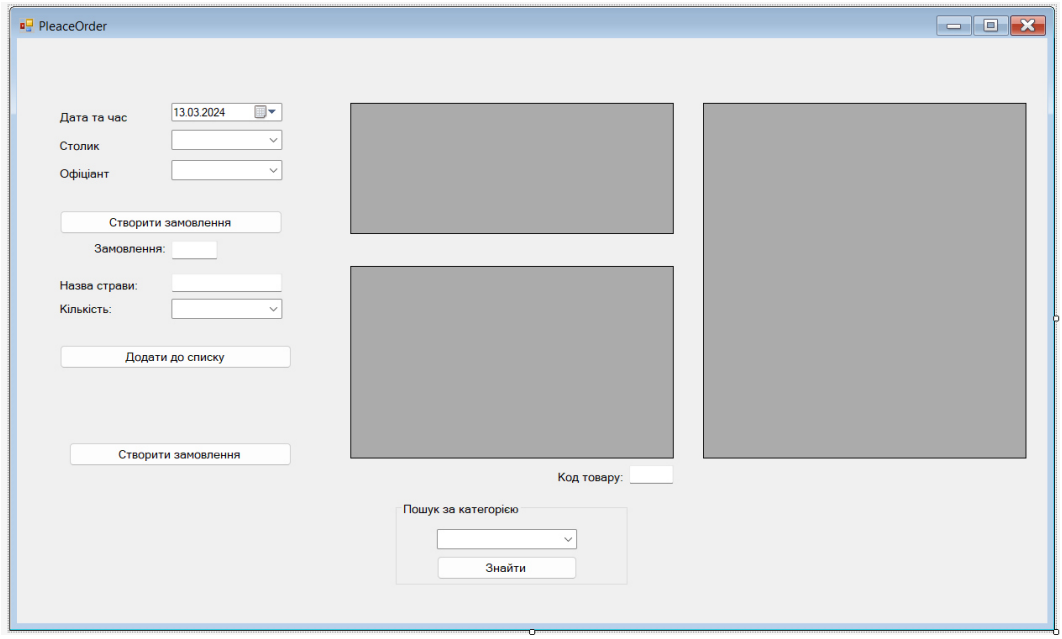


Рисунок 3.4 –Ілюстрація форми Place Order в режимі конструктура

Під час створення програми зазвичай необхідно обробляти винятки (ексепшени), які можуть виникнути внаслідок різних ситуацій, таких як пусті поля, невірні дані або невірний ввід користувача. Нижче наведено опис того, як ми можемо обробляти такі випадки на формі "Дані" та формі оформлення замовлення.

1. **Обробка ексепшинів на формі "Дані"**:

- Коли користувач намагається додати новий продукт або страву, програма повинна перевіряти, чи всі поля заповнені.

- При натисканні кнопки "Додати" програма перевіряє, чи всі необхідні поля (наприклад, назва продукту або страви, вартість тощо) були заповнені користувачем.

- Якщо яке-небудь поле залишилося порожнім, програма видає повідомлення про помилку або виводить відповідне сповіщення на екран, щоб повідомити користувача про необхідність заповнення всіх полів перед додаванням нового елементу.

2. **Обробка ексепшинів на формі оформлення замовлення**:

- При оформленні замовлення також необхідно перевіряти, чи користувач вибрав необхідні страви та вказав усю необхідну інформацію про замовлення.

- При натисканні кнопки "Підтвердити замовлення" програма перевіряє, чи всі необхідні дані заповнені користувачем, такі як дата, номер столика, вибрані страви тощо.

- Якщо яке-небудь поле залишилося порожнім або не вибрані страви, програма видає повідомлення про помилку або виводить відповідне сповіщення на екран, щоб повідомити користувача про необхідність заповнення всіх полів та вибору страв перед оформленням замовлення.

Завдяки обробці ексепшинів програма забезпечує коректну роботу та запобігає непередбаченим ситуаціям, які можуть виникнути через неправильний ввід користувача або інші причини.

Давайте розглядемо по одному з методів на кожній з форм та опишемо.

3.3 З’єднання та функціонування провайдера з БД

Ініціалізація або встановлення з’єднання є типовим шаблоном у багатьох областях програмування Windows. Якщо ви хочете почати малювати у вікні, вам потрібно спочатку отримати контекст пристрою, викликавши GetDCабо CreateDC, а коли ви закінчите, закрийте його, викликавши ReleaseDCабо DeleteDC. Якщо ви хочете розпочати розмову між клієнтом і сервером за допомогою сокетів Windows, спочатку встановіть з’єднання, а коли закінчите, закрийте його. За аналогічною схемою, якщо ви хочете запитати базу даних щодо певних записів, вам потрібно встановити з’єднання, а коли ви закінчите, закрийте його. У цій статті я покажу вам, як почати з’єднання за допомогою споживчих класів ATL OLE DB і як отримати сеанс, щоб ви могли використовувати його для запитів або редагування бази даних.

Щоб ініціалізувати підключення до бази даних за допомогою постачальника OLE DB, вам потрібні два класи ATL: CDataSourcei CSession. Щоб використовувати їх, додайте файл atldbcli.h :

#include <atldbcli.h>

Чому два класи? Це тому, що ви можете створити багато сеансів на одному з'єднанні. Щойно ви відкриєте джерело даних за допомогою методів Openабо OpenFromInitializationString, швидше за все, ви також захочете розпочати сеанс. Під час цього сеансу ви можете створювати транзакції та робити запити або редагувати базу даних. Це дійсно необхідно, навіть якщо ви використовуєте класи схеми. Існує лише одне перевантаження для OpenFromInitializationStringметоду та дев'ять перевантажень для Openметоду.

Я поясню всі Open перевантаження методів, але здебільшого використовуватиму перший, якщо мені потрібно відобразити діалогове вікно властивостей посилання даних, щоб дозволити користувачеві вибрати постачальника OLE DB і властивості його з'єднання.

```
HRESULT Open(HWND hWnd = GetActiveWindow(), DBPROMPTOPTIONS dwPromptOptions
= DBPROMPTOPTIONS_WIZARDSHEET);
HRESULT Open(const CLSID& clsid, DBPROPSET* pPropSet = NULL, ULONG
nPropertySets=1);
HRESULT Open(const CLSID& clsid, LPCTSTR pName, LPCTSTR pUserName = NULL,
LPCTSTR pPassword = NULL, long nInitMode = 0);
HRESULT Open(LPCSTR szProgID, DBPROPSET* pPropSet = NULL, ULONG
nPropertySets=1);
HRESULT Open(LPCWSTR szProgID, LPCTSTR pName, LPCTSTR pUserName = NULL,
LPCTSTR pPassword = NULL, long nInitMode = 0);
HRESULT Open(LPCWSTR szProgID, DBPROPSET* pPropSet = NULL, ULONG
nPropertySets=1);
HRESULT Open(LPCSTR szProgID, LPCTSTR pName, LPCTSTR pUserName = NULL,
LPCTSTR pPassword = NULL, long nInitMode = 0);
HRESULT Open(const CEnumerator& enumerator, DBPROPSET* pPropSet = NULL,
ULONG nPropertySets=1);
HRESULT Open(const CEnumerator& enumerator, LPCTSTR pName, LPCTSTR
pUserName = NULL, LPCTSTR pPassword = NULL, long nInitMode = 0);
```

Зауважте, що споживчі класи ATL базуються на COM, а це означає:

Ми повинні викликати CoInitialize перед використанням цих класів, ймовірно, у InitInstance, а потім подібним чином викликати CoUninitializeпрограму в ExitInstance.

Нам потрібно буде мати справу з HRESULTповерненими значеннями як показниками успіху чи невдачі.

Перерахування постачальників OLE DB за допомогою класу CEnumerator

```
CEnumerator oProviders;

HRESULT hr = oProviders.Open( );
if(SUCCEEDED(hr))
{
    // The following macro is to initialize
    // the conversion routines
    USES_CONVERSION;

    while(oProviders.MoveNext( ) == S_OK)
    {
// Now you have the provider name
// in oProviders.m_szName and description
// in oProviders.m_szDescription

#ifdef _UNICODE
        TRACE(oProviders.m_szName);
        TRACE(L"\n");
#else
        TRACE(W2A(oProviders.m_szName));
        TRACE("\n");
#endif
    }
    oProviders.Close( );
}
```

CEnumeratorклас надає засоби для перерахування всіх постачальників OLE DB, встановлених у системі. Це зручний спосіб надання користувачеві спеціального інтерфейсу для вибору постачальника та його властивостей підключення. Якщо вам потрібно перерахувати постачальників OLE DB за допомогою стандартного діалогового вікна, просто викличте Openметод класу CDataSourceбез будь-яких параметрів.

Це метод Dani_Load викликається при завантаженні форми і виконує наступні дії:

Створення підключення до бази даних:

Спочатку створюється об'єкт OleDbConnection з вказаною рядком підключення до бази даних "restApp.mdb".

Вибірка даних з таблиці "products":

Виконується SQL-запит для вибірки всіх даних з таблиці "products".

Результати запиту зберігаються в об'єкті DataTable під назвою ProductsTable.

Створюються колонки для ProductsTable на основі назв стовпців у вибірці даних.

Кожен рядок результату запиту додається в ProductsTable.

Відображення даних таблиці "products" у DataGridView контролі:

DataSource для DataGridView контролю ProductsDataGridView встановлюється на ProductsTable.

Вибірка даних з таблиці "dish":

Аналогічні операції виконуються для таблиці "dish". Результати зберігаються в DataTable під назвою DishTable.

Відображення даних таблиці "dish" у DataGridView контролі:

DataSource для DataGridView контролю DishDataGridView встановлюється на DishTable.

Вибірка даних з таблиці "order":

Аналогічні операції виконуються для таблиці "order". Результати зберігаються в DataTable під назвою OrderTable.

Відображення даних таблиці "order" у DataGridView контролі:

DataSource для DataGridView контролю OrderDataGridView встановлюється на OrderTable.

Закриття підключення до бази даних:

Після завершення вибірки даних з кожної таблиці підключення до бази даних закривається.

Отже, цей метод завантажує дані з таблиць "products", "dish" та "order" і відображає їх у відповідних DataGridView контролях на вашій формі.

```
private: System::Void Dani_Load(System::Object^  sender, System::EventArgs^  e) {
                    auto OleDbConnection1 = gcnew
OleDb::OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\\restApp.mdb");
                OleDbConnection1->Open();
                auto comm1 = gcnew OleDb::OleDbCommand("Select * From
[products]", OleDbConnection1);
                auto ExecuteReader1 = comm1-> ExecuteReader();
                auto ProductsTable = gcnew DataTable();
                ProductsTable->Columns->Add(ExecuteReader1->GetName(0));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(1));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(2));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(3));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(4));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(5));
```

```
                while (ExecuteReader1->Read() == true)
                    ProductsTable->Rows->Add(ExecuteReader1->GetValue(0),
                    ExecuteReader1->GetValue(1), ExecuteReader1->GetValue(2),
                            ExecuteReader1->GetValue(3), ExecuteReader1-
>GetValue(4),
                            ExecuteReader1->GetValue(5));
                ExecuteReader1->Close();
                        OleDbConnection1->Close();
                        ProductsDataGridView->DataSource = ProductsTable;
                        OleDbConnection1->Open();
                auto comm2 = gcnew OleDb::OleDbCommand("Select * From [dish]",
OleDbConnection1);
                auto ExecuteReader2 = comm2-> ExecuteReader();
                auto DishTable = gcnew DataTable();
                DishTable->Columns->Add(ExecuteReader2->GetName(0));
                        DishTable->Columns->Add(ExecuteReader2->GetName(1));
                        DishTable->Columns->Add(ExecuteReader2->GetName(2));
                        DishTable->Columns->Add(ExecuteReader2->GetName(3));
                        DishTable->Columns->Add(ExecuteReader2->GetName(4));
                        DishTable->Columns->Add(ExecuteReader2->GetName(5));
                while (ExecuteReader2->Read() == true)
                    DishTable->Rows->Add(ExecuteReader2->GetValue(0),
                    ExecuteReader2->GetValue(1), ExecuteReader2->GetValue(2),
                            ExecuteReader2->GetValue(3), ExecuteReader2-
>GetValue(4),
                            ExecuteReader2->GetValue(5));
                ExecuteReader2->Close();
                        OleDbConnection1->Close();
                        DishDataGridView->DataSource = DishTable;

                        ////////////////////////////////////

                        OleDbConnection1->Open();
                auto comm3 = gcnew OleDb::OleDbCommand("Select * From [order]",
OleDbConnection1);
                auto ExecuteReader3 = comm3-> ExecuteReader();
                auto OrderTable = gcnew DataTable();
                OrderTable->Columns->Add(ExecuteReader3->GetName(0));
                        OrderTable->Columns->Add(ExecuteReader3->GetName(1));
                        OrderTable->Columns->Add(ExecuteReader3->GetName(2));
                        OrderTable->Columns->Add(ExecuteReader3->GetName(3));
                while (ExecuteReader3->Read() == true)
                    OrderTable->Rows->Add(ExecuteReader3->GetValue(0),
                    ExecuteReader3->GetValue(1), ExecuteReader3->GetValue(2),
                            ExecuteReader3->GetValue(3));
                ExecuteReader3->Close();
                        OleDbConnection1->Close();
                        OrderDataGridView->DataSource = OrderTable;



                        }
```

Лістинги даного класу представлено у додатку В.

3.4 Тестування системи

Сьогодні є безліч фреймворків для тестування, що підтримують практично всі існуючі мови. Здавалося б - можна брати і автоматизувати. Але навіть зараз ручні тести важливі.

Одне з пояснень їх необхідності полягає в тому в тому, що при ручному тестуванні функціоналу ми можемо набагато швидше отримати інформацію про стан продукту, який аналізуємо, про якість розробки. Крім того, при автоматизації попередньо розроблені кейси часто доводиться міняти і актуалізувати, а на написання Автотест потрібен певний час.

Все це означає, що головна мета ручних тестів - попередньо переконатися в тому, що заявлений функціонал працездатний, не має помилок і видає очікувані, заплановані результати. Без них не можна бути впевненим у тому, що можна працювати далі. Особливо це актуально для функцій, на реалізацію яких зав'язана подальша розробка. В такому випадку метушня зі створенням Автотест на ці фічі стає блокуючим фактором для всієї розробки продукту, зрушуючи терміни і зриваючи дедлайни. Момент, коли кейси прийде пора автоматизувати, все одно рано чи пізно настане - але не варто прагнути наблизити його штучно в гонитві за тотальним винятком ручної праці.

Під час використання форми "Дані" користувач може здійснювати маніпуляції з даними, такі як додавання нових записів або редагування існуючих. Ось як можна описати цей процес з точки зору третьої особи:

На формі "Дані" користувач зустрічається з таблицею, що містить записи інформації, такої як продукти, страви чи замовлення. При відкритті форми користувач має можливість переглядати наявні записи та виконувати різні дії з ними.

Вибір запису:

Користувач може обирати запис, щоб відредагувати або переглянути деталі. Для цього він використовує вбудовані елементи інтерфейсу, такі як DataGridView, який дозволяє відображати дані у вигляді таблиці. Вибір запису здійснюється шляхом кліку на відповідний рядок у таблиці.

Відображення даних представлено на рисунку 3.5.

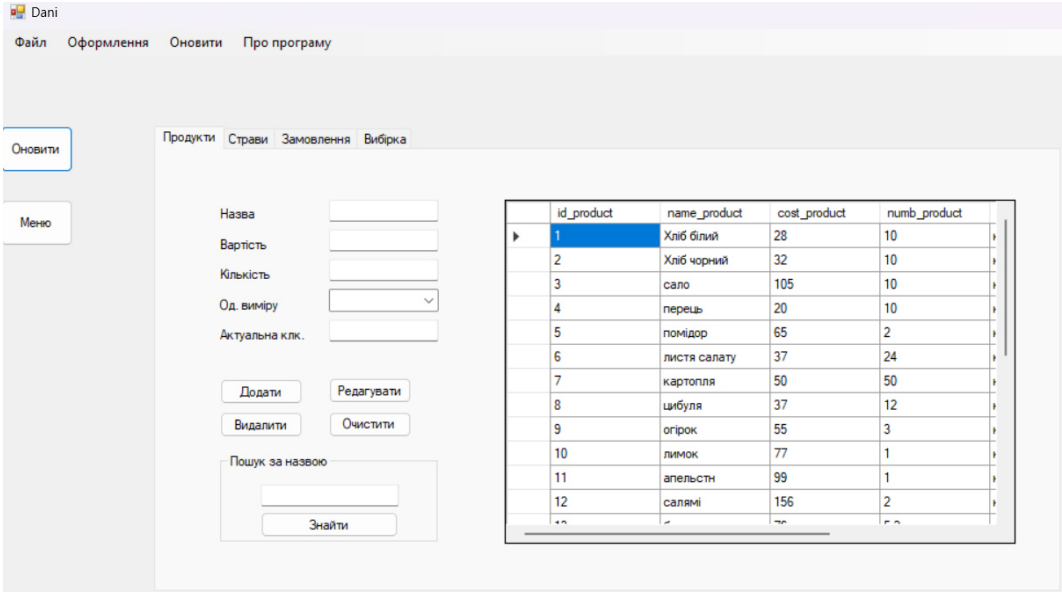


Рисунок 3.5 – Ілюстрація відображення даних на вкладці Продукти

Додавання нового запису:

Користувач може додавати новий запис, натиснувши на кнопку "Додати" або відповідний пункт меню. Після цього відкривається форма для введення нових даних.

При спробі зберегти новий запис програма автоматично перевіряє, чи всі поля заповнені. Якщо яке-небудь поле залишилося порожнім, виводиться повідомлення про необхідність заповнення всіх полів перед збереженням.

Повідомлення про помилку продемострировано на рисунку 3.6.

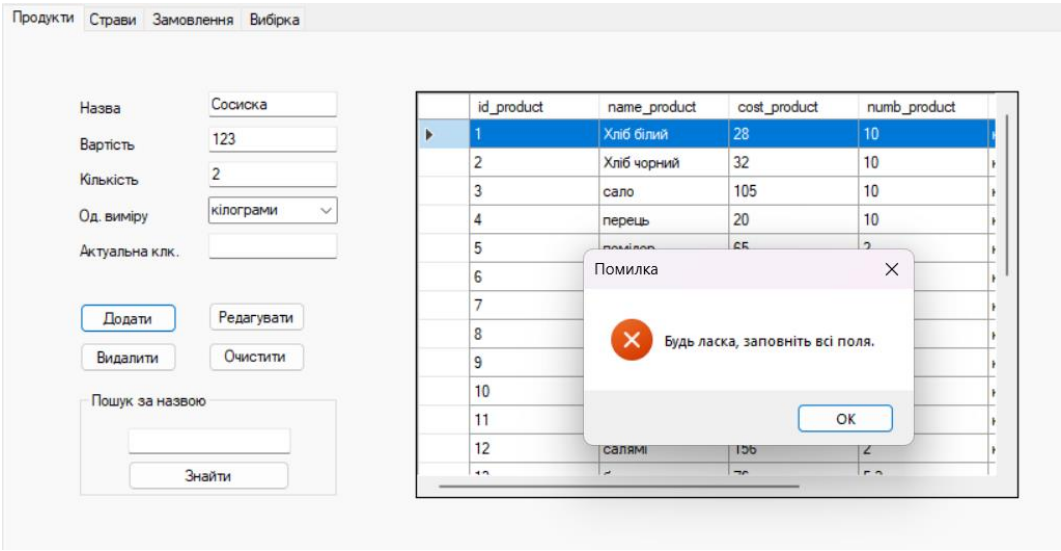


Рисунок 3.6 – Ілюстрація помилки про незаповненість полів.

Успішне додавання даних:

Якщо всі поля заповнені відповідними даними, і користувач натиснув кнопку "Зберегти" або "ОК", програма зберігає новий запис і виводить повідомлення про успішне додавання даних. Це підтверджує користувачеві, що його дії були успішно виконані.

Повідомлення про успішне додавання даних до БД продемостровано на рисунку 3.7.

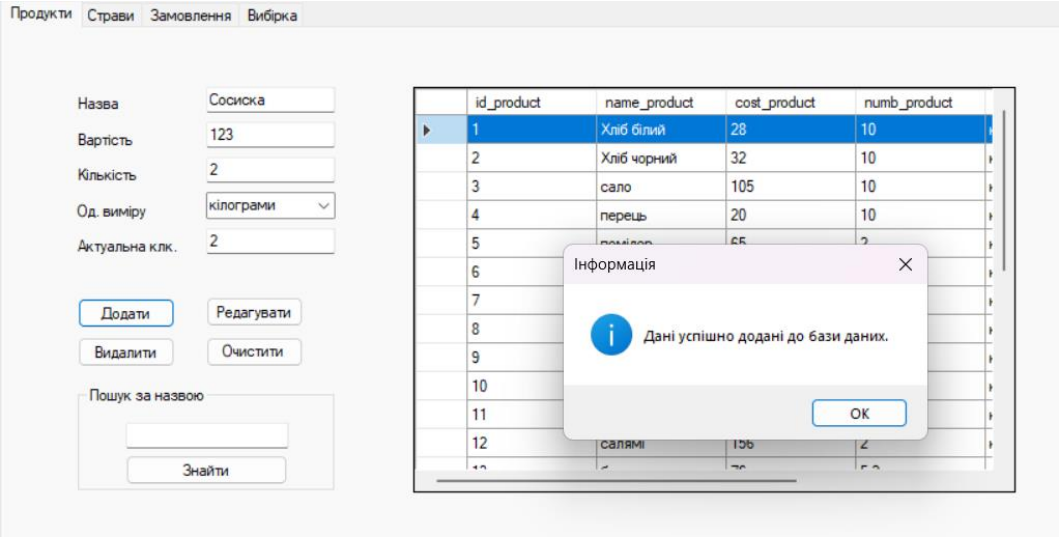


Рисунок 3.7 – Ілюстрація повідомлення про успішне додавання запису в таблицю Продукти

Таким чином, форма "Дані" надає користувачеві зручний інтерфейс для роботи з даними, а обробка помилок та відображення повідомлень про успішні операції забезпечує йому зрозумілу взаємодію з програмою.

Ось опис процесу оформлення замовлення на формі з точки зору третьої особи:

Вибір існуючого замовлення або створення нового:

При відкритті форми оформлення користувач має можливість обрати існуюче замовлення зі списку або створити нове. Для цього він використовує

відповідні елементи інтерфейсу, такі як ComboBox для вибору замовлення або кнопку "Створити нове замовлення".

Перевірка наявності обраного замовлення:

Якщо користувач обрав існуюче замовлення зі списку, програма перевіряє, чи було обрано замовлення. Якщо замовлення не обрано, виводиться відповідне повідомлення про помилку, яке повідомляє користувача про необхідність обрати замовлення перед продовженням.

Сповіщення про помилки наведені на зображені в додатку В.

Після успішного створення замовлення в полі номер замовлення заповняється номером та можна обрати страву і її кількість.

Представлення процесу продовження оформлення замовлення зображено на рисунку в додатку В.

Перевірка наявності обраних страв:

Після обрання замовлення користувач може вибрати страви для додавання до замовлення. Якщо користувач не обрав жодної страви, програма виводить повідомлення про помилку, що попереджує користувача про необхідність обрати хоча б одну страву перед створенням замовлення.

Результат роботи перевірки наведено на рисунках в додатку В.

Формування замовлення:

Якщо користувач обрав замовлення та вибрав хоча б одну страву, програма формує замовлення. Вона збирає всю необхідну інформацію про замовлення, таку як дата оформлення, номер столика, вибрані страви та їх кількість.

Після цього програма виводить усю інформацію про замовлення на екран користувачу. Вона підтверджує успішне створення замовлення та вказує загальну оплату замовлення.

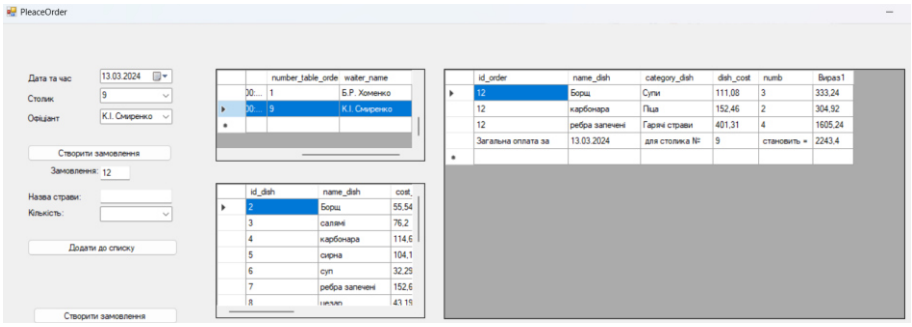


Рисунок 3.12 – Ілюстрація результат успішного оформлення замовлення

Завершення процесу оформлення:

Після відображення інформації про замовлення користувач має можливість продовжити роботу з програмою або закрити форму оформлення.

Якщо користувач вирішить закрити форму, він повертається до попередньої форми з можливістю продовжити роботу з програмою.

3.5 Висновки до розділу 3

В даному розділі представлено етапи розробки інформаційної системи та покрокове виконання.

Розробку розпочата із так званої БД де зберігаються дані про записи, також в системі виділено бізнес-логіку у вигляді сутностей класів, покроково описано розробку пакетів та класів даного модуля.

Останнім підпунктом розробки системи за допомогою так званих екранів активностей котрі взаємодіють між собою, а також між БД, отримуючи дані та відображаючи їх.

Дана система протестована за допомогою ручного тестування.

4 ОХОРОНА ПРАЦІ

4.1 Значення охорони праці і навколишнього середовища в забезпечені безпечних і здорових умов праці

Проблема охорони праці набуває особливого значення в умовах сучасного виробничого середовища. Складність технологічних систем та процесів ставить підвищенні вимоги до організму людини, їй доводиться діяти на межі своїх фізичних та психологічних можливостей. В таких умовах людина не завжди може досконало сприймати швидкі зміни обставин в процесі виробничої діяльності і адекватно на них реагувати. Навіть звичайна праця у науковому відділі вже стає небезпечною для здоров'я працівника, тому що при цьому використовуються персональні обчислювальні машини (ПЕОМ), факси, ксерокси та інші прилади, без яких сучасна професійна діяльність неможлива, але всі вони мають високо небезпечні для людини фактори.

На всіх підприємствах, в установах, організаціях створюються безпечні і нешкідливі умови праці.

Небезпечні і шкідливі виробничі фактори виробничого середовища впливають на здоров'я і працездатність людини. Вони можуть бути причиною травм за певних умов.

Причини нещасних випадків поділяються на організаційні, технічні та санітарно-гігієнічні.

Організаційні причини:

– порушення режиму роботи і відпочинку;

– незадовільна організація, розташування і утримання робочих місць, проходів та проїздів;

– використання невідповідного інструмента, обладнання, пристроїв;

– незадовільна якість або відсутність індивідуальних захисних засобів;

– неправильна організація праці, нераціональний режим роботи;

– тривале вимушене одноманітне або ненормальне положення тіла чи окремих його частин та їх перенапруження та інші.

Технічні причини:

– недосконаліть технологічних процесів;

– недосконаліть обладнання і пристроїв;

– відсутність огорож і запобіжних пристроїв;

– незадовільний стан обладнання, інструмента і пристроїв.

Санітарно-гігієнічні причини:

– недостатність об'єму і площі виробничих приміщень;

– ненормальні метеорологічні умови (температура, вологість, швидкість руху і тиск повітря);

– освітлення не відповідає нормам;

– шкідливі випромінювання.

Аналіз виробничого травматизму проводиться з метою встановлення закономірностей виникнення травм на виробництві та розробки ефективних профілактичних заходів. Аналіз травматизму серед людей, що обслуговують ПЕОМ та працюють на ній, вказує на те, що в основному нещасні випадки трапляються під впливом небезпечних виробничих факторів при недотриманні інструкцій по безпеці праці.

Для аналізу виробничого травматизму застосовують чотири основних методи: статистичний, монографічний, економічний, метод фізичного і математичного моделювання.

За коефіцієнт частоти травматизму $K_ч$ береться кількість нещасних випадків, що припадають на тисячу працівників за певний період:

$$K_ч = \frac{H \cdot 1000}{T}, \qquad 4.1$$

де Н – число нещасних випадків за звітний період (для підприємства Н становить 2);

Т – середньооблікова кількість працівників за той же період (в нашому випадку Т = 45).

$$K_ч = \frac{2 \cdot 1000}{45} = 44.4, \qquad 4.2$$

Коефіцієнт важкості травматизму $K_т$ характеризує середня кількість днів непрацездатності, що припадають на один нещасний випадок:

$$K_т = \frac{D}{H}, \qquad 4.3$$

де D – сумарна кількість днів непрацездатності всіх потерпілих при нещасних випадках за звітний період (55 днів).

$$K_т = \frac{55}{2} = 27,5. \qquad 4.4$$

Зміна коефіцієнтів частоти, важкості і втрат протягом ряду періодів характеризує динаміку промислового травматизму й ефективність заходів щодо попередження травматизму.

Ці коефіцієнти дають можливість вивчати динаміку травматизму на підприємстві, порівнювати його з іншими підприємствами, аналізувати причини травматизму і розробляти заходи для попередження нещасних випадків. Але при цьому не вивчаються ґрунтовно-виробничі умови, при яких сталися нещасні випадки.

До витрат на охорону навколишнього природного середовища належать усі види витрат, спрямовані на запобігання, зменшення чи ліквідацію забруднення, інших видів шкідливого впливу господарської та іншої діяльності на навколишнє природне середовище, при наданні послуг чи використанні продукції.

У трудовому праві прийнято розуміти охорону праці в широкому сенсі як всю сукупність норм законодавства про працю, спрямованих на охорону і захист трудових прав працівників, їх положення в сфері праці.

Основне завдання охорони праці - профілактика та запобігання виробничого травматизму, професійних захворювань і мінімізація соціальних наслідків. Іншими словами, основне завдання охорони праці полягає в тому, щоб

забезпечити па кожному робочому місці соціально прийнятний ризик.

Такі напрями витрат, як економія ресурсів та енергозбереження, ураховуються тільки в тому випадку, коли вони спрямовані передусім на захист охорони навколишнього природного середовища, наприклад утилізацію відходів, яка здійснюється з метою охорони навколишнього природного середовища.

4.2 Аналіз умов праці і виявлення потенційно небезпечних та шкідливих виробничих факторів

Оператори ЕОМ стикаються з впливом таких фізичних і небезпечних психологічних факторів, як підвищена температура, відсутність або недостатність природного світла на робочому місці, електричний струм, статична електрика, розумове перенапруження, перенапруження зорових аналізаторів, монотонність роботи.

Основним джерелом проблем, пов'язаних з охороною здоров'я людей, що використовують у своїй роботі ПК, є дисплеї з електронно-променевими трубками. Вони є джерелами найбільш шкідливих випромінювань, що несприятливо впливають на здоров'я людини. Існує два типи випромінювань, які виникають при роботі монітора: статичне і електромагнітне. Перше виникає при опроміненні екрану потоком заряджених частинок. Неприємності, викликані ним, пов'язані з пилом, що накопичується на електростатичних заряджених екранах, яка летить на людину під час його роботи за дисплеєм. Результати медичних досліджень показали, що така наелектризована пил може викликати запалення шкіри.

При роботі з ПЕОМ можуть виникнути потенційно небезпечні та шкідливі фактори, вплив яких на організм людини може принести йому шкоди і призвести до травматизму.

ПЕОМ встановлюються і розміщуються відповідно до вимог технічних умов заводів-виготовлювачів. Вплив шкідливих електромагнітних

випромінювань зменшується за рахунок видалення їх джерел від оператора і установкою захисного екрана на монітор ПЕОМ. Вплив загазованості, запиленості і шкідливих парів, що виділяються ізоляцією установки усувається за рахунок правильного розміщення обладнання, що забезпечує хорошу природну вентиляцію.

Індекс ізоляції повітряного шуму між залом для глядачів і апаратної звукового забезпечення (при закритих оглядових вікнах) повинен бути не гірше 50 дБ. Стіни апаратної звукового забезпечення та стеля повинні оброблятися звукопоглинальними матеріалами з коефіцієнтом звукопоглинання не менше 0,6 в діапазоні частот 500 - 2000 Гц.

Праця працівників ОЦ повинна відносити до I – II класу за гігієнічними умовами праці: його тяжкість не повинна перевищувати оптимальних, а напруженість допустимих значень. Небезпечні фактори і їх допустимі значення наведені в таблиці 4.1

Таблиця 4.1 - Аналіз потенційних небезпек виробничих факторів

Джерело небезпек	**Характеристика потенційно-небезпечних виробничих факторів та їх допустимі значення**
1 Головний офіс: - електричний струм - електромагнітне поле	Фактичні (середні) дані вимірів: напруга 220-230 В, струм 25 А, частота 50 Гц. Можливість ураження електричним струмом. Діюче значення напруженості ЕМП: Е=30 А/м в діапазоні частот 50 Гц- 100кГц ГДР: Ен=50 В/м в діапазоні частот 60 Гц- 3МГц Н=30 А/м в діапазоні частот 50 Гц- 100кГц ГДР: Нн=5 А/м в діапазоні частот 60 Гц- 3МГц

2 ЕОМ: - ультрафіолетове випромінювання - рентгенівське випромінювання - електростатичне поле - ІЧ – випромінювання - видимий діапазон - яскравість	Фактичні дані вимірів: 0,1Вт/м^2. Допустима інтенсивність: 0,01 Вт/м^2 Фактичні дані вимірів: 15 мкР/год. ГДД: 75мкр.год. Фактичні дані: 15 кВ/м (0 Гц) Допустима напруженість поля 20-60 кВ/м. Фактичні дані вимірів: 0,06-7 Вт/м^2 (в діапазоні 700 нм-1мм). Допустима інтенсивність: 100Вт/м^2 Фактичні дані: 10,5 Вт/м^2 (в діапазоні 4-700 нм). Допустима інтенсивність: 10 Вт/м^2. Фактичні дані: 70 кд/м^2. Допустиме значення: 35 кд/м^2

Для збереження працездатності і попередження розвитку захворювань опорно-рухового апарату користувачів ПЕОМ організовано для них робочі місця, що відповідають вимогам ГОСТ 12.2.032-78.

При тривалій роботі за екраном дисплея в операторів відзначається виражена напруга зорового апарату з появою скарг на незадоволеність роботою, порушення сну, головну біль, дратівливість і хворобливі відчуття в очах, у попереку, руках і ін.

Згідно ГОСТ 12.2.032-78 конструкція робочого місця і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця користувача ПЕОМ дотримано наступні основні умови:

– достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення;

– оптимальне розміщення устаткування;

– природне і штучне освітлення.

4.3 Забезпечення нормальних умов праці

Одним з найважливіших завдань охорони праці є забезпечення таких умов

праці, які б вилучали можливість дії на працюючих різного роду небезпечних і шкідливих виробничих факторів.

Згідно зі статтею 153 Кодексу закону про працю, власник підприємства зобов'язаний забезпечити належне технічне об ладнання всіх робочих місць і створювати на них умови праці відповідно до нормативних актів з охорони праці.

Приміщення являє собою кімнату площею 21м^2 та об'ємом 56,7м^3. Кількість робочих місць у кімнаті – два. На одного працівника припадає 28,35м^3 об'єму приміщення та 10,5м^2 площі, що задовольняє вимогам "Державних санітарних правил та норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98", п.2 "Вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та ПЕОМ", згідно з якими площа на одне робоче місце має становити не менше ніж 6 м^2, а об'єм не менше ніж 20м^3.

Роботи відносяться до легкого характеру роботи категорії Іа, що виконуються сидячи й не потребують фізичного напруження.

При нормуванні умов для різних галузей промисловості виходять із загальних міжгалузевих норм

ГОСТ 12.1.005-88 "Загальні санітарно-гігієнічні вимоги до повітря робочої зони". На сьогодні основним нормативним документом, що визначає параметри мікроклімату виробничих приміщень є санітарні норми ДСН 3.3.6.042-99.

Оптимальні параметри мікроклімату у робочій зоні виробничого приміщення в теплий та холодний періоди року наведені в таблиці 4.2.

Таблиця 4.2 – Оптимальні параметри мікроклімату

Назва приміщення	**Категорія важкості фізичних робіт**	**Період Року**	**Температура ^oC**	**Відносна вологість %**	**Швидкість руху повітря м/с**
Приміщення з ЕОМ	Легка-1а	Холодний	22-24	40-60	0.1
	Легка-1а	Теплий	23-25	40-60	0.1

Кожна будівля чи окреме приміщення характеризуються, в залежності від властивостей їх зовнішніх захищень певною величиною максимальних тепловтрат. При розрахунку тепловтрат будівлі беруться до уваги мінімальні температурні показники для даної місцевості.

У приміщенні обладнана система опалення та кондиціювання повітря відповідно до СНиП 2.04.05-91. Використовується кондиціонер EWT G-091GS. Характеристики наведені в таблиці 4.3

Таблиця 4.3 – Характеристики кондиціонера EWT G-091GS

Характеристики по холоду	
Площа що обслуговується (м2)	25
Споживана потужність, кВт	1,01
Коефіцієнт ефективності, EER	2,61
Гарантований діапазон зовнішніх температур	+21 - +43
Характеристики по теплу	
Площа що обслуговується (м2)	27
Споживана потужність, кВт	1,07
Коефіцієнт ефективності, COP	2,61
Гарантований діапазон зовнішніх температур	-5 - +24
Технічні характеристики внутрішнього блоку	
Рівень шуму (мін-макс), дБ (А)	27-35
Вага, кг	8
Габарити (ВхШхГ), мм	255x730x174
Технічні характеристики зовнішнього блоку	
Рівень шуму (мін-макс), дБ (А)	50
Вага, кг	33
Габарити (ВхШхГ), мм	430x720x310
Загальні характеристики	
Джерело живлення, В / фаза / Гц	220-240/1/50
Максимальна довжина магістралі, м	10
Максимальний перепад висот, м	5
Фреон	R22

Для підтримання в приміщеннях нормальних параметрів повітряного середовища, яке відповідає санітарно-гігієнічним і технологічним вимогам, влаштовують вентиляцію.

Таблиця 4.4 – Характеристика системи вентиляції

Приміщення	Тип вентиляції	Вентиляційне обладнання	Кратність повітряного обміну 1/год.
Приміщення для експлуатації ЕОМ	Комбінована	Кондиціонер EWT G-091GS	3

Важливе місце в комплексі заходів з охорони праці працюючих з ПК займає утворення оптимального світлого середовища, тобто, раціональна організація природного і штучного освітлення приміщення і робочих місць.

При незадовільному освітленні знижується продуктивність праці користувачів ЕОМ, можлива поява короткозорості, швидка стомлюваність.

Система освітлення відповідає таким вимогам:

– освітленість на робочому місці відповідає характеру зорової роботи, який визначається трьома параметрами: об'єктом розрізнення - найменшим розміром об'єкта, що розглядається на моніторі персонального комп'ютера (ПК) та робочої станції; фоном, який характеризується коефіцієнтом відбиття; контрастом об'єкта і фону;

– забезпечено достатньо рівномірний розподіл яскравості на робочій поверхні монітора, а також в межах навколишнього простору;

– на робочій поверхні відсутні різкі тіні;

– у полі зору відсутні відблиски (підвищеної яскравості поверхонь, які світяться та викликають осліплення);

– величина освітленості є постійною під час роботи.

Джерела світла відносно робочого місця розташовують таким чином, щоб включити влучення в очі прямого світла. Захисний кут арматури в цих джерелах становить більше 30^0.

Розраховується площа світлопрорізів наступним чином:

- при боковому освітленні приміщень за формулою наведеною нижче:

$$S_{\text{в}} = \frac{D_{\text{н}}}{100\ m} \cdot \frac{K_{\text{з}} \eta_{\text{в}} K_{\text{буд}}}{\tau_o r_1} \qquad 4.5$$

- при верхньому освітленні приміщень за формулою наведеною нижче:

$$S_{\text{л}} = \frac{D_{\text{н}}}{100\,m} \cdot \frac{K_{\text{з}}\,\eta_{\text{л}}}{\tau_o r_1 K_{\text{л}}} \cdot S_n \qquad 4.6$$

Де:

$S_{\text{в}}$ і $S_{\text{л}}$ – площі світлових прорізів (в світлі) відповідно при боковому та верхньому освітленні;

S_n – площа підлоги приміщення, яка = 21 м^2;

$D_{\text{н}}$ – нормоване значення КПО, $D_{\text{н}} = 3$;

m – коефіцієнт світлового клімату світлопрорізу, який згідно даного регіону м. Івано-Франківськ m = 1,02

$К_{з}$- коефіцієнт запасу. згідно з ДБН 3.25-28-2018 для даної категорії приміщень $К_{з}$ = 1,2;

$\eta_{\text{л}}\ \eta_{\text{в}}$ – коефіцієнти, що враховують світлову активність вікон та ліхтарів, які $\eta_{\text{л}}$= 7,8 $\eta_{\text{в}}$ =8,5;

$K_{\text{л}}$ – коефіцієнти, що враховує тип ліхтарів, $K_{\text{л}}$=1,2;

$K_{\text{буд}}$ – коефіцієнт, що враховує затемнення вікон будівлями, що знаходяться напроти, $K_{\text{буд}}$ = 1,0;

$r_1\ r_2$ – коєфіцієнти, що враховується підвищення КПО за рахунок світла, відбитого від внутрішніх поверхонь приміщення, які r_1 =1,6 , $r_2 = 1{,}2$;

τ^0 - загальний коефіцієнт світлопропускання, який розраховується за формулою:

$$\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4 \cdot \tau_5, \qquad 4.7$$

τ^1 - коефіцієнт світлопропускання матеріалу вікон, τ^1=0,9;

τ^2 - коефіцієнт, що враховує витрату світла, для металопластикових τ^2 = 0,75;

τ^3 - коефіцієнт, що враховує витрату світла в несучих конструкціях, τ^3 = 1,2;

τ^4 - коефіцієнт, що враховує витрату світла в сонцезахисних пристроях, τ^4 = 1,0;

τ^5 - коефіцієнт, що враховує витрату світла в захисній сітці, τ^5 = 1.

Тоді:

- коефіцієнт світлопропускання розрахований:

$$\tau_o = 0{,}9 \cdot 0{,}75 \cdot 1{,}2 \cdot 1{,}0 \cdot 1 = 0{,}81$$

- розраховуємо при боковому освітленні приміщень:

$$S_в = \frac{3}{100 \cdot 1{,}02} \cdot \frac{1{,}2 \cdot 8{,}5 \cdot 1}{0{,}81 \cdot 1{,}6} = 0{,}03 \cdot 7{,}87 = 2{,}36$$

- розраховуємо при верхньому освітленні:

$$S_л = \frac{3}{100 \cdot 1{,}02} \cdot \frac{1{,}2 \cdot 7{,}8}{0{,}81 \cdot 1{,}6 \cdot 1{,}2} \cdot 21 = 0{,}03 \cdot 6 \cdot 21 = 3{,}78$$

Відомо, що вісімдесят відсотків інформації зовнішнього світу людина отримує через очі. Якість інформації залежить від освітлення. Тому правильно організована система освітлення має велике значення в зниженні виробничого травматизму, створює нормальні умови для роботи органів зору, підвищує працездатність організму і відповідно, продуктивність праці: при зорових роботах середньої важкості на 5…6%, при важкій зоровій роботі на 15%, а при роботі в межах зорового сприйняття – на 40%.

Відповідно площі світлових прорізів (в світлі)при боковому освітленні $S_в = 2{,}36$ та верхньому освітленні $S_л = 3{,}78$

Скористаємося методом використання світлового потоку. Для визначення потрібної кількості світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F = \frac{E \cdot K \cdot S \cdot Z}{\eta}, \qquad 4.8$$

де F – світловий потік, що розраховується, лм;

E – нормована мінімальна освітленість, Лк; E = 300 лк;

S – площа освітлюваного приміщення (у нашому випадку S=21м^2);

Z – відношення середньої освітленості до мінімальної (Z =1,1);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації;

η – коефіцієнт використання світлового потоку освітлювальної системи, який залежить від розподілу сили світла світильників, показника приміщення та коефіцієнтів відбиття стелі, стін і робочої поверхні.

Показник приміщення визначається за формулою:

$$i = \frac{a \cdot b}{h(a+b)}, \qquad 4.9$$

де а, в - ширина і довжина приміщення відповідно, а = 3,5м, в = 6м;

h - висота розташування світильників, h = 2,7м. Підставивши значення отримаємо:

$$i = \frac{3,5 \cdot 6}{2,7(3,5+6)} \approx 0,8$$

Для світильників типу ОДО при і = 0,8, $\rho_{стелі}$ = 50%, $\rho_{стін}$ = 30%, $\rho^{роб.і}$ =30%, ŋ = 0,36. Світловий потік буде дорівнювати:

$$Ф = \frac{300 \cdot 21 \cdot 1,5 \cdot 1,1}{0,36} = 28875 лм.$$

Визначаємо кількість світильників за формулою:

$$n = \frac{Ф}{Ф_л}, \qquad 4.10$$

де $Ф_л$ - світловий потік однієї лампи.

Для ламп Philips TLD світловий потік становить $Ф_л$ = 2800 лм.

$$n = \frac{28875}{2800} \approx 10.$$

В приміщенні кожен світильник комплектується двома лампами. Тобто необхідно використовувати 5 світильників із 10 працюючими лампами в них. Загальну робочу освітленість визначаємо за формулою:

$$E = \frac{Ф_л \cdot n \cdot \eta}{S \cdot K \cdot Z} = \frac{2800 \cdot 10 \cdot 0,36}{21 \cdot 1,5 \cdot 1,1} = 300 лк.$$

Дане розрахункове значення знаходиться в межах 300-400лк, встановлених ДБН 3.25-28-2018.

4.4 Забезпечення безпеки технологічних процесів, монтажу, пусконалагоджувальних, ремонтних робіт та експлуатації обладнання, приладів та пристроїв.

При проектуванні систем електропостачання, при монтажі силового електроустаткування і електричного освітлення в будівлях і приміщеннях для ЕОМ необхідно дотримуватися вимог нормативно-технічної документації (ПУЕ, ПТЕ, ПТБ і ін.).

ЕОМ є однофазним споживачем електроенергії, що живиться від трифазної чотирьох проводної мережі з глухо заземленою нейтраллю змінного струму частотою 50 Гц. Електробезпека ЕОМ забезпечується комплексом конструктивних, схемноконструктивних і експлуатаційних засобів і способів захисту [46].

Експлуатаційні заходи захисту ґрунтуються на дотриманні правил техніки безпеки при роботі з високою напругою і наступних запобіжних засобів:

– не підключати і не відключати роз'єми кабелів при включеній напрузі мережі;

– технічне обслуговування і ремонтні роботи проводити тільки при вимкненому живленні мережі.

Конструктивні заходи безпеки забезпечують захист від випадкового дотику до струмопровідних частин за допомогою захисних оболонок і ізоляції струмопровідних елементів. Ступінь захисту оболонки ЕОМ повинен відповідати класу пожежонебезпечної зони приміщення П-IIа [47] і бути не нижчим IP-44. В мережах напругою до 1000В з глухо заземленною нейтрально як схемно-конструктивною мірою захисту застосовується занулення.

Експлуатаційні заходи.

Необхідно дотримувати правила техніки безпеки при роботі з високою напругою і наступні запобіжні засоби:

– монтаж, обслуговування, ремонт і наладка ЕОМ, заміна деталей, пристосувань і блоків повинна здійснюватися тільки при повному виключенні живлення;

– в приміщеннях, де експлуатуються більше 5 комп'ютерів на видному і доступному місці встановлюється аварійний і резервний вимикач для повного відключення електроживлення;

– заземлені конструкції в приміщенні повинні бути надійно захищений діелектричними щитками або сітками від випадкового дотику.

Працівник, що поступає на роботу, обов'язково проходить вступний і первинний інструктаж по техніці безпеки в цілях профілактики нещасних випадків, а також знайомиться з інструктажем по дотриманню заходів техніки безпеки при роботі з ПЕВМ.

При організації праці, пов'язаної з використанням ПК, для збереження здоров'я працюючих, запобігання професійним захворюванням І підтримки працездатності передбачаються внутрішньо змінні регламентовані перерви для відпочинку.

Внутрішньозмінні режими праці й відпочинку містять додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення й зниження працездатності.

При виконанні робіт, що належать до різних видів трудової діяльності, за основну роботу з ПК слід вважати таку, що займає не менше 50% робочого часу. Впродовж робочої зміни мають передбачатися:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку й особистих потреб (згідно із трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

За характером трудової діяльності розрізняють три професійні групи, згідно з діючим класифікатором професій:

– розробники програм інженери-програмісти виконують роботу переважно з відеотерміналом та документацією при необхідності інтенсивного обміну Інформацією з ЕОМ і високою частотою прийняття рішень. Робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті верхніх кінцівок. Робота виконується в режимі діалогу з ПК у вільному темпі з періодичним пошуком помилок в умовах дефіциту часу;

– оператори електронно-обчислювальних машин виконують роботу, пов'язану з обліком інформації, одержаної із ВДТ за попереднім запитом, або тієї, що надходить з нього, супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи й характеризується напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у вільному темпі;

– оператор комп'ютерного набору виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран дисплея, з введенням даних з високою швидкістю. Робота характеризується як фізична праця з підвищеним навантаженням на кисті верхніх кінцівок на фоні загальної гіподинамії з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням.

Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ПК при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм із застосуванням ПК слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ПК;

– для операторів із застосуванням ПК слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ПК.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ПК не повинна перевищувати 4 години.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин робота аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин.

Для зниження нервово-емоційного напруження, втомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які наведені у Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСаяПІН 3.3.2.018.

4.5 Пожежна безпека та безпека у надзвичайних ситуаціях

Пожежна безпека об’єкта, відповідно до ГОСТ 12.1.004-91 забезпечується системою запобігання пожежі, системою протипожежного захисту і системою організаційно-технічних заходів.

Пожежну безпеку забезпечують такі основні компоненти виробництва:

– технічна система, яка передбачає надійність обладнання, використання безпечних технологій, проектні рішення, впровадження систем виявлення та гасіння пожеж, розміщення обладнання тощо;

– персонал, його підготовка, забезпечення регламентами та правилами роботи;

– система управління.

Системи пожежної безпеки спрямовані на:

– комплексний аналіз із метою створити ефективні засоби попередження пожежі шляхом нейтралізації дії сприяючих їй обставин;

– вивчення засобів і методів локалізації та гасіння пожеж;

– запобігання виникненню пожежі;

– пожежну безпеку людей та матеріальних цінностей.

Електрична енергія при визначених умовах легко переходить у теплову, і це може викликати пожежі і вибухи. Пожежна небезпека електрообладнання, електронних приладів, радіоелектронної апаратури, апаратури управління, електроприймачів пов'язана з використанням горючих матеріалів: гуми, пластмас, лаків, олій.

Джерелами займання можуть бути електричні іскри, дуги, коротке замикання, струмові перевантаження, перегріті опірні поверхні, несправність обладнання. Окиснювачем звичайно є кисень. Але потужність і тривалість дії цих джерел займання порівняно малі, тому горіння, зазвичай, не розвивається.

Таблиця 4.5 – Засоби вогнегасіння

Найменування приміщення	**Тип приміщення(категорія по СніП)**	**Площа, яка захищається м3**	**Типи первинних засобів пожежегасіння**			
			Тип вогнегасників	**Марка вогнегасників**	**Інші види**	**Кількість, шт.**
Головний офіс	В	21	Вуглекислотний	ОУ – 8х	Кошма 2х1, волок	1

Пожежні норми щодо роботи з паливними речовинами регламентовані ГОСТ 12.1.004 – 85, ГОСТ 12.1.018 – 86, СНіП 2.01.02 – 85. Засобами гасіння (таблиця 4.5) є: піна, вуглекислота, порошкові суміші. Ці засоби знаходяться в усіх приміщеннях в зручних для доступу місцях.

4. 6 Висновки до розділу 4

У цьому розділі дипломної роботи було викладено законодавче та

нормативно-правове забезпечення охорони праці, організацію роботи з охорони праці на підприємстві, значення охорони праці і навколишнього середовища в забезпеченні безпечних та здорових умов праці.

Проведено аналіз умов праці та щодо виявлення потенційно небезпечних та шкідливих виробничих факторів.

Розглянуто забезпечення нормальних умов праці. Проведено розрахунок природнього та штучного освітлення. Ознайомлено з забезпеченням безпеки технологічних процесів, монтажу та експлуатації обладнання пристроїв, також проведено огляд з пожежної безпеки та безпеки у надзвичайних ситуаціях.

Було визначено, що на стан охорони праці на підприємстві відповідає вимогам законодавчих та нормативно-правових актів, а розрахунок за методом відносної площі світлових прорізів засвідчив, що в заданому приміщенні можна виконувати зорову роботу.

ВИСНОВКИ

В дипломному проекті було розроблено Система управління рестораном - це програмне рішення, розроблене для оптимізації та автоматизації різних аспектів роботи ресторану, включаючи управління замовленнями, меню та інвентарем. Розроблена за допомогою Visual Studio та C++/CLI, система надає інтуїтивний користувацький інтерфейс для ресторанного персоналу, щоб ефективно виконувати завдання, такі як прийом замовлень, оновлення меню та створення рахунків.

Ключові особливості даного проекту:

Форма вітання: Після запуску програми користувачі вітаються з формою вітання, де є можливість переходу до різних розділів системи, таких як прийом замовлень чи управління даними.

Форма оформлення замовлення: Ця форма керує користувачами під час процесу оформлення замовлень. Користувачі можуть вибрати існуючі замовлення зі списку або створити нові. Якщо будь-які обов'язкові поля залишаються порожніми, система попереджає користувача про їх заповнення перед продовженням. Після успішного оформлення замовлення система відображає повідомлення підтвердження разом з деталями замовлення, такими як дата, номер столика, обрані позиції та загальна вартість.

Форма управління даними: Форма управління даними дозволяє користувачам переглядати, додавати, оновлювати або видаляти дані, пов'язані з пунктами меню, замовленнями та інвентарем. Вона включає функціонал для фільтрації даних, пагінації для великих наборів даних та опції сортування для кращої організації.

Обробка винятків: Протягом всього застосунку реалізовані надійні механізми обробки винятків для забезпечення плавної роботи та надання користувачам інформативних повідомлень про помилки у разі неправильних введених даних або проблем з підключенням до бази даних.

Інтеграція з базою даних: Система безперешкодно інтегрується з базою даних для ефективного зберігання та отримання даних. Виконуються SQL-запити для виконання операцій CRUD (створення, читання, оновлення, видалення) з таблицями бази даних, забезпечуючи цілісність та послідовність даних.

Інтерфейс користувача зручний для використання: Графічний інтерфейс користувача розроблений з урахуванням зручності користувача, з інтуїтивним навігаційним меню, чітким позначенням елементів управління та інтерактивними елементами для зручної взаємодії.

Звітність та аналітика: Надається базовий функціонал для генерації звітів для отримання інформації про тенденції продажів, популярні позиції меню та аналіз доходів. Користувачі можуть генерувати звіти на основі різних критеріїв, що допомагає керівникам ресторану приймати обґрунтовані рішення.

Висновок: Система управління рестораном пропонує комплексне рішення для ефективного управління ресторанною діяльністю. З інтуїтивним інтерфейсом, надійною інтеграцією з базою даних та розширеним набором функцій вона дозволяє персоналу ресторану оптимізувати робочі процеси, покращувати обслуговування клієнтів та оптимізувати загальну ефективність бізнесу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Статті про програмування для Windows [Електронний ресурс] Режим доступу: http://flashbot.ru/android-dev

2. Вендров А.М. Практикум із проектування програмного забезпечення економічних інформаційних систем: Навч. допомога. – М.: Фінанси та статистика, 2004. – 192 с.

3. Н. Єлманова, С. Трепалін, А. Тенцлер. Delphi та технологія COM – СПБ.:, 2013 – 698 с.

4. Роберт Вієйра. Програмування бази даних Microsoft SQL Server 2005. Базовий курс = Beginning Microsoft SQL Server 2005 Programming. – М.: «Діалектика», 2017. – С. 832. – ISBN 0-7645-8433-2 Посилання на літру.

5. Роберт Шелдон, Джоффрей Моє MySQL: базовий курс = Beginning MySQL. – М.: «Діалектика», 2017. – С. 880. – ISBN 0-7645-7950-9 Посилання на літру.

6. С.В. Маклаків. Створення інформаційних систем із ALLFusion Modelling Suite. М., 2013.

7. С.В. Маклаків. ERwin та Bpwin. CASE-засоби розробки інформаційних систем. М., 2019.

8. Вендров А.М. Один із підходів до вибору засобів проектування баз даних та додатків. «СУБД», 2015 №3.

9. Уенді Боггс, Майкл Боггс. UML та Rational Rose. 2014. - С. 346.

10. Ільїн В.В. Designer Реінжиніринг бізнес-процесів за допомогою ARIS Практика реального бізнесу. "Вільямс", 2018. - С. 456.

11. Олексій Ковязін, Сергій Востриков «Світ InterBase. 3-тє видання» 2015. С. 496.

12. Гусєва О.А. Visual Basic. "Кін". 2018. – С. 390.

13. Б. Страуструп. Мова програмування C++ = The C++ Programming Language/Пер. з англ. - 3-тє вид. - СПб.; М.: Невський діалект – Біном, 2019. – 991 с.

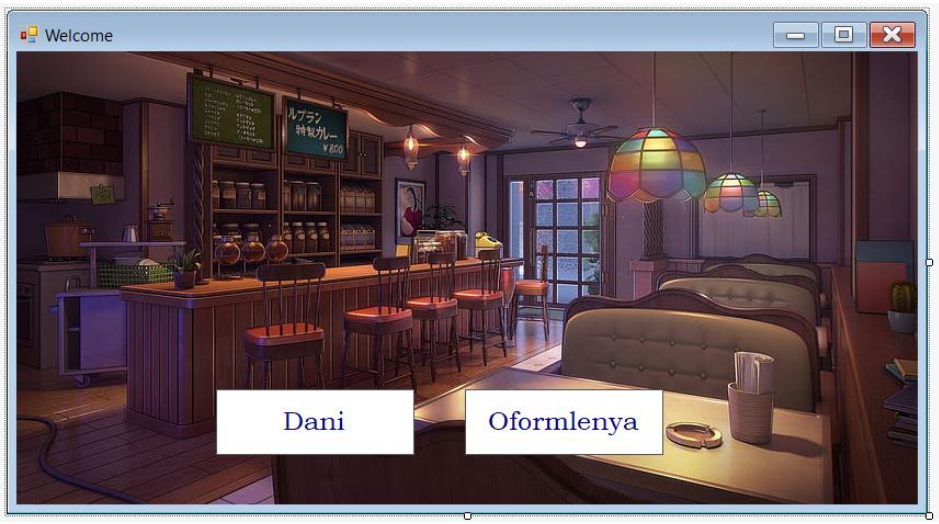
ДОДАТКИ

Додаток А

Код класу DANI:

```
      private: System::Void Dani_Load(System::Object^  sender,
System::EventArgs^  e) {
                          auto OleDbConnection1 = gcnew
OleDb::OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\\restApp.mdb");
                       OleDbConnection1->Open();
                       auto comm1 = gcnew OleDb::OleDbCommand("Select *
From [products]", OleDbConnection1);
                       auto ExecuteReader1 = comm1-> ExecuteReader();
                       auto ProductsTable = gcnew DataTable();
                       ProductsTable->Columns->Add(ExecuteReader1-
>GetName(0));
                              ProductsTable->Columns->Add(ExecuteReader1-
>GetName(1));
                              ProductsTable->Columns->Add(ExecuteReader1-
>GetName(2));
                              ProductsTable->Columns->Add(ExecuteReader1-
>GetName(3));
                              ProductsTable->Columns->Add(ExecuteReader1-
>GetName(4));
                              ProductsTable->Columns->Add(ExecuteReader1-
>GetName(5));
                       while (ExecuteReader1->Read() == true)
                           ProductsTable->Rows->Add(ExecuteReader1-
>GetValue(0),
                           ExecuteReader1->GetValue(1), ExecuteReader1-
>GetValue(2),
                                     ExecuteReader1->GetValue(3),
ExecuteReader1->GetValue(4),
                                     ExecuteReader1->GetValue(5));
                       ExecuteReader1->Close();
                              OleDbConnection1->Close();
                              ProductsDataGridView->DataSource =
ProductsTable;
                              OleDbConnection1->Open();
                       auto comm2 = gcnew OleDb::OleDbCommand("Select *
From [dish]", OleDbConnection1);
                       auto ExecuteReader2 = comm2-> ExecuteReader();
                       auto DishTable = gcnew DataTable();
                       DishTable->Columns->Add(ExecuteReader2->GetName(0));
                              DishTable->Columns->Add(ExecuteReader2-
>GetName(1));
                              DishTable->Columns->Add(ExecuteReader2-
>GetName(2));
                              DishTable->Columns->Add(ExecuteReader2-
>GetName(3));
                              DishTable->Columns->Add(ExecuteReader2-
>GetName(4));
                              DishTable->Columns->Add(ExecuteReader2-
>GetName(5));
                       while (ExecuteReader2->Read() == true)
                           DishTable->Rows->Add(ExecuteReader2-
>GetValue(0),
                           ExecuteReader2->GetValue(1), ExecuteReader2-
>GetValue(2),
                                     ExecuteReader2->GetValue(3),
ExecuteReader2->GetValue(4),
                                     ExecuteReader2->GetValue(5));
                       ExecuteReader2->Close();
                              OleDbConnection1->Close();
```

```
                        DishDataGridView->DataSource = DishTable;
                        OleDbConnection1->Open();
                    auto comm3 = gcnew OleDb::OleDbCommand("Select *
From [order]", OleDbConnection1);
                    auto ExecuteReader3 = comm3-> ExecuteReader();
                    auto OrderTable = gcnew DataTable();
                    OrderTable->Columns->Add(ExecuteReader3-
>GetName(0));
                        OrderTable->Columns->Add(ExecuteReader3-
>GetName(1));
                        OrderTable->Columns->Add(ExecuteReader3-
>GetName(2));
                        OrderTable->Columns->Add(ExecuteReader3-
>GetName(3));
                    while (ExecuteReader3->Read() == true)
                        OrderTable->Rows->Add(ExecuteReader3-
>GetValue(0),
                        ExecuteReader3->GetValue(1), ExecuteReader3-
>GetValue(2),
                                ExecuteReader3->GetValue(3));
                    ExecuteReader3->Close();
                        OleDbConnection1->Close();
                        OrderDataGridView->DataSource = OrderTable;
                      }
```



Ілюстрація форми Welcome в режимі конструктура

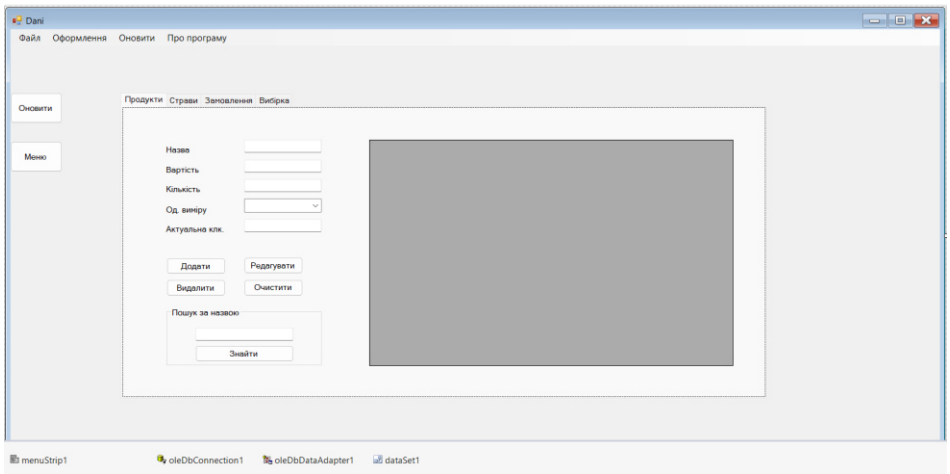


Рисунок 3.3 – Ілюстрація форми Dani в режимі конструктура

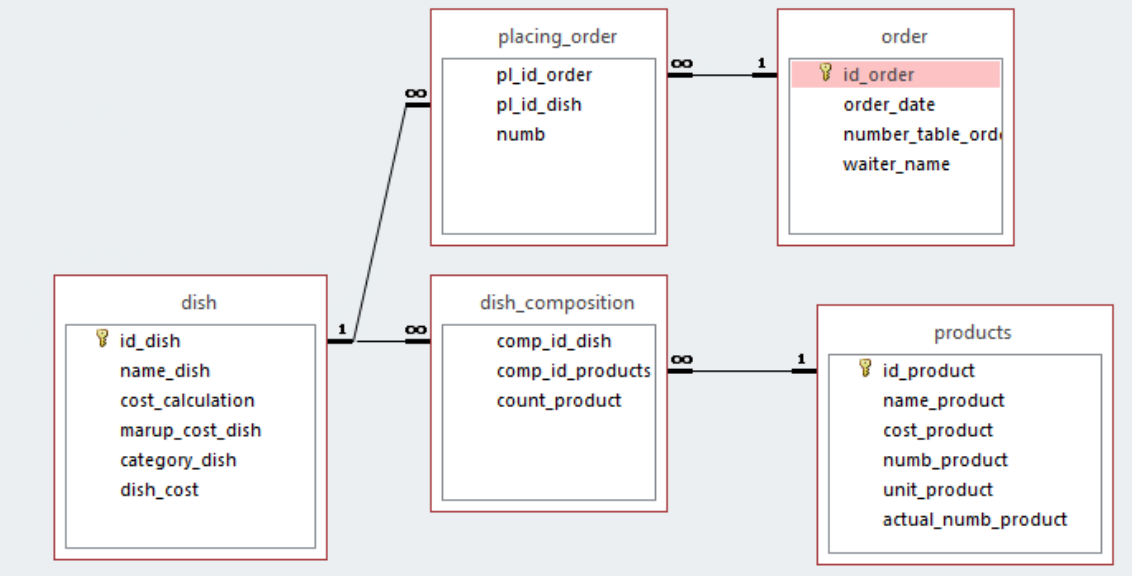
Додаток Б

Код класу DANI_LOAD:

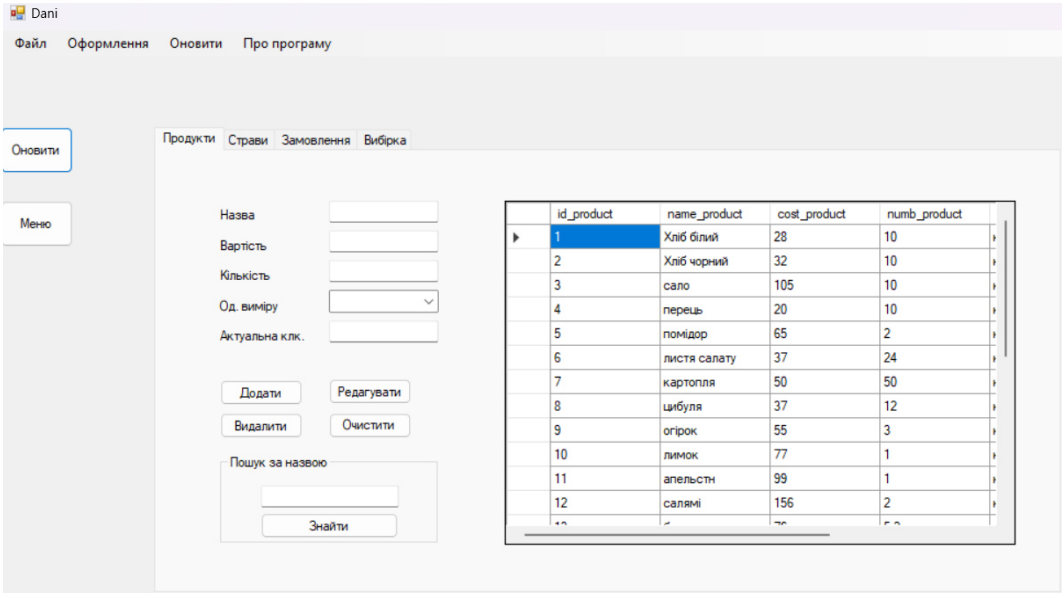
```
private: System::Void Dani Load(System::Object^  sender, System::EventArgs^  e) {
                  auto OleDbConnection1 = gcnew
OleDb::OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\\restApp.mdb");
                OleDbConnection1->Open();
                auto comm1 = gcnew OleDb::OleDbCommand("Select * From
[products]", OleDbConnection1);
                auto ExecuteReader1 = comm1-> ExecuteReader();
                auto ProductsTable = gcnew DataTable();
                ProductsTable->Columns->Add(ExecuteReader1->GetName(0));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(1));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(2));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(3));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(4));
                        ProductsTable->Columns->Add(ExecuteReader1->GetName(5));
                while (ExecuteReader1->Read() == true)
                    ProductsTable->Rows->Add(ExecuteReader1->GetValue(0),
                    ExecuteReader1->GetValue(1), ExecuteReader1->GetValue(2),
                                ExecuteReader1->GetValue(3), ExecuteReader1-
>GetValue(4),
                                ExecuteReader1->GetValue(5));
                ExecuteReader1->Close();
                        OleDbConnection1->Close();
                        ProductsDataGridView->DataSource = ProductsTable;
                        OleDbConnection1->Open();
                auto comm2 = gcnew OleDb::OleDbCommand("Select * From [dish]",
OleDbConnection1);
                auto ExecuteReader2 = comm2-> ExecuteReader();
                auto DishTable = gcnew DataTable();
                DishTable->Columns->Add(ExecuteReader2->GetName(0));
                        DishTable->Columns->Add(ExecuteReader2->GetName(1));
                        DishTable->Columns->Add(ExecuteReader2->GetName(2));
                        DishTable->Columns->Add(ExecuteReader2->GetName(3));
                        DishTable->Columns->Add(ExecuteReader2->GetName(4));
                        DishTable->Columns->Add(ExecuteReader2->GetName(5));
                while (ExecuteReader2->Read() == true)
                    DishTable->Rows->Add(ExecuteReader2->GetValue(0),
                    ExecuteReader2->GetValue(1), ExecuteReader2->GetValue(2),
                                ExecuteReader2->GetValue(3), ExecuteReader2-
>GetValue(4),
                                ExecuteReader2->GetValue(5));
                ExecuteReader2->Close();
                        OleDbConnection1->Close();
                        DishDataGridView->DataSource = DishTable;

                        ////////////////////////////////

                        OleDbConnection1->Open();
                auto comm3 = gcnew OleDb::OleDbCommand("Select * From [order]",
OleDbConnection1);
                auto ExecuteReader3 = comm3-> ExecuteReader();
                auto OrderTable = gcnew DataTable();
                OrderTable->Columns->Add(ExecuteReader3->GetName(0));
                        OrderTable->Columns->Add(ExecuteReader3->GetName(1));
                        OrderTable->Columns->Add(ExecuteReader3->GetName(2));
                        OrderTable->Columns->Add(ExecuteReader3->GetName(3));
                while (ExecuteReader3->Read() == true)
                    OrderTable->Rows->Add(ExecuteReader3->GetValue(0),
                    ExecuteReader3->GetValue(1), ExecuteReader3->GetValue(2),
                                ExecuteReader3->GetValue(3));
```

```
ExecuteReader3->Close();
        OleDbConnection1->Close();
        OrderDataGridView->DataSource = OrderTable;


    }
```

Вид схеми даних створеної БД представлено на рисунку 3.1.



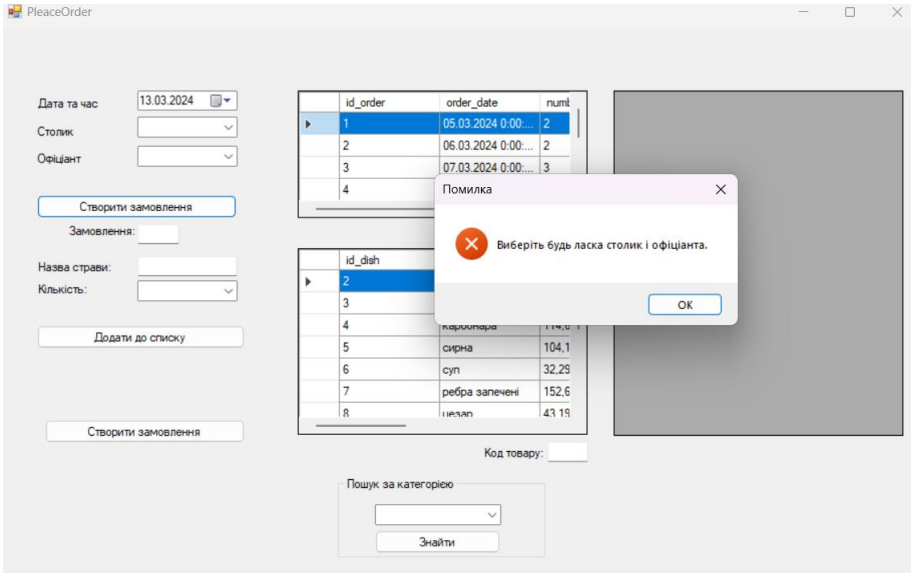
Ілюстрація того, як активності утворюють відгалуження у макеті та містять активності



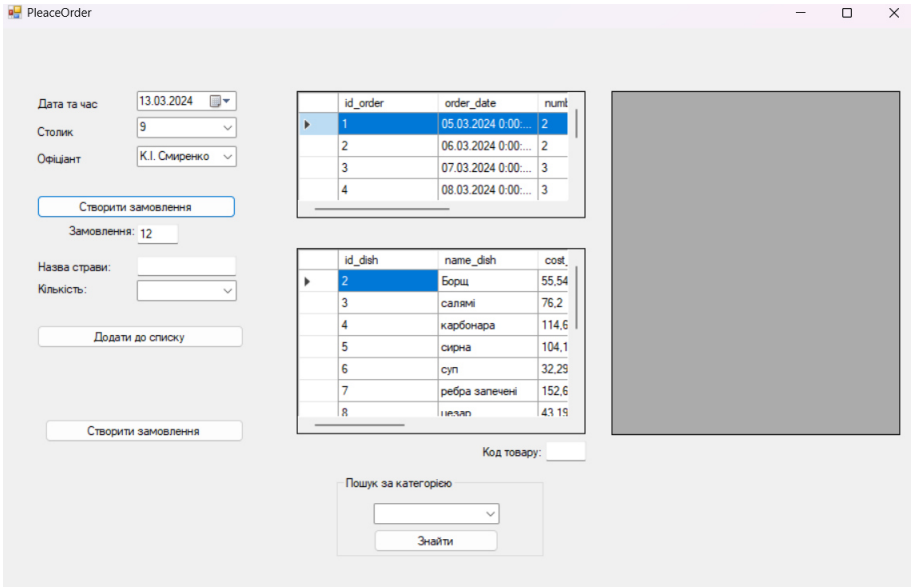
Ілюстрація відображення даних на вкладці Продукти

Додаток В

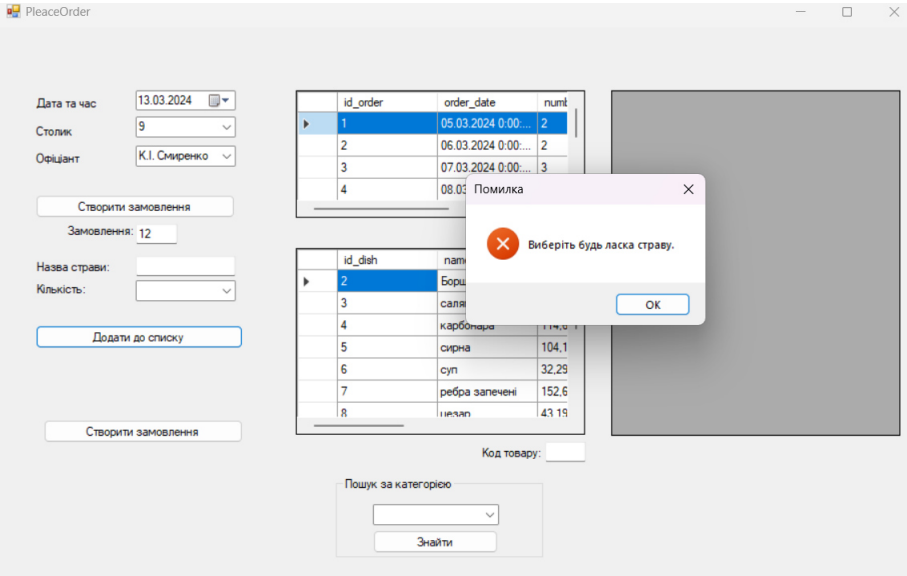
Валідаця даних



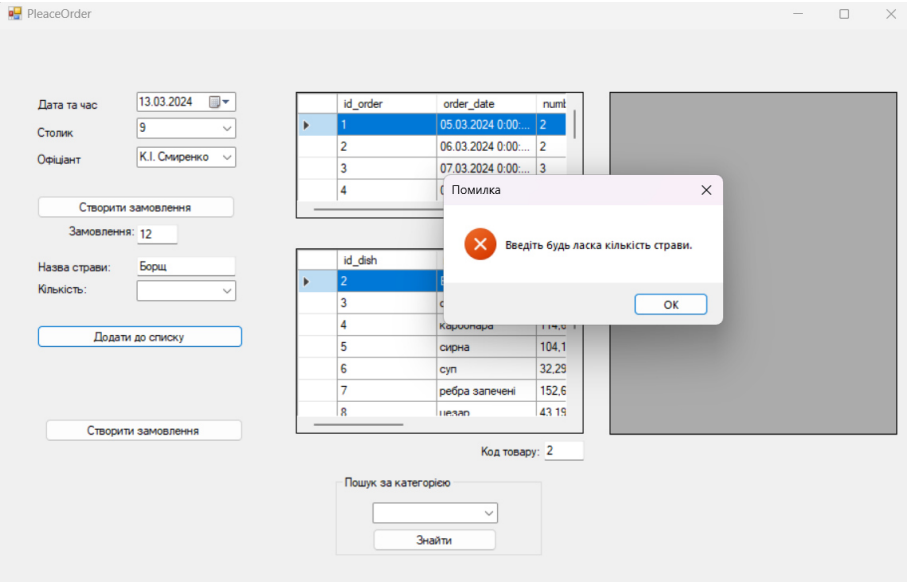
Ілюстрація помилки про неправильне оформлення замовлення



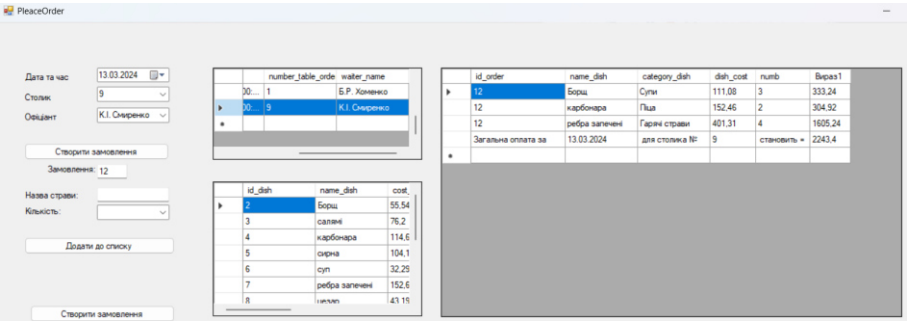
Ілюстрація успішно створенго замовлення



Ілюстрація про помилку додавання страви до замовлення без страви



Ілюстрація результа помилки про відсутність кількості



люстрація результат успішного оформлення замовлення

БІБЛІОГРАФІЧНА ДОВІДКА

Індивідуальна тема дипломної роботи: *«*Створення інформаційної системи обліку та обігу продукції в готельно ресторанному бізнесі*»*

Обсяг пояснювальної записки: 85 аркушів

Кількість рисунків 21

Кількість таблиць 5