Державний вищий навчальний заклад

«Чернівецький політехнічний коледж»

Відділення інформаційних технологій

Циклова комісія інженерії програмного забезпечення

**Пояснювальна записка**

до дипломного проекту

молодшого спеціаліста

на тему: Електронне меню для закладів громадського харчування

Виконав: студент IV курсу, групи 541

спеціальність 5.05010301

«Розробка програмного забезпечення»

Гудан Ю.В.

Керівник Серьогіна І.В.

Рецензент «невідомо»

Чернівці – 2018 року**зміст**

[ВСТУП 3](#_Toc516699563)

[РОЗДІЛ 1. ПЕРЕДПРОЕКТНЕ ДОСЛІДЖЕННЯ 5](#_Toc516699564)

[1.1 Аналіз предметної області 5](#_Toc516699565)

[1.2 Аналіз існуючого програмного забезпечення 6](#_Toc516699566)

[1.3 Аналіз сучасного стану та перспективи розвитку 7](#_Toc516699567)

[1.4 Постановка задачі 9](#_Toc516699568)

[1.5 Маркетингове дослідження 10](#_Toc516699569)

[РОЗДІЛ 2. ТЕХНІЧНИЙ ПРОЕКТ 11](#_Toc516699570)

[2.1 Технічне завдання 11](#_Toc516699571)

[2.2 Моделювання програмного забезпечення 14](#_Toc516699572)

[2.3 Моделювання даних 17](#_Toc516699573)

[2.4 Проектування інтерфейсу програмного забезпечення 21](#_Toc516699574)

[РОЗДІЛ 3. робочий проект 27](#_Toc516699575)

[3.1 Засоби розробки 27](#_Toc516699576)

[3.2 Інструкція програміста 28](#_Toc516699577)

[3.3 Інструкція користувача 33](#_Toc516699578)

[3.4 Тестування програмного забезпечення 39](#_Toc516699579)

[ВИСНОВКИ 42](#_Toc516699580)

[перелік джерел 43](#_Toc516699581)

[Додатки 44](#_Toc516699582)

# ВСТУП

Актуальність теми розробки, в даний час дуже популярні заклади громадського харчування, і в таких закладах часто виникає проблеми з обслуговуванням. Дане тема було обрана для зручного, швидкого та ефективного обслуговування клієнтів закладів громадського харчування.

Мета дипломного проектування розробити програмне забезпечення електронне меню для закладів громадського харчування.

Завдання дипломного проектування є розробки програмного забезпечення для автоматизації заданої предметної області.

Об’єктом дослідження є заклади громадського харчування .

Предметом дослідження є методології та засоби реалізації функцій адміністратора та офіціанта, клієнт. В залежності від рівня доступу,для клієнта надається доступ для перегляду меню в електронному вигляді, для офіціанта є доступ: обліку та редагування замовлення, перегляд та вибір інформації меню в електронному вигляді, для адміністратора надається повний доступ до програмного забезпечення.

Практичне значення отриманих результатів можна застосувати в закладах громадського харчування, так як даний програмний продукт спрощує та автоматизує роботу в закладі.

Використане програмне забезпечення, буде розроблено за допомогою середовище Embarcadero RAD Studio 10.2 Architect 2017 С++ Builder, та Microsoft SQLEXPRESS Server в зв’язку з відповідним функціоналом для створення даного програмного забезпечення.

Дипломний проект складається із програмного забезпечення та пояснювальної записки до нього.

У першому розділі пояснювальної записки до дипломного проекту представлено аналіз предметної області, проаналізовано програмні продукти аналогічного призначення, виконано постановку задачі, визначено основні функціональні вимоги до розробки та проведено маркетингове дослідження інноваційності розробки.

У другому розділі описується процес проектування програмного забезпечення, зокрема процес проектування інтерфейсу, наводяться концептуальна модель додатку, логічна та фізична моделі бази даних.

Третій розділ містить опис засобів розробки програмного забезпечення, опис пакетів та їх класів, а також загальні принципи роботи з програмним забезпеченням та результати тестування.

# РОЗДІЛ 1. ПЕРЕДПРОЕКТНЕ ДОСЛІДЖЕННЯ

## 1.1 Аналіз предметної області

Ресторанне господарство – це вид економічної діяльності суб'єктів господарської діяльності з надання послуг для задоволення потреб споживачів у харчуванні з організацією дозвілля або без нього. Суб’єкти господарювання провадять діяльність у ресторанному господарстві через підприємства (заклади) ресторанного господарства.

Заклад ресторанного господарства – підприємство, призначене для виробництва кулінарної продукції, борошняних, кондитерських і булочних виробів, їх реалізації та організації їх споживання.

Однією із основних вимог сучасної індустрії ресторанного господарства є швидке і якісне обслуговування споживачів. Заклади ресторанного господарства, що займаються торгівлею конкурентоспроможною продукцією, необхідної широкому споживачеві, стискаються з проблемою збільшення кількості клієнтів. А так, як при ручному веденні документації потрібного багато часу, трудовитрат і матеріалів, необхідна автоматизація процесів заповнення, поновлення, зберігання і обробки документації – такі можливості реалізовані в системі інтерактивного електронного меню.

Цей новітній засіб комунікації є потужним інструментом, що дозволяє надавати клієнтам високий рівень обслуговування, а також більш ефективно керувати закладом. Якісне обслуговування в ресторані – це суть послуг ресторану й умова його успішності. Асортимент, смакові якості страв, ціни, дизайн, імідж, місце розташування важливі, але вони ніколи не компенсують в очах клієнта неуважного і недбалого ставленням з боку персоналу. З іншого боку, ресторан, що зумів надати клієнтам відмінний сервіс, одержує найсильнішу конкурентну перевагу.

Електронне меню повністю замінює традиційне меню і пропонує цілий ряд додаткових функцій. Воно є не тільки каталогом продукції, яку пропонує заклад, але і прискорює процес обслуговування, зменшує «людський фактор» (помилки) під час приймання замовлення. Меню представлене у вигляді візуально дуже схожому на паперовий прототип, що сприяє легкому сприйняттю інтерфейсу.

На практиці експлуатації такого програмного забезпечення приведе до вирішення таких завдань як: фізичний перерозподіл обчислень і даних, забезпечення паралелізму обчислень, реплікація бази даних, забезпечення безпеки доступу до програмного забезпечення, оптимізація балансування навантаження інформаційної системи, стійкість до збоїв і т.п. З цим програмним забезпеченням робота в сфері ресторанного господарства прискориться в багато раз.

## 1.2 Аналіз існуючого програмного забезпечення

Існують багато програмних забезпечень для закладів громадського харчування, які спрощують обслуговування, швидкість замовлення тощо.

Перший аналог «Quick Resto» – це системи для залучення гостей, інструментів обліку та аналітики для збільшення прибутку, а також універсальна POS-система для обслуговування відвідувачів. За допомогою неї можна контролювати роботу і спостерігати за ростом бізнесу.

Інтерфейс програмного забезпечення зображено на рисунку 1.



Рисунок 1 – Інтерфейс ПЗ «Quick Resto»

Дане програмне забезпечення має можливості: ведення обліку страв, стеження за прибутком, повний контроль над програмним забезпеченням, формування звітності за різними вимогами, налаштування повідомлень для постійного стеження за роботою в закладі, прибутком.

Другий аналог «Ultra Ресторан» – це програма для ресторанів, яка впорядкує облік і дозволить поліпшити систему обслуговування клієнтів.

Інтерфейс програмного забезпечення зображено на рисунку 2.



Рисунок 2 – Інтерфейс ПЗ «Ultra Ресторан»

Дане програмне забезпечення має можливості: захист від зловживання з боку персоналу, доступ до інформації в залежності від рівня доступу, внесення нововведених пунктів за декілька натискань, автоматизація тарифікації часу перебування клієнта, структурування бази клієнтів і активне застосування дисконтних програм, постійний доступ до максимально повної інформації щодо тарних і фінансових запасів.

## 1.3 Аналіз сучасного стану та перспективи розвитку

Ресторанне господарство постійно розвивається та удосконалюється. Якість та рівень обслуговування закладів є об’єктом постійних досліджень та аналізу. Стратегічна мета розвитку даної індустрії в Україні полягає у створенні конкуренто спроможного, високоякісного, здатного максимально задовольнити потреби споживачів ресторанного господарство, забезпечити на цій основі комплексний розвиток галузі.

Сучасні електронні пристрої сьогодні з’являються в усіх сферах життя, не стала винятком і сфера ресторанного господарства.

Електронне меню – це ресторанне меню 21-го століття. Меню, яке дозволить відмовитися від звичних паперових меню і дасть гостям готелю відчути себе комфортно у виборі страв і при оформленні замовлення.

Статистика по закладах, що ввели дану інновацію показує, що результатом покращення якості обслуговування клієнтів, якого ми досягнемо з електронним меню, стане:

* оборот закладів збільшується, аж до 40%;
* збільшення пропускної здатності закладу в прайм- тайм на 20-25%;
* зменшення помилок, пов’язаних з прийомом замовлень і випискою рахунків;
* зменшення витрат на обслуговуючий персонал;
* збільшення середньої суми чека на 15% (за рахунок стимулювання імпульсивних замовлень);
* зменшення часу очікування на прийом і обробку замовлень на 21%;
* скорочення часу очікування на прийом повторного замовлення на 15%;
* збільшення кількості клієнтів (за рахунок підвищення популярності та пізнаваності);
* виняток витрат, пов’язаних з «людським фактором»;
* можливість обслуговування іноземних клієнтів (багатомовна підтримка);
* відсутність тимчасових і фінансових витрат на внесення змін в меню ;
* швидка окупність системи електронного меню (3-6 місяців)

Було проведено порівняння доходу закладу при використанні старого та нового зразку меню за роки їх використання (див. рис. 3).



Рисунок 3 – Рівень доходу закладу при використанні електронного меню та меню паперового

За результатами дослідження було виявлено, що надання послуг в ресторанах ще не досягло бажаного рівня. В Україні ресторани мають підвищувати свою конкурентоспроможність і виходити на світовий ринок підготовленими. Тому потрібно сконцентрувати свою увагу на розвитку чогось нового, що покращить рівень обслуговування, та вводити в дію нові технології.

Електронне меню – це потужний інструмент, що дозволяє ефективно управляти закладом, і надати клієнтам найбільш високий рівень обслуговування.

## 1.4 Постановка задачі

Завданням дипломного проекту є створити програмне забезпечення «Електронне меню» для закладів громадського харчування під операційну систему Windows 7 і вище, що забезпечить обліку інформації про страви, інгредієнти, напої, замовлення, цінові характеристики, розрахунок вартості замовлень, виведення меню в електронному вигляді. Звернути увагу, що кількість інгредієнта вимірюється в різних одиницях виміру (штуки, кілограми і літри).

Програмний продукт повинен мати простий та зручний, інтуїтивно-зрозумілий інтерфейс користувача.

Вхідні дані:

* інформація про страви та напої: фотографія страви, назва страви, категорія, собівартість, ціна, націнка, відомості;
* інформація про склад страви: назва інгредієнту, собівартість, одиниця виміру;
* інформація про замовлення: дата замовлення, прізвище та ініціали офіціанта, номер стола, оплата замовлення, дисконтна картка, дата відкриття та закриття замовлення, активність замовлення.

Вихідні дані: інформація про збір страви, рахунок відвідувача (чек), звітність за різними критеріями.

Функціональна специфікація:

* доступ до інформації в залежності від доступу до системи: адміністратора, персонал та відвідувач;
* пошук для перегляду інформації за різними критеріями;
* формування вихідних даних, тобто рахунки з інформацією про замовлення, звіти для перегляду: рейтинг страв та напоїв, рейтинг працівників, денного заробітку, витрат інгредієнтів, рейтинг клієнтів;
* візуально в інтерактивному режимі ознайомитися з меню закладу та наочно й оперативно побачити високоякісні фотографії і детальний опис кожної страви;
* розрахунок ціни та ваги за допомогою інгредієнтів для страви;
* реплікація бази даних.

Розроблена постановка задачі описує необхідні вхідні дані, які мають бути введені, та вихідні дані для виводу відповідної інформації, та функціональні можливості системи.

## 1.5 Маркетингове дослідження

Програмне забезпечення повинно надавати клієнтам можливість переглядати список меню закладів громадського харчування в електронному вигляді, офіціантам можливість швидко і якісно оформити замовлення, та адміністраторам для доступу до всіх даних.

Програмне забезпечення, що розробляється, є аналогом вже існуючих програмних продуктів. Потенційним користувачем є заклади громадського харчування.

З часом очікується високий попит на програмне забезпечення, так як він буде надавати максимально свіжу інформацію, мати зручний у використанні інтерфейс та пришвидшити обслуговування в закладах громадського харчування.

Слід надати перевагу функціоналу програмного забезпечення, так як передбачено, що дане програмне забезпечення являється багатофункціональним, в залежності від інших аналогів. Для цього потрібно розробити детальну документацію з інструкціями по встановленню та використанню програмного забезпечення. Також слід передбачити ряд певних налаштувань для більш зручної роботи з даним програмним забезпеченням.

Конкуренція на ринку висока, так як подібна технологія в нашій області достатньо швидко набирає популярність і аналогічні програмні засоби відштовхують замовників, зокрема, через велику вартість їх розробки.

З’являються спеціалізовані сайти з продажу аналогічних програмних забезпечень, на яких широкий вибір та функціонал. У даний час спеціалізовані портали з розповсюдження використовують рекламу на різних сайтах, або рекламу в мобільних додатках, що дуже сильно набирає популярність розповсюдження інформації.

Для оцінки конкурентоспроможності додатку потрібно визначити основні параметри базового та нового варіантів програмного забезпечення, виконати конструювання еталону конкурентоспроможності, розрахувати інтегральний показник конкурентоспроможності, визначити ефективність нового програмного забезпечення порівняно з базовим.

У процесі дослідження виділяються основні технічні та економічні параметри базового та нового варіантів додатку (див. табл. 1.1).

Таблиця 1.1 – Основні техніко-економічні параметри додатку

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва параметру | Одиниця виміру | Варіант | | Характеристика параметра нового варіанта відносно базового(↑,↓ чи =) |
| базовий  (аналог) | новий |
| Вартість додатку | грн | 7800 | 6999 | ↑ |
| Результати дослідження інтерфейсу методом GOMS | сек | 3,5 | 3,1 | ↑ |
| Кількість функцій додатку | шт. | 5 | 5 | = |
| Вага додатку | Мб | 105 | 66 | ↑ |
| Кількість інформації, що передається | біт | 10 | 12 | ↑ |
| Можливість нарощування функціональних характеристик | бали | 4 | 3 | ↓ |
| Кількість людей, необхідних для обслуговування | чол. | 1 | 1 | = |

Виходячи з отриманих результатів, розробка є кращою відносно базового варіанта за наступними параметрами: вартість програмного забезпечення, результати дослідження методом GOMS, кількість функцій, вага програмного забезпечення, кількість інформації, що передається. Новий варіант програмного забезпечення поступається аналогу за можливістю нарощування функціональних характеристик.

Для визначення конкурентоспроможності необхідно сконструювати еталон конкурентоспроможності додатку.

Еталон – це міра для відтворення, зберігання і передачі одиниць будь-якої величини, що утворена за таким правилом: серед показників-стимуляторів (здійснюється позитивний вплив на конкурентоспроможність) відбираємо дані з максимальним значенням, а серед показників де стимуляторів (чинить негативний плив на конкурентоспроможність) відбираємо дані з мінімальним значенням. Визначення еталонного значення наведено у таблиці 1.2 (колонки 2, 3).

Інтегральний показник конкурентоспроможності відносно еталона () визначається за формулою (1.1):

, (1.1)

де ,  – величини за *і*-тим параметром відповідно базового, нового варіанта ПЗ та еталона.

Результати розрахунків заносимо до таблиці 1.2 (колонки 4, 5).

Проведені розрахунки будуть свідчити про те, що базове та нове програмне забезпечення може переважати еталон за параметрами. Результатом буде одна із наступних умов:

* якщо , то базове програмне забезпечення перевищує новий зразок за конкурентоспроможністю;
* якщо , то поступається йому;
* при , нове програмне забезпечення знаходиться на одному рівні з базовим.

Таким чином, базовий додаток переважає за еталон параметрами на «7,8», а новий – на «6,5». Тобто новий мобільний додаток є безперечно конкурентоспроможним порівняно з базовим.

Загальна ефективність розробки нового програмного забезпечення порівняно з базовим визначається за формулою (1.2):

 (1.2)

Якщо серед параметрів є такі, для яких мінімальне значення є найбільш ефективним, то має місце рівність (1.3):

, (1.3)

де ,  – величини за *і*-тим параметром відповідно нового та базового варіанту програмного забезпечення.

Проте, якщо серед параметрів є такі, для яких максимальне значення є найбільш ефективним, то має місце рівність (1.4):

 (1.4)

Для розрахунку загальної ефективності визначимо відносне значення параметрів нового програмного забезпечення відносно базового (див. табл. 1.2, колонка 6).

Розрахунок ефективності окремого параметру нового програмного забезпечення здійснюється за формулами (1.3) та (1.4), результати розрахунків заносимо до таблиці 1.2 (колонки 7, 8).

Таблиця 1.2 – Параметри конкурентоспроможності та показники ефективності

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва параметра | Еталонне значення | | Варіант | |  |  |  |
|  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Вартість додатку | min | 6999 | 1,1 | 1 | 1,1 | 0,2 |  |
| Результати дослідження інтерфейсу методом GOMS | min | 3,1 | 1,1 | 1 | 1,1 | 0,3 |  |
| Кількість функцій додатку | max | 5 | -1 | 1 | 1 |  | 0,7 |
| Вага додатку | min | 66 | 1,6 | 1 | 1,6 | 0,5 |  |
| Кількість інформації, що передається | max | 12 | -0,8 | 1 | 1,3 |  | 1,5 |
| Можливість нарощування функціональних характеристик | max | 4 | 1 | 0,75 | 0,75 |  | -0,5 |
| Кількість людей, необхідних для обслуговування | min | 1 | 1 | 1 | 1 |  |  |
| Разом: |  |  | 4 | 5,75 |  | 2,7 | |

Оскільки загальна вартість додатку, що розробляється, не перевищує вартість аналогу, то виграш від цього буде складати: 1 – 0,8 = 0,2. При розробці додатку вдалося покращити його інтерфейс відносно базового (вдосконалення призвело до спрощення та зручності користування), тому виграш складає: 1 – 0,7 = 0,3. При запланованій кількість функцій додатку, виграш буде складати: 1,7 – 1 = 0,7. За рахунок зменшення ваги додатку можна отримати виграш у розмірі: 1 – 0,5 = 0,5. Передбачена кількість інформації, що передається, дозволить отримати виграш: 2,5 –1 = 1,5, при запланованій можливості нарощення функціональних характеристик, програш складе: 0,5 – 1 = –0,5.

Відповідно до цього, загальна економічна ефективність складе:

*Ефзаг* = 0,2 + 0,3 + 0,7 + 0,5 + 1,5– 0,5 = 2,7.

Таким чином, ефективність виробництва нового програмного забезпечення відносно аналогу складає 2,7.

Отже, у проекті обґрунтовано доцільність розробки нового програмного забезпечення, розраховано його собівартість 13659 грн., визначено договірну ціну – 21 001 грн., розраховано, що період, протягом якого можливо здійснити розробку додатку та запровадити його в дію, – цей період складає 60 днів.

При виконанні маркетингового дослідження інно вації розробки мобільного додатку, виконавши усі необхідні розрахунки, встановлено, що технологія нової розробки відповідає оптимальному рівню витрат, і, у підсумку, розроблений додаток є економічно доцільним та конкурентоспроможним для вільного розповсюдження на ринку додатків для мобільних пристроїв.

# РОЗДІЛ 2. ТЕХНІЧНИЙ ПРОЕКТ

## 2.1 Технічне завдання

Застосування та вимоги до програмного забезпечення для закладів громадського харчування досить різноманітні, тому існують різні версії, які можуть не містити стандартні функцій, проте мати певні специфічні додаткові функції, необхідні конкретній категорії користувачів.

Призначення розробки: розробити програмне забезпечення для обліку замовлень в закладах громадського харчування під управління ОС Windows.

Програмне забезпечення, що розробляється повинен вирішувати наступні задачі: принципові, функціональні, сервісні.

До принципових задач відносять:

* забезпечення збереження та обробка інформації меню та ефективний пошук даних за ключовими словами;
* мінімізація можливості виникнення помилок у роботі з програмним забезпеченням;
* контроль операцій, що використовуються для пошуку;
* зручний, простий та зрозумілий інтерфейс;
* контроль операцій, які можуть привести до змінення або пошкодження даних.

До функціональних даних відносять:

* збереження зміни даних;
* можливість керуванням рівнем доступу, для обходу зловживання з боку персоналу;
* можливість перегляду звітності за різний період часу;
* можливість керуванням активності відображення інформації для клієнтів.

До сервісних задач відносяться:

* організація головного меню та адміністративної частити;
* організація рівня доступу до програмного забезпечення;
* інформування користувача про причину відмови у виконанні тієї чи іншої операції;
* забезпечення маніпуляції з даними та синхронність даних;

Системні й апаратні вимоги які необхідні для встановлення програми наведено у таблиці 2.1.

Таблиця 2.1 – Системні й апаратні вимоги для встановлення програми

|  |  |  |
| --- | --- | --- |
| Компонент | Мінімальні вимоги | Рекомендовані вимоги |
| Процесор | Pentium / Celeron, AMD K6 / Athlon і інші, 1.1 ГГц | Pentium IV, 1.5ГГц |
| Дисковий простір | 300 Мб | 500 Мб |
| Оперативна пам’ять | 256 Мб | 512 Мб |
| Операційна система | Windows 7, 8, 8.1, 10 | |
| Додатки | ODBC-драйвери для SQL Server | |

Програмне забезпечення повинно підтримувати спільний режим з операційною системою Windows 7 та вище. Вимоги до мови програмування не передбачаються.

У процесі розробки програмного забезпечення можливе використання стандартних програмних засобів, які використовуються для розробки програмного забезпечення під управління операційної системи Windows.

Етапи розробки програмного забезпечення можуть уточнюватися згідно календарного плану робіт по узгодженню між замовником та виконавцем (див. табл. 2.2).

Таблиця 2.2 – Етапи роботи над дипломним проектом

| Етапи виконання роботи | Термін виконання та приблизний обсяг робіт | Звітні матеріали |
| --- | --- | --- |
| Передпроектні дослідження | 25.04-06.05  Аналіз предметної області та існуючих аналогів, вивчення технологій, структури даних, методів рішення тощо | Опис предметної області, аналіз аналогів, вибору методів рішення завдання та засобів розробки, оформлення технічного завдання |
| Технічний проект | 10.05-16.05  Проектування програмного забезпечення. Розробка алгоритму, визначення форми представлення даних, архітектури програми | Специфікація вимог (опис алгоритму, правил, функціональних вимог, критеріїв якості тощо), опис концептуальної, інформаційної та функціональної моделей |
| Робочий проект | 17.05-11.06  Реалізація програмного забезпечення | Опис засобів розробки, розробка документів на супроводження ПЗ (інструкції програмісту та користувачу), опис плану тестування (розробка тестів, аналіз результатів тестування) |

Оцінка результатів розробки і доцільність її продовження здійснюється замовником по представленню наступних матеріалів:

* встановлене програмне забезпечення на комп’ютерному обладнанню замовника;
* перелік файлів на резервному носії;
* стислий опис роботи додатку та опис всіх файлів, які необхідні для його роботи;
* перелік документів: технічне завдання, пояснювальна записка, додаток та методика тестування, керівництва користувача та програміста.

Склад і зміст дипломного проекту: передпроектні дослідження: аналіз предметної області та існуючих аналогів, постановка задачі, маркетингове дослідження інновацій розробки мобільного додатку, технічний проект: технічне завдання специфікація вимог, концептуальна, логічна та фізична моделі, проектування інтерфейсу, робочий проект: засоби розробки, документи на супроводження додатку (інструкції програмісту та користувачу), план тестування (розробка тестів та аналіз результатів тестування), охорона праці та техніка безпеки, висновки, додатки.

Все програмне забезпечення та супроводжуюча технічна документація повинні відповідати наступним ГОСТам:

* ІСО 5807-85 ГОСТ на розробку програмних документів, схем алгоритмів програм, даних та систем.
* ГОСТ 34.602-89 – Комплекс стандартів на автоматизовані системи. Технічне завдання на створення автоматизованої системи.
* ГОСТ 19.201-78 – Єдина система програмної документації. Технічне завдання. Вимоги до змісту та оформлення.
* ГОСТ 19.781-74 – Вимоги до розробки програмного забезпечення.
* ГОСТ 19.101-77 – Держстандарт на розробку програмної документації, видів програм та програмних документів.
* ГОСТ 29.401-78 – Текст програми. Вимоги до змісту та оформлення.
* ГОСТ 19.106-78 – Вимоги до програмної документації.
* ГОСТ 7.1-84 та ДСТУ 3008-95 – Розробка технічної документації.

Тестування виконується відповідно до «Програми та методики тестування», яка розробляється виконавцем та затверджується замовником.

У процесі розробки програмного забезпечення виконуються наступні види випробувань:

* тестування елементів системи;
* тестування системи у цілому;
* дослідна експлуатація.

Після закінчення відповідного етапу робіт формується відповідний комплект документації.

Завершення етапу розробки фіксується відповідним протоколом захисту дипломних проектів.

## 2.2 Моделювання програмного забезпечення

Після дослідження предметної області, можна побудувати концептуальну модель даних, у рамках якої інформація предметної області виражається деякими абстрактними засобами.

Концептуальна модель – це абстрактна модель, що визначає структуру модельованої системи, властивості її елементів і причинно-наслідкові зв’язки, властиві системі та суттєві для досягнення мети моделювання. Для зображення концептуальної моделі використовується діаграма прецедентів.

Діаграма прецедентів – діаграма, на якій зображено відношення між акторами та прецедентами в системі (див. рис. 2.1).



Рисунок 2.1 – Діаграма прецедентів

Опис діаграми прецедентів:

* Актор «Користувач» – даний актор поділяється на рівні доступу до програмного забезпечення. Після авторизації для звичайного користувач, з доступом до програмного забезпечення є: «Офіціант» – при взаємодії з програмним забезпеченням який має доступ до: додавання та редагування інформації про замовлення та перегляду звітності. Актор «Адміністратор» – даний актор при взаємодії з програмним забезпеченням має повний доступ до маніпуляції над інформацією;
* Актор «Клієнт» – даний актор при взаємодії з програмним забезпеченням, має доступ для перегляду інформації про страви та напої в електронному вигляді, для вибори під час оформлення замовлення.

Розроблена діаграма прецедентів описує структуру взаємодії між співробітником, клієнтом та програмним забезпеченням.

Діаграма взаємодій – діаграма, на якій зображено етапи взаємодії користувача з програмним забезпеченням (див. рис. 2.2).



Рисунок 2.2 – Діаграма взаємодії

Діаграма послідовності – діаграма якій описується послідовність дій під час оформлення замовлення. Коли клієнт вибирає пункти меню, офіціант оформляє замовлення, подає готове замовлення на кухню, очікує оплату, після чого офіціант віддає замовлення та рахунок клієнту (див. рис. 2.3)



Рисунок 2.3 – Діаграма послідовності

Отже, проаналізувавши предметну область було спроектовано концептуальну модель, діаграму взаємодії та послідовності.

## 2.3 Моделювання даних

Створена модель відображає концептуальні аспекти побудови об’єктної моделі програмного забезпечення і відносяться до логічного рівня представлення. Основне значення логічного представлення складається із аналізу структурних і функціональних зв’язків між елементами моделі.

Однак, для створення конкретної фізичної схеми необхідно певним чином реалізувати всі елементи логічного представлення в конкретні матеріальні сутності. Для опису таких реальних сутностей призначений другий аспект модельного представлення, а саме фізичне представлення об’єктної моделі. Для зберігання даних спроектована база даних, яка містить 8 таблиць зв’язки мiж якими представлені на рисунку 2.4.



Рисунок 2.4 – Логічна модель бази даних

В якості бази даних використовується СУБД MSSQL Server. База даних складається з наступних таблиць: OrderMenu, ListOrdeMenu, Food, Personal, Discounts, ListIngredientFood, ListTable, Category.

У таблиці «OrderMenu» зберігається основна інформація про замовлення. Структура таблиці вказана в таблиці 2.3.

Таблиця 2.3 – Структура таблиці «OrderMenu»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Признак ключа | Примітки |
| id\_order | INT | PK | Унікальне значення |
| date\_open\_order | DATETIME |  | Оформлення замовлення |
| date\_close\_order | DATETIME |  | Закриття замовлення |
| kod\_table | INT | FK | Ідентифікатор столів. Вказує на запис в таблиці ListTable |
| close\_order | BIT |  | Активність замовлення |
| payment | NUMERIC |  | Сума оплати за замовлення |
| kod\_personal | INT | FK | Ідентифікатор персоналу. Вказує на запис таблиці Personal |
| kod\_discount | INT | FK | Ідетнифікатор знижки. Вказує на запис таблиці Discounts |

У таблиці «ListOrderMenu» зберігається інформація про вміст замовлення. Структура таблиці вказана в таблиці 2.3.

Таблиця 2.3 – Структура таблиці «ListOrderMenu»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Признак ключа | Примітки |
| id\_listordermenu | INT | PK | Унікальне значення |
| kod\_order | INT | FK | Ідентифікатор замовлення. Вказує на запис таблиці OrderMenu |
| kod\_food | INT | FK | Ідентифікатор страв або напоїв. Вказує на запис таблиці Food |
| counts | NUMERIC |  | Кількість страв або напоїв |

У таблиці «Food» зберігається інформація про страви та напої. Структура таблиці вказана в таблиці 2.4.

Таблиця 2.4 – Структура таблиці «Food»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Признак ключа | Примітки |
| id\_food | INT | PK | Унікальне значення |
| name\_food | VARCHAR(50) |  | Найменування страви або напою |
| picture | VARBINARY |  | Зображення |
| kod\_category | INT | FK | Ідентифікатор категорії. Вказує на запис таблиці Category |
| weight\_food | NUMERIC |  | Кількість (вага) для страви або напою |
| unit\_food | VARCHAR(10) |  | Одиниця виміру |
| cost\_price\_food | NUMERIC |  | Собівартість страви або напою |
| mark\_up | INT |  | Націнка до собівартості страви або напою |
| price\_food | NUMERIC |  | Ціна для продажу страви або напою |
| data\_food | VARCHAR |  | Опис страви або напою |
| visible | BIT |  | Відображення страви або напою в електронному меню |

У таблиці «Personal» зберігається інформація про користувача та доступ до програмного забезпечення. Структура таблиці вказана в таблиці 2.5.

Таблиця 2.5 – Структура таблиці «Personal»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Признак ключа | Примітки |
| id\_personal | INT | PK | Унікальне значення |
| pib\_personal | VARCHAR(50) |  | Прізвище ініціали користувача |
| logins | VARCHAR(50) |  | Логін для доступу до програмного забезпечення |
| passwords | VARCHAR(50) |  | Пароль для доступу до програмного забезпечення |
| access | BIT |  | Рівень доступу (адміністратор, користувач) |
| activity | BIT |  | Активність користувача |
| telefon | VARCHAR(20) |  | Номер телефону |
| address | VARCHAR(50) |  | Адреса проживання |
| birthday | DATE |  | Дата народження |
| data\_of\_work | DATETIME |  | Дата обліку |
| release\_data | DATETIME |  | Дата звільнення |

У таблиці «Discounts» зберігається інформація про знижки та додаткова інформація про клієнта. Структура таблиці вказана в таблиці 2.6.

Таблиця 2.6 – Структура таблиці «Discounts»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Признак ключа | Примітки |
| id\_discount | INT | PK | Унікальне значення |
| pib\_client | VARCHAR(50) |  | Прізвище ініціали клієнта |
| birthday | DATETIME |  | Дата народження клієнта |
| address | VARCHAR(50) |  | Адреса проживання клієнта |
| telefon | VARCHAR(20) |  | Номер телефону клієнта |
| discount | INT |  | Знижка |

У таблиці «ListIngredientFood» зберігається інформація інгредієнти пов’язані зі стравою. Структура таблиці вказана в таблиці 2.7.

Таблиця 2.7 – Структура таблиці «ListIngredientFood»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Признак ключа | Примітки |
| id\_listingrfood | INT | PK | Унікальне значення |
| kod\_food | INT | FK | Ідентифікатор страви. Вказує на запис таблиці Food |
| counts | NUMERIC |  | Кількість (вага) інгредієнта |
| name\_ingredient | VARCHAR(50) |  | Найменування інгредієнта |
| unit | VARCHAR(10) |  | Одиниця виміру інгредієнта |
| price | NUMERIC |  | Ціна за одиницю виміру |

У таблиці «ListTable» зберігається інформація про найменування місць для обслуговування клієнтів. Структура таблиці вказана в таблиці 2.8.

Таблиця 2.8 – Структура таблиці «ListTable»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Признак ключа | Примітки |
| id\_table | INT | PK | Унікальне значення |
| name\_table | VARCHAR(50) |  | Найменування місця для обслуговування |
| occupation | BIT |  | Зайнятість |

У таблиці «Category» зберігається інформація про найменування місць для обслуговування клієнтів. Структура таблиці вказана в таблиці 2.9.

Таблиця 2.9 – Структура таблиці «Category»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Признак ключа | Примітки |
| id\_category | INT | PK | Унікальне значення |
| name\_category | VARCHAR(50) |  | Найменування категорії |

Зв’язок між таблицями «OrderMenu» і «ListOrderMenu» – один-до-багатьох, так як замовлення може містити декілька записів вибраних страв або напоїв, зв’язок між таблицями «Food» і «ListOrderMenu» – один-до-багатьох, так як в таблиці страви можна вибрати один запис страви або напою для списку замовлення, зв’язок між таблицями «Food» і «ListIngredientFood» – один-до-багатьох, так як одна страва може містити багато записів про інгредієнти, зв’язок між таблицями «Category» і «Food» – один-до-багатьох, так як одна категорія містить багато записів про страви або напої, зв’язок між таблицями «OrderMenu» і «Discounts» – один-до-одного, так як замовлення може містити лише один запис пов’язаний з знижкою, зв’язок між таблицями «OrderMenu» і «Personal» – один-до-одного, так як замовлення може містити лише один запис пов’язаний з працівником, який обслуговував клієнта, зв’язок між таблицями «OrderMenu» і «ListTable» – один-до-одного, так як одне замовлення містить лише один запис про місце обслуговування клієнта.

Для реалізації поставленої задачі використано принцип об’єктно-орієнтованого програмування та було спроектована ієрархію класів, яка представлена у вигляді діаграми класів (див. додаток А).

## 2.4 Проектування інтерфейсу програмного забезпечення

Взаємодія між користувачем і програмним забезпечення відбувається за допомогою інтерфейсу. Найчастіше ефективність програмного забезпечення визначають у більшому ступені тим, як побудований його інтерфейс. Тому інтерфейс програмного забезпечення має бути простий, зручний та інтуїтивно-зрозумілий.

Для реалізації інтерфейсу програмного забезпечення необхідно спроектувати декілька вікон, для зручності роботи з програмним забезпеченням враховано, що адміністратор та офіціант, та клієнт маються різні можливості, так як у них є різний рівень доступу.

Інтерфейс форми «Головне вікно» програмного забезпечення зображено на рисунку 2.5. Дана форма призначена для перегляду вмісту інформації меню в електронному вигляді, який можна переглядати при виборі категорії меню, в якому буде відображено повну інформацію пов’язану з даною категорію. Також реалізовано пошук за допомогою якого, можна знайти потрібно інформацію яка пов’язана з стравами або напоями. Також дана форма використовується для оформлення замовлення, після авторизації в систему, відображаються скриті компоненти, які надаються можливість офіціанту вибрати страви та напої для оформлення замовлення (див. рис. 2.6).



Рисунок 2.6 – Форма «Головне вікно»

Інтерфейс форми «Замовлення» програмного забезпечення зображено на рисунку 2.7. Дана форма призначена для офіціанта, для якого буде реалізовано, управління даними тобто: оформлення замовлення, закриття замовлення, додавання та видалення вибраних страв або напоїв з замовлення, знижки для клієнтів, друк рахунку, перегляд вільних місць, інформація про замовлення.



Рисунок 2.7 – Форма «Замовлення»

Інтерфейс форми «Адміністратор» програмного забезпечення зображено на рисунку 2.8. Дана форма призначена для адміністратора, який має повний доступ до програмного забезпечення, тобто: обробка даних, пошук інформації за різним ключем, створення звітності за різними критеріями, управління користувачами та доступ до програмного забезпечення.



Рисунок 2.8 – Форма «Адміністратор»

Для авторизації користувача під різний рівень доступу спроектовано інтерфейс форми «Авторизація», яка зображена на рисунку 2.9. Дана форма призначена для вводу особистих даних та вибору необхідної дії.



Рисунок 2.9 – Форма «Авторизація»

Інтерфейс форми «Редагування інформації про категорію» зображено на рисунку 2.10. Дана форма призначена для додавання та редагування інформації про категорію.



Рисунок 2.10 – Форма «Редагування інформації про категорію»

Інтерфейс форми «Редагування інформації про знижку» зображено на рисунку 2.11. Дана форма призначена для додавання та редагування інформації про знижку та додаткову інформацію пов’язану з клієнтом.



Рисунок 2.11 – Форма «Редагування інформації про знижку»

Інтерфейс форми «Редагування інформації про елемент меню» зображено на рисунку 2.12. Дана форма призначена для додавання та редагування інформації пов’язано з напоєм або стравою та інгредієнтами до страви, також передбачена можливість розрахунку ціни або ваги страви враховуючи склад страви.



Рисунок 2.12 – Форма «Редагування інформації про елемент меню»

Інтерфейс форми «Додати інформацію про інгредієнт» зображено на рисунку 2.13. Дана форма призначена для додавання інформації про інгредієнт до відповідної страви.



Рисунок 2.13 – Форми «Додати інформацію про інгредієнт»

Інтерфейс форми «Редагування інформації користувача» зображено на рисунку 2.14. Дана форма призначена для додавання та редагування інформації про користувача, який має доступ до програмного забезпечення.



Рисунок 2.14 – Форми «Редагування інформації про користувача»

Інтерфейс форми «Додати інформацію про стіл» зображено на рисунку 2.15. Дана форма призначена для додавання інформації про місце обслуговування клієнтів.



Рисунок 2.15 – Форма «Додати інформацію про стіл»

Отже, в даному розділі було спроектовано архітектуру, структуру та інтерфейс програмного забезпечення, також було спроектована моделювання програмного забезпечення, моделювання даних та описано технічне завдання.

# РОЗДІЛ 3. робочий проект

## 3.1 Засоби розробки

Для створення дієвого і простого у використанні програмного продукту було обрано середовище Embarcadero RAD Studio 10.2 Tokyo С++ Builder.

Embarcadero RAD Studio 10.2 – це комплексне рішення для швидкої розробки додатків Window, .NET, веб-додатків і додатків баз даних. Embarcadero RAD Studio 10.2 об’єднує Delphi 10.2 і С++ Builder 10.2 в єдину інтегровану середу розробки. У даній версії також реалізована підтримка таких технологій: .NET, як WinForms, WPF, ADO.NET, ASP.NET і LINQ.

Додаткова підтримка фреймворку Mono забезпечує можливості створення крос-платформних додатків, які працюють під операційними системами Windows, Linux і Mac OS X.

Спочатку розроблявся компанією Borland Software, а потім її підрозділом CodeGear, що на сьогоднішній день належить компанії Embarcadero Technologies.

Embarcadero RAD Studio 10.2 об’єднує в собі комплекс об’єктних бібліотек (STL, VCL, CLX, MFC та ін.), компілятор відкладчик, редактор коду та багато інших компонентів.

Можливості Embarcadero RAD Studio 10.2:

* підтримка Windows 7 API і Direct2D;
* підтримка жестів і технології multi-touch для Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8-8.1 та Windows 10;
* IDE Insight – миттєвий доступ до будь-якої функції компонентів, налаштувань;
* понад 120 удосконалень для підвищення продуктивності;
* візуалізатори налагодження;
* підтримка Firebird.

Класичний інтерфейс Delphi 7 і C++ Builder 6 – RAD Studio Architect включає всю функціональність версії Enterprise, а також потужні можливості моделювання та проектування баз даних:

* зворотне проектування, аналіз і оптимізація баз даних;
* автоматична генерація коду баз даних на основі моделей для прямого проектування;
* створення логічних і фізичних моделей на основі відомостей, витягнутих з баз даних і файлів сценаріїв;
* робота із зручними для читання і навігації діаграмами;
* вдосконалене порівняння і об’єднання за допомогою двонаправлених операцій порівняння і об’єднання моделей і структур баз даних.

Основні можливості для розробки додатків Windows:

* інтегроване середовище розробки Delphi і С++ Builder з функцією перетягування курсором;
* більше 250 компонентів в бібліотеці VCL;
* вбудована підтримка торкань і жестів (більше 30 готових жестів);
* dbExpress з підтримкою 9 основних баз даних, включаючи InterBase;
* DataSnap з підтримкою JSON, REST, HTTP, COM і XML;
* Можливість налагодження багато поточних додатків;
* UML / аудит коду та облік показників;
* підтримка Windows 2000 та вище.

## 3.2 Інструкція програміста

Даний дипломний проект розроблявся під операційну систему Windows 7 та вище. Дипломний проект складається з 8 форм, 1 модуля та 2 файлів в яких описані класи та запуск програмного забезпечення (див. рис. 3.1).



Рисунок 3.1 – Вміст проекту

Модуль «DataModule» – модуль в якому розроблено компоненти для маніпуляції з базою даних. Опис компонентів:

* компонент класу TADOConnection – призначений для підключення програмного забезпечення до бази даних;
* компоненти класу TImageList – призначені для збереження зображення, які будуть відображуватися в програмному забезпечені;
* компоненти класу TADOTable – компоненти призначені для з’єднання з таблицями в базі даних;
* компоненти класу TADOTable – компоненти призначені для виконання запиту до бази даних;
* компоненти класу TDataSource – компоненти призначені для з’єднання компонентів класу TADOTable та TADOQuery з компонентами для перегляду або маніпулювання базою даних;
* компоненти класу TMainMenu – компоненти для відображення головного меню в програмного забезпечені;
* компоненти класу TfrxReport – компоненти призначені для формування звітності та відображення їх вмісту;
* компоненти класу TfrxDBDataset – компоненти, які зв’язують компоненти класу TADOQuery з TfrxReport.

Для реалізації функціональної частини програмного забезпечення було використано об’єктно-орієнтоване програмування, ієрархію класів описано в файлі «eMenuPCH1.h», також можна переглянути в додатку А.

Розроблено 8 класів для маніпулювання даними в базі даних: TPersonal, TOrderMenu, TListOrder, TFood, TListTable, TCategory, TDiscount, TIngredient.

Методи InsertDB\*,UpdateDB\*,DeleteDB\* призначені для додавання, оновлення та видалення даних в базі даних. Методи set\*, get\* призначені оновлення, та отримання даних з атрибутів класу.

Методи класу TDM:

* OpenDB() – призначений для оновлення даних в усіх таблицях;
* OpenDBMain() – призначений для оновлення даних в таблицях для головного вікна;
* OpenDBOficiant() – призначений для оновлення даних в таблицях, які пов’язані з замовленням.

Meтоди класу TFAuth:

* NotUserStatus() – метод який надає звичайний доступ до програмного забезпечення тільки для перегляду даних.
* UserStatus() – метод який надає доступ до програмного забезпечення для маніпулюванням даних пов’язані з замовленням;
* AdminStatus() – метод який надає повний доступ до програмного забезпечення.

Клас TPersonal – призначений для маніпулювання даними в базі даних з таблицею «Personal». Даний клас містить такі атрибути:

* id\_personal – для зберігання ідентифікатора користувача;
* pib\_personal – для зберігання прізвище ініціали;
* logins – для зберігання логіну;
* passwords – для зберігання паролю;
* access – для зберігання доступу до програмного забезпечення;
* activity – для зберігання активності користувача;
* telefon – для зберігання номеру телефону;
* address – для зберігання адреси проживання
* birthday – для зберігання дати народження;
* data\_of\_work – для зберігання дати обліку в програмному забезпеченні;
* release\_date – для зберігання дати зняття з обліку в програмному забезпеченні.

Методи даного класу:

* методи InsertDB, UpdateDB, DeleteDB – для маніпулювання з базою даних;
* сетери та гетери для оновлення та отримання даних в атрибутах класу;
* UpdateDBPersonalRelease() – оновлення даних про дату звільнення.

Клас TListTable – призначений для маніпулювання даними в базі даних з таблицею «ListTable». Даний клас містить такі атрибути:

* id\_table – для збереження ідентифікатора місця обслуговування клієнта;
* name – для збереження назви місця обслуговування клієнта;
* occupation – для збереження зайнятості місця.

Методи даного класу:

* методи InsertDB, UpdateDB, DeleteDB – для маніпулювання з базою даних;
* сетери та гетери для оновлення та отримання даних в атрибутах класу;
* UpdateOccupation() – оновлення поля occupation в базі даних;
* UpdateInverseOccupation() – оновлення поля occupation в базі даних на протилежне його значення.

Клас TDiscount – призначений для маніпулювання даними в базі даних з таблицею «Discounts». Даний клас містить такі атрибути:

* id\_discount – для збереження ідентифікатора знижки;
* discount – для збереження знижки;
* pib\_client – для збереження прізвище ініціали клієнта;
* address – для збереження адреси проживання клієнта;
* telefon – для збереження номеру телефону клієнта;
* birthday – для збереження дати народження клієнта.

Методи даного класу:

* методи InsertDB, UpdateDB, DeleteDB – для маніпулювання з базою даних;
* сетери та гетери для оновлення та отримання даних в атрибутах класу.

Клас TCategory – призначений для маніпулювання даними в базі даних з таблицею «Category». Даний клас містить такі атрибути:

* id\_category – для збереження ідентифікатора категорії;
* name\_category – для збереження найменування категорії.

Методи даного класу:

* методи InsertDB, UpdateDB, DeleteDB – для маніпулювання з базою даних;
* сетери та гетери для оновлення та отримання даних в атрибутах класу.

Клас TFood – призначений для маніпулювання даними в базі даних з таблицею «Food». Даний клас наслідує методи та атрибути класу «TCategory», також містить такі атрибути:

* id\_food – для збереження ідентифікатора страви або напою;
* name\_food – для збереження найменування страви або напою;
* data\_food – для збереження відомості;
* unit\_food – для збереження одиниці виміру;
* weight\_food – для збереження кількості або ваги;
* cost\_price\_food – для збереження собівартості;
* mark\_up – для збереження націнки;
* price\_food – для збереження ціни;
* picture – для збереження зображення;
* visible – для збереження відображення в електронному меню.

Методи даного класу:

* методи InsertDB, UpdateDB, DeleteDB – для маніпулювання з базою даних;
* сетери та гетери для оновлення та отримання даних в атрибутах класу;
* UpdateFotoDBFood() – метод, якрий призначений для оновлення поля picture.

Клас TIngredient – призначений для маніпулювання даними в базі даних з таблицею «ListIngredientFood». Даний клас наслідує методи та атрибути класу «TFood», також містить такі атрибути:

* id\_ingredient – для збереження ідентифікатору інгредієнта;
* name\_ingredient – для збереження найменування інгредієнта;
* unit – для збереження одиниці виміру інгредієнта;
* price – для збереження ціни інгредієнта;
* count\_unit – для збереження кількості або ваги інгредієнта.

Методи даного класу:

* методи InsertDB, UpdateDB, DeleteDB – для маніпулювання з базою даних;
* сетери та гетери для оновлення та отримання даних в атрибутах класу.

Клас TOrderMenu – призначений для маніпулювання даними в базі даних з таблицею «OrderMenu». Даний клас наслідує методи та атрибути класів «TDiscount», «TPersonal» та «TListTable», також містить такі атрибути:

* id\_order – для збереження ідентифікатору замовлення;
* date\_open\_order – для збереження дати оформлення замовлення;
* date\_close\_order – для збереження дати закриття замовлення;
* close\_order – для збереження активності замовлення;
* payment – для збереження оплати за замовлення.

Методи даного класу:

* методи InsertDB, UpdateDB, DeleteDB – для маніпулювання з базою даних;
* сетери та гетери для оновлення та отримання даних в атрибутах класу;
* CloseOrderMenu() – метод призначений для закриття замовлення;
* UpdatePayment() – метод призначений для оновлення оплати замовлення;
* UpdateDiscount() – метод призначений для оновлення знижки до замовлення.

Клас TListOrder – призначений для маніпулювання даними в базі даних з таблицею «ListOrderMenu». Даний клас наслідує методи та атрибути класів «TFood», «TOrderMenu», також містить такі атрибути:

* id\_listingrfood – для збереження ідентифікатора
* counts – для збереження кількості.

Методи даного класу:

* методи InsertDB, DeleteDB – для маніпулювання з базою даних;
* сетери та гетери для оновлення та отримання даних в атрибутах класу.

Лістинг програми наведено у додатку Б.

## 3.3 Інструкція користувача

Для роботи з програмним забезпеченням спочатку потрібно встановити SQL Server, створити базу даних запустивши файл «Create DB eMenu.sql», після цього розпакувати файл з програмним забезпеченням на персональний комп’ютер, налаштувати дані для доступу до локальної бази даних в файлі «options.ini». Всі ці дії повинен зробити досвідчений спеціаліст. Після завершення цих дій, для запуску програми потрібно запустити файл «eMenu.exe». Після запуску програми, можна побачити стартове вікно програмного забезпечення (див. рис. 3.2). На вікні розташовано:

* головне меню, для вибору певної дії;
* поля для пошуку інформації в меню;
* компонент для вибору категорії меню;
* компонент для вибору страви або напою, який фільтрується за вибраною категорією.



Рисунок 3.2 – Стартове вікно

При натисканні на пункт меню «Авторизація», з’являється вікно «Авторизація» (див. рис. 3.3) для введення особистих даних, для доступу до програмного забезпечення під різний рівень доступу: адміністратор та користувач.



Рисунок 3.3 – Вікно «Авторизація»

Якщо, був здійснений вхід з доступом під користувача, тоді з’являються додатковий пункт в головному меню: замовлення. Також відображаються компоненти для маніпулюванням над замовленням (див. рис. 3.4) і надається доступ до вікна «Замовлення» (див. рис. 3.5). На головному вікні розташовано додаткові компоненти, як :

* панель на якому відображається інформація про поточне замовлення;
* кнопка для переходу на вікно «Замовлення»;
* кнопка для додавання страви або напою до поточного замовлення.

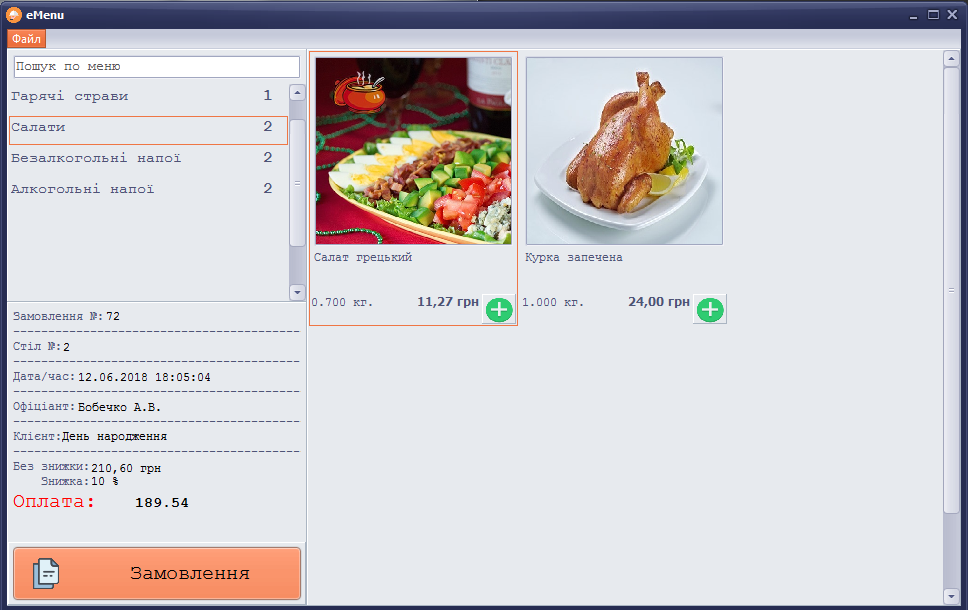


Рисунок 3.4 – Головне вікно в режимі користувача

На вікні «Замовлення» розташовано:

* кнопка для оформлення замовлення;
* кнопка для закриття замовлення;
* випадаючий список з інформацією про оформлені замовлення;
* випадаючий список з інформацією про вільні місця для обслуговування;
* кнопка для додавання страв або напоїв до замовлення;
* кнопка для видалення страви або напою з замовлення;
* кнопка для друку рахунку за вибраним замовлення, при умові, що замовлення не закрите;
* кнопка для знижки до замовлення;
* кнопка для відображення денного заробітку.

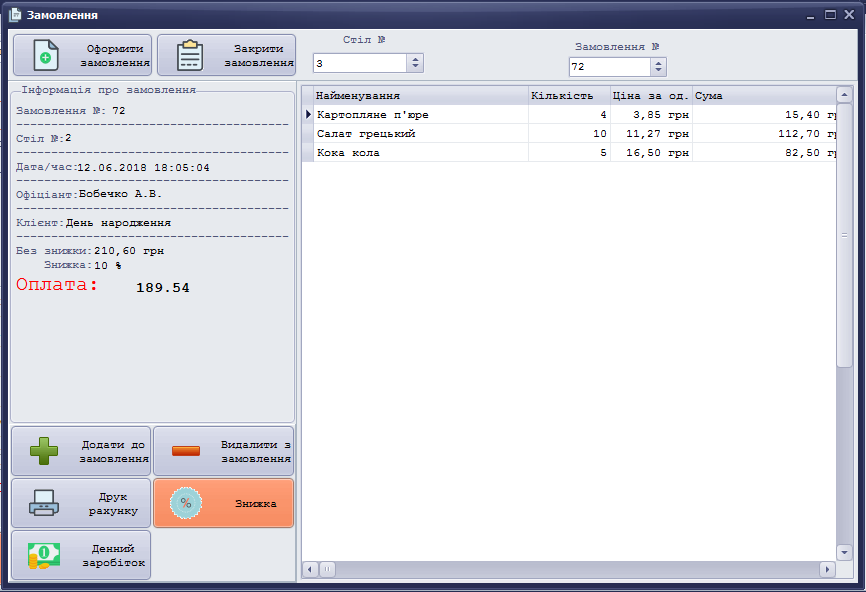


Рисунок 3.5 – Вікно «Замовлення»

Якщо, був здійснений вхід з доступом під адміністратор, тоді надається повний доступ до програмного забезпечення, надається доступ до вікна «Замовлення» (див. рис. 3.4), вікна «Адміністратор» (див. рис. 3.6), та компонентів головного вікна для маніпулюванням над замовленням, в головному меню з’являються додаткові пункти: замовлення та адміністратор. На вікні «Адміністратор» розташовано:

* бокове меню з випадаючим списком кнопок при натисканні на яких, виконується якась дія;
* поле для пошуку даних за різними критеріями;
* кнопки для додавання, редагування або видалення даних;
* компонент для перегляду даних;
* кнопка для формування звітності.

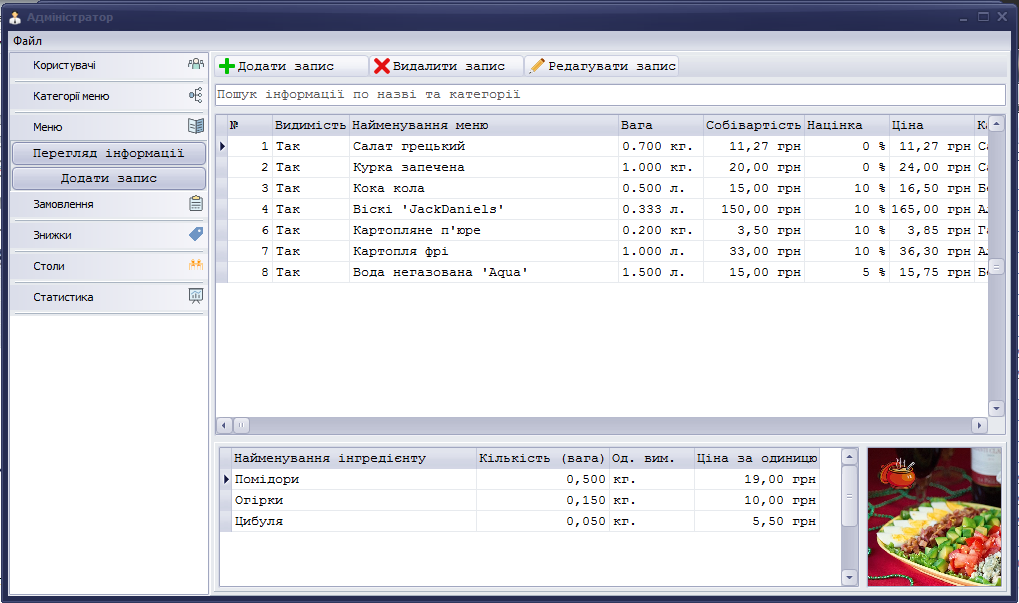


Рисунок 3.6 – Вікно «Адміністратор»

При натисканні на кнопку «Додати запис» відкривається відповідне вікно для додавання нового запису. При натисканні на кнопку «Редагувати запис» відкривається відповідне вікно для редагування вибраного запису (див. рис. 3.7). На даному вікні зображено:

* кнопки підтвердження та відміни виконаних дій;
* компоненти для відображення відповідно до інформації.



Рисунок 3.7 – Вікно «Редагування» після натискання кнопки «Редагувати запис»

При натисканні на кнопку «Видалити запис» відображається вікно з підтвердженням видалення інформації про вибраний запис (див. рис. 3.8).



Рисунок 3.8 – Вікно з підтвердженням дії

Для формування звітності потрібно натиснути на кнопку «Звітність» вибравши відповідний звіт в боковому меню «Статистика» та вибравши період (див. рис. 3.9). Вигляд звіту зображено на рисунку 3.10.

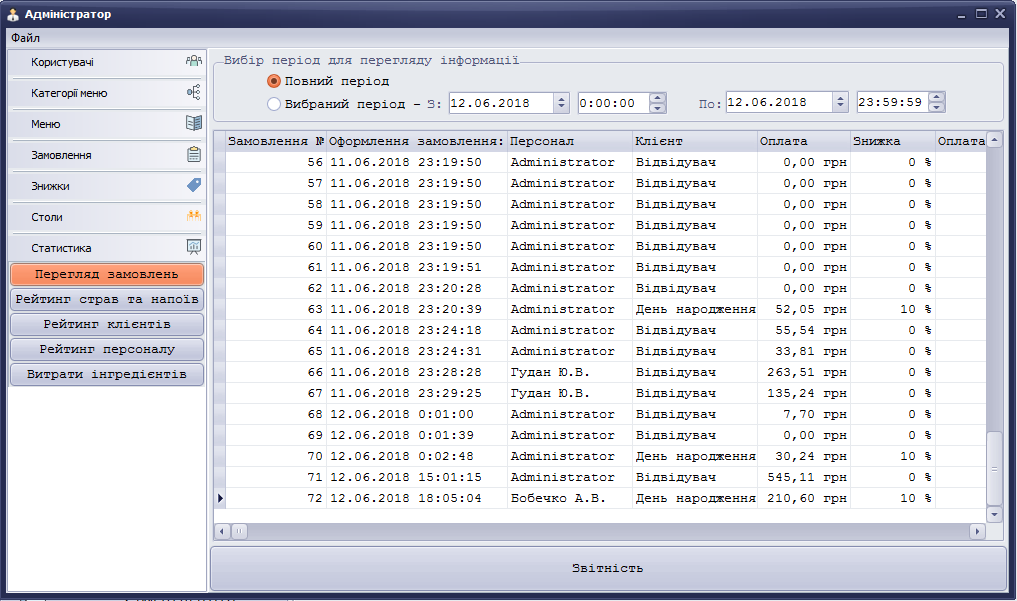


Рисунок 3.10 – Вікно «Адміністратор» після вибору звіту «Перегляд замовлень»

В оформленому звіті, є можливості: збереження, відкриття, та друк звіту.

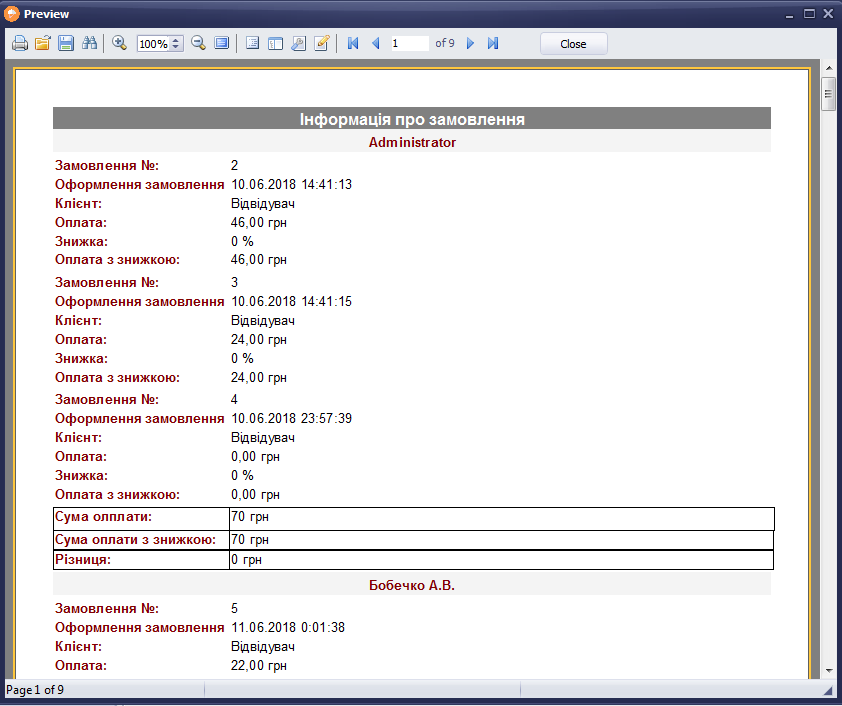


Рисунок 3.10 – Звіт «Перегляд замовлень»

## 3.4 Тестування програмного забезпечення

Проведемо тестування програмного забезпечення, перевіривши якість розроблюваного програмного забезпечення, коректність, повноту, безпечність доступу для адміністратора та користувача. Тестування програмного забезпечення проводилась на операційній системі Windows 7. Для тестування програмного забезпечення перевіримо: коректність авторизації в систему під режимом доступу адміністратор, коректність даних в рахунку замовлення та правильність збереження даних в базі даних.

### 3.4.1 Тестування з режимом доступу адміністратора

Перевіримо коректність входу під режим доступу адміністратор. Для цього треба натиснути пункт головного меню «Авторизація», введемо дані для входу у систему: логін «admin» та пароль «admin». Після авторизації в системі, має з’явитися на головному вікні компоненти для маніпулюванням інформацією про замовлення, та пункти: «Адміністратор» та «Замовлення» в головному меню, також має бути доступ до вікна «Адміністратор» і «Замовлення». Результат авторизації в системі в режимі доступу адміністратор, зображено на рисунку 3.11.



Рисунок 3.11 – Головне вікно в режимі доступу адміністратор

В результаті чого, можемо побачити, що був наданий доступ адміністратора до програмного забезпечення.

### 3.4.2 Тестування з режимом доступу користувач

Перевіримо правильність вмісту рахунку замовлення з оформленим замовленням. Для цього створимо замовлення під іменем користувача «Бобечко А.В.», з вмістом замовлення: «Курка запечена» – 3, та «Салат грецький» – 1. Після створення замовлення треба перевірити рахунок замовлення, чи всі дані збігаються. Результат виконання перевірки зображено на рисунку 3.12.

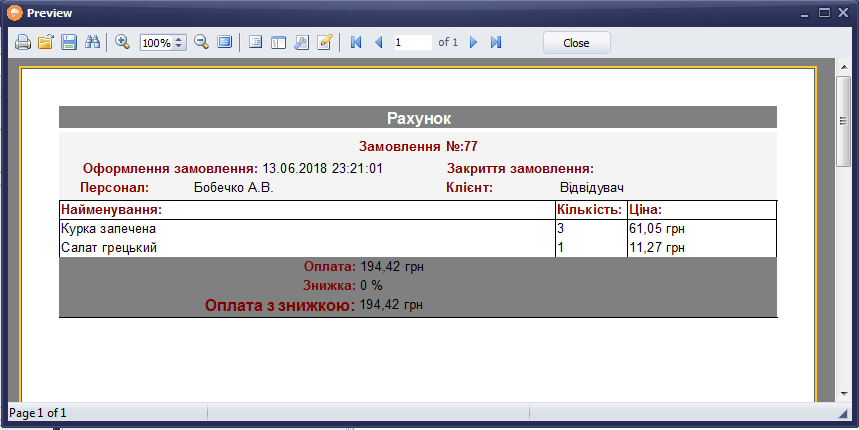


Рисунок 3.12 – Результат рахунку замовлення

В результаті чого можна побачити, що вся інформація збігається.

### 3.4.3 Тестування роботи з базою даних.

Перевіримо правильність збереження нового користувача для програмного забезпечення, для цього введемо інформацію про користувача (див. рис. 3.13) і перевіримо чи дані були збережені в базі даних.

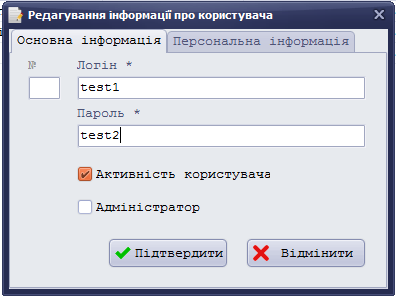


Рисунок 3.13 – Інформація про нового користувача

Після введення даних та збереження, можна переглянути чи дані були збережені в базі даних (див. рис. 3.14).

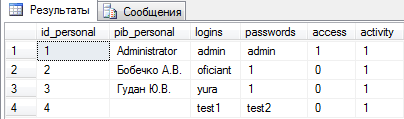


Рисунок 3. 14 – Перегляд інформації про користувачів в базі даних

В результаті чого можна побачити, що всі дані, які були збереженні були коректно додані в базу даних.

Отриманий результат відповідає очікуваному, з чого можна зробити висновок, що програмне забезпечення працює коректно.

В даному розділі було описано середовище розробки програмного забезпечення, інструкції програміста та користувача та проведено тестування програмного забезпечення.

# ВИСНОВКИ

Результатом дипломного проекту є програмне забезпечення «Електронне меню для закладів громадського харчування» під управлінням операційної системи Windows 7 та вище. Наведено детальну інструкцію користувача по експлуатації даної автоматизованої інформаційної-системи, описано методику проведення випробувань, яка показує можливість її введення в експлуатацію.

У результаті розробки програмного забезпечення було ознайомлено із основами розробки під операційну систему Windows, отримано навички розробки програмних забезпечень під різний рівень доступу, навички використання основних компонентів в середовищі розробки для Windows.

Програмне забезпечення було розроблена в середовищі Embarcadero RAD Studio 10.2 Tokyo С++ Builder. Було використано принцип об’єктно-орієнтованого програмування. Структура бази даних була розроблена в Miscrosoft Express SQL Server.

Перевагами такого програмного забезпечення є простота та зручність, ефективність та надійність, вона допоможе закладам громадського харчування поліпшити та пришвидшити обслуговування клієнтів, а клієнтам надасть можливість переглядати інформацію в електронному вигляді. Потенційний попит на створене програмне забезпечення передбачається високим, так як дане програмне забезпечення розраховано для споживання в різних закладах громадського харчування.

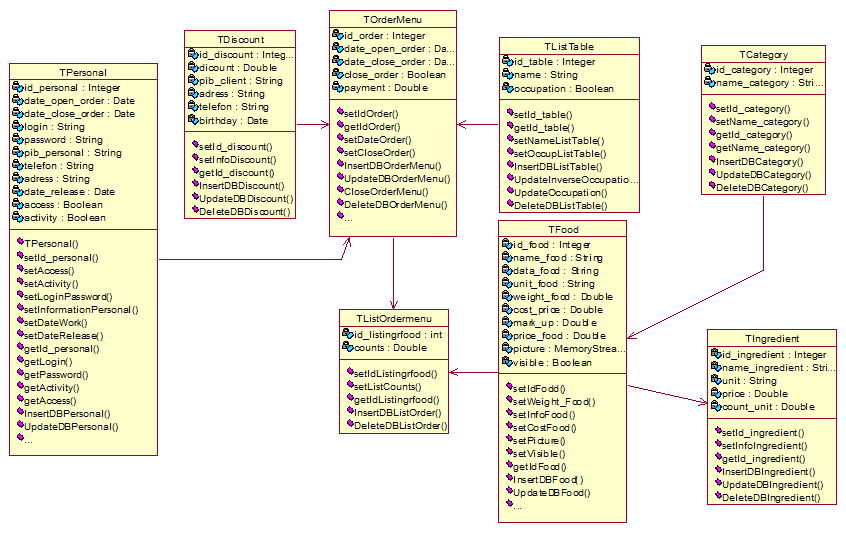
# перелік джерел

1. Страуструп Б. Язык программирования С++, спец.изд. /Пер. с англ – М.: «Издательство БИНОМ», 2002. – 1099с.
2. Буч Г. Объектно-ориентированное проектирование с примерами применения: Пер. с англ. – М.: Конкорд, 2010. – 519с.,ил.
3. Збірка бібліотек для робот из Embarcadero RAD Studio C++ Builder. [Електронний ресурс]. – Режим доступу: http://docwiki.embarcadero.com/
4. Короткий довідник t-sql запитів. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-2017
5. Форум з зразками t-sql та с++ функцій. [Електронний ресурс]. – Режим доступу: http://www.cyberforum.ru
6. Практичне використання реактивного програмування в програмуванні систем. [Електронний ресурс]. – Режим доступу: http://habrahabr.ru/post/140719/.

# Додатки

**Додаток А**

**Діаграма класів**



**Додаток Б**

**Лістинг програмного коду**

Б.1 Вміст файлу eMenuPCH1.h

#include <vcl.h>

#include <tchar.h>

class TPersonal

{

private:

int id\_personal;

AnsiString login,password,pib\_personal,telefon,adress,date\_release;

TDateTime date\_work,birthday;

bool access,activity;

protected:

public:

TPersonaL(){

this->id\_personal = 0;

this->login = "Ëîã³í";

this->password = "Ïàðîëü";

this->pib\_personal = "Ïð³çâèùå ³í³ö³àëè";

this->telefon = "Òåëåôîí";

this->adress = "Àäðåñ";

this->date\_work = Date();

this->date\_release = "";

this->birthday = StrToDate("01.01.1970");

this->access = false;

this->activity = true;

}

void setId\_personal(int);

void setAccess(bool);

void setActivity(bool);

void setLoginPassword(AnsiString,AnsiString);

void setInformationPersonal(AnsiString,AnsiString,AnsiString,TDateTime);

void setDateWork(TDateTime);

void setDateRelease(AnsiString);

int getId\_personal();

AnsiString getLogin();

AnsiString getPassword();

bool getActivity();

bool getAccess();

void InsertDBPersonal();

void UpdateDBPersonal();

void UpdateDBPersonalRelease();

void DeleteDBPersonal();

};

class TListTable

{

private:

int id\_table;

AnsiString name;

bool occupation;

protected:

public:

void setId\_table(int);

int getId\_table();

void setNameListTable(AnsiString);

void setOccupListTable(bool);

void InsertDBListTable();

void UpdateInverseOccupation();

void UpdateOccupation();

void DeleteDBListTable();

};

class TDiscount

{

private:

int id\_discount;

float discount;

String pib\_client,adress,telefon;

TDateTime birthday;

protected:

public:

TDiscount(){

this->id\_discount = 0;

this->discount = 0;

this->pib\_client = "Ï²Á Êë³ºíòà";

this->adress = "Àäðåñ Êë³ºíòà";

this->telefon = "Ìîá³ëüíèé òåëåôîí";

this->birthday = StrToDate("01.01.1970");

}

void setId\_discount(int);

void setInfoDiscount(float,String,String,String,TDateTime);

int getId\_discount();

void InsertDBDiscount();

void UpdateDBDiscount();

void DeleteDBDiscount();

};

class TCategory

{

private:

int id\_category;

AnsiString name\_category;

protected:

public:

TCategory(){

this->id\_category = 0;

this->name\_category = "Name\_category";

}

void setId\_category(int);

void setName\_category(AnsiString);

int getId\_category();

AnsiString getName\_category();

void InsertDBCategory();

void UpdateDBCategory();

void DeleteDBCategory();

};

class TFood: public TCategory

{

private:

int id\_food;

AnsiString name\_food,data\_food,unit\_food;

float weight\_food,cost\_price\_food,mark\_up,price\_food;

TMemoryStream \*picture;

bool visible;

protected:

public:

TFood(){

this->id\_food = 0;

this->name\_food = "Íàçâà";

this->data\_food = "Â³äîìîñò³";

this->unit\_food = "Îäèíèö³ÿ âèì³ðó";

this->weight\_food = 0;

this->cost\_price\_food = 0;

this->mark\_up = 0;

this->price\_food = 0;

this->visible = true;

this->picture = new TMemoryStream;

}

void setIdFood(int);

void setWeigh\_Food(float);

void setInfoFood(AnsiString,AnsiString,AnsiString);

void setCostFood(float,float,float);

void setPicture(TMemoryStream\*);

void setVisible(bool);

int getIdFood();

void InsertDBFood();

void UpdateDBFood();

void UpdateFotoDBFood();

void DeleteDBFood();

};

class TIngredient: public TFood

{

private:

int id\_ingredient;

AnsiString name\_ingredient,unit;

float price,count\_unit;

public:

TIngredient(){

this->id\_ingredient = 0;

this->name\_ingredient = "Íàçâà";

this->unit = "Îä.âèì.";

this->price = 0;

this->count\_unit = 0;

}

void setId\_ingredient(int);

void setInfoIngredient(AnsiString,AnsiString,float,float);

int getId\_ingredient();

void InsertDBIngredient();

void UpdateDBIngredient();

void DeleteDBIngredient();

};

class TOrderMenu: public TDiscount, public TPersonal,public TListTable

{

private:

int id\_order;

TDateTime date\_open\_order,date\_close\_order;

bool close\_order;

float payment;

public:

TOrderMenu(){

this->id\_order = 0;

this->date\_close\_order = Date();

this->date\_close\_order = Date();

this->close\_order = false;

this->payment = 0;

}

void setIdOrder(int);

int getIdOrder();

void setDateOrder(TDateTime,TDateTime);

void setCloseOrder(bool);

void setPayment(float);

void InsertDBOrderMenu();

void UpdateDBOrderMenu();

void DeleteDBOrderMenu();

void CloseOrderMenu();

void UpdatePayment();

void UpdateDiscount();

};

class TListOrder: public TFood,public TOrderMenu

{

private:

int id\_listingrfood;

float counts;

public:

void setIdListingrfood(int);

void setListCounts(float);

int getIdListingrfood();

void InsertDBListOrder();

void DeleteDBListOrder();

};

Б.2 Вміст файлу DataModule.h

#pragma hdrstop

#include "DataModule.h"

#include "Admin.h"

#include "eMenuPCH1.h"

#include "MainForm.h"

#include "Authorization.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditIngredient.h"

#include "EditPersonal.h"

#include "EditOrderMenu.h"

#include "EditTabel.h"

#pragma package(smart\_init)

#pragma classgroup "Vcl.Controls.TControl"

#pragma link "frxClass"

#pragma link "frxDBSet"

#pragma resource "\*.dfm"

TDM \*DM;

\_\_fastcall TDM::TDM(TComponent\* Owner)

: TDataModule(Owner)

{

this->ini = new TIniFile(GetCurrentDir()+"\\options.ini");

try {

ConnectionToDB->Connected = false;

ConnectionToDB->ConnectionString = ini->ReadString("SQL Server","ConnectionString","ConnectionString=Provider=SQLOLEDB.1;Password=1;Persist Security Info=True;User ID=sa;Initial Catalog=eMenu;Data Source=127.0.0.1;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=YURAHUDAN-ÏÊ;Use Encryption for Data=False;Tag with column collation when possible=False");

ConnectionToDB->Connected = true;

OpenDB();

}

catch(...) {

ShowMessage("Íå âäàëîñü ï³äëþ÷èòèñü äî ÁÄ!");

}

}

void TDM::DoSQL(TADOQuery \*Query,AnsiString str){

Query->Close();

Query->SQL->Clear();

Query->SQL->Add(str);

Query->ExecSQL();

Query->Open();

}

void TDM::DoSQLExec(TADOQuery \*Query,AnsiString str){

Query->Close();

Query->SQL->Clear();

Query->SQL->Add(str);

Query->ExecSQL();

}

void TDM::RefreshADO(TADOTable \*Table){

if(Table->Active){

int a = Table->RecNo;

Table->Close();

Table->Open();

Table->RecNo = a;

} else {

Table->Close();

Table->Open();

}

}

void TDM::RefreshADO(TADOQuery \*Query){

if(Query->Active){

int a = Query->RecNo;

Query->Close();

Query->Open();

Query->RecNo = a;

}else{

Query->Close();

Query->Open();

}

}

void TDM::OpenDB(){

try{

RefreshADO(ATPersonal);

RefreshADO(ATCategory);

RefreshADO(ATFood);

RefreshADO(ATDiscount);

RefreshADO(ATFoodIngredient);

RefreshADO(ATListTable);

RefreshADO(ATOrderMenu);

RefreshADO(ATListOrderMenu);

this->OpenReport();

this->OpenDBMain();

this->OpenDBOficiant();

}catch(...){}

}

void TDM::OpenReport(){

RefreshADO(TReportDiscount);

RefreshADO(TReportIngredient);

RefreshADO(TReportFood);

RefreshADO(TReportPersonal);

RefreshADO(TReportOrder);

}

void TDM::OpenDBMain(){

RefreshADO(MTShowCategory);

RefreshADO(MTShowFood);

}

void TDM::OpenDBOficiant(){

OTNotOcupTable->Close();

OTNotOcupTable->Open();

OTOpenOrder->Close();

OTOpenOrder->Open();

OTListOpenOrder->Close();

OTListOpenOrder->Open();

}

void \_\_fastcall TDM::N3Click(TObject \*Sender)

{

FAdmin->Close();

}

void \_\_fastcall TDM::DataModuleDestroy(TObject \*Sender)

{

ini->WriteString("SQL Server","ConnectionString",ConnectionToDB->ConnectionString);

}

void \_\_fastcall TDM::N2Click(TObject \*Sender)

{

DM->OpenDB();

}

void \_\_fastcall TDM::N4Click(TObject \*Sender)

{

FMainForm->Close();

}

void \_\_fastcall TDM::N5Click(TObject \*Sender)

{

FAuth->Show();

}

void \_\_fastcall TDM::N10Click(TObject \*Sender)

{

FMainForm->Close();

}

void \_\_fastcall TDM::N11Click(TObject \*Sender)

{

FAuth->NotUserStatus(Sender);

DM->OpenDB();

}

void \_\_fastcall TDM::N12Click(TObject \*Sender)

{

if (DM->admin && DM->auth) FAdmin->Show();

}

void \_\_fastcall TDM::N15Click(TObject \*Sender)

{

FAuth->Show();

}

void \_\_fastcall TDM::N19Click(TObject \*Sender)

{

DM->OpenDB();

}

void \_\_fastcall TDM::N21Click(TObject \*Sender)

{

FAuth->NotUserStatus(Sender);

DM->OpenDB();

}

void \_\_fastcall TDM::N22Click(TObject \*Sender)

{

FMainForm->Close();

}

void \_\_fastcall TDM::TimerUpdateTimer(TObject \*Sender)

{

// DM->DoSQL(DM->ADOQuery1,"SELECT OBJECT\_NAME(object\_id) AS DatabaseName, last\_user\_update FROM sys.dm\_db\_index\_usage\_stats WHERE (database\_id = DB\_ID('eMenu'))");

// DM->ADOQuery1->First();

// DM->MonitorServer->First();

// bool f = false;

// for (int i = 0; i < DM->ADOQuery1->RecordCount; i++) {

// if (DM->MonitorServer->FieldByName("last\_user\_update")->AsString != DM->ADOQuery1->FieldByName("last\_user\_update")->AsString) f = true;

// DM->ADOQuery1->Next();

// DM->MonitorServer->Next();

// }

// if(f)DM->OpenDB();

}

void \_\_fastcall TDM::N13Click(TObject \*Sender)

{

FEditOrderMenu->Show();

}

void \_\_fastcall TDM::N17Click(TObject \*Sender)

{

FEditOrderMenu->Show();

}

void \_\_fastcall TDM::N8Click(TObject \*Sender)

{

DM->OpenDB();

}

Б.3 Вміст файлу DataModule.cpp

#pragma hdrstop

#include "DataModule.h"

#include "Admin.h"

#include "eMenuPCH1.h"

#include "MainForm.h"

#include "Authorization.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditIngredient.h"

#include "EditPersonal.h"

#include "EditOrderMenu.h"

#include "EditTabel.h"

#pragma package(smart\_init)

#pragma classgroup "Vcl.Controls.TControl"

#pragma link "frxClass"

#pragma link "frxDBSet"

#pragma resource "\*.dfm"

TDM \*DM;

\_\_fastcall TDM::TDM(TComponent\* Owner)

: TDataModule(Owner)

{

this->ini = new TIniFile(GetCurrentDir()+"\\options.ini");

try {

ConnectionToDB->Connected = false;

ConnectionToDB->ConnectionString = ini->ReadString("SQL Server","ConnectionString","ConnectionString=Provider=SQLOLEDB.1;Password=1;Persist Security Info=True;User ID=sa;Initial Catalog=eMenu;Data Source=127.0.0.1;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=YURAHUDAN-ÏÊ;Use Encryption for Data=False;Tag with column collation when possible=False");

ConnectionToDB->Connected = true;

OpenDB();

}

catch(...) {

ShowMessage("Íå âäàëîñü ï³äëþ÷èòèñü äî ÁÄ!");

}

}

void TDM::DoSQL(TADOQuery \*Query,AnsiString str){

Query->Close();

Query->SQL->Clear();

Query->SQL->Add(str);

Query->ExecSQL();

Query->Open();

}

void TDM::DoSQLExec(TADOQuery \*Query,AnsiString str){

Query->Close();

Query->SQL->Clear();

Query->SQL->Add(str);

Query->ExecSQL();

}

void TDM::RefreshADO(TADOTable \*Table){

if(Table->Active){

int a = Table->RecNo;

Table->Close();

Table->Open();

Table->RecNo = a;

} else {

Table->Close();

Table->Open();

}

}

void TDM::RefreshADO(TADOQuery \*Query){

if(Query->Active){

int a = Query->RecNo;

Query->Close();

Query->Open();

Query->RecNo = a;

}else{

Query->Close();

Query->Open();

}

}

void TDM::OpenDB(){

try{

RefreshADO(ATPersonal);

RefreshADO(ATCategory);

RefreshADO(ATFood);

RefreshADO(ATDiscount);

RefreshADO(ATFoodIngredient);

RefreshADO(ATListTable);

RefreshADO(ATOrderMenu);

RefreshADO(ATListOrderMenu);

this->OpenReport();

this->OpenDBMain();

this->OpenDBOficiant();

}catch(...){}

}

void TDM::OpenReport(){

RefreshADO(TReportDiscount);

RefreshADO(TReportIngredient);

RefreshADO(TReportFood);

RefreshADO(TReportPersonal);

RefreshADO(TReportOrder);

}

void TDM::OpenDBMain(){

RefreshADO(MTShowCategory);

RefreshADO(MTShowFood);

}

void TDM::OpenDBOficiant(){

OTNotOcupTable->Close();

OTNotOcupTable->Open();

OTOpenOrder->Close();

OTOpenOrder->Open();

OTListOpenOrder->Close();

OTListOpenOrder->Open();

}

void \_\_fastcall TDM::N3Click(TObject \*Sender)

{

FAdmin->Close();

}

void \_\_fastcall TDM::DataModuleDestroy(TObject \*Sender)

{

ini->WriteString("SQL Server","ConnectionString",ConnectionToDB->ConnectionString);

}

void \_\_fastcall TDM::N2Click(TObject \*Sender)

{

DM->OpenDB();

}

void \_\_fastcall TDM::N4Click(TObject \*Sender)

{

FMainForm->Close();

}

void \_\_fastcall TDM::N5Click(TObject \*Sender)

{

FAuth->Show();

}

void \_\_fastcall TDM::N10Click(TObject \*Sender)

{

FMainForm->Close();

}

void \_\_fastcall TDM::N11Click(TObject \*Sender)

{

FAuth->NotUserStatus(Sender);

DM->OpenDB();

}

void \_\_fastcall TDM::N12Click(TObject \*Sender)

{

if (DM->admin && DM->auth) FAdmin->Show();

}

void \_\_fastcall TDM::N15Click(TObject \*Sender)

{

FAuth->Show();

}

void \_\_fastcall TDM::N19Click(TObject \*Sender)

{

DM->OpenDB();

}

void \_\_fastcall TDM::N21Click(TObject \*Sender)

{

FAuth->NotUserStatus(Sender);

DM->OpenDB();

}

void \_\_fastcall TDM::N22Click(TObject \*Sender)

{

FMainForm->Close();

}

void \_\_fastcall TDM::TimerUpdateTimer(TObject \*Sender)

{

// DM->DoSQL(DM->ADOQuery1,"SELECT OBJECT\_NAME(object\_id) AS DatabaseName, last\_user\_update FROM sys.dm\_db\_index\_usage\_stats WHERE (database\_id = DB\_ID('eMenu'))");

// DM->ADOQuery1->First();

// DM->MonitorServer->First();

// bool f = false;

// for (int i = 0; i < DM->ADOQuery1->RecordCount; i++) {

// if (DM->MonitorServer->FieldByName("last\_user\_update")->AsString != DM->ADOQuery1->FieldByName("last\_user\_update")->AsString) f = true;

// DM->ADOQuery1->Next();

// DM->MonitorServer->Next();

// }

// if(f)DM->OpenDB();

}

void \_\_fastcall TDM::N13Click(TObject \*Sender)

{

FEditOrderMenu->Show();

}

void \_\_fastcall TDM::N17Click(TObject \*Sender)

{

FEditOrderMenu->Show();

}

void \_\_fastcall TDM::N8Click(TObject \*Sender)

{

DM->OpenDB();

}

Б.4 Вміст файлу Admin.cpp

#include <vcl.h>

#pragma hdrstop

#include "Admin.h"

#include "eMenuPCH1.h"

#include "DataModule.h"

#include "MainForm.h"

#include "Authorization.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditPersonal.h"

#include "EditIngredient.h"

#include "EditTabel.h"

#include "EditOrderMenu.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFAdmin \*FAdmin;

\_\_fastcall TFAdmin::TFAdmin(TComponent\* Owner)

: TForm(Owner)

{

PanelPersonal->Align = alClient;

PanelFood->Align = alClient;

PanelCategory->Align = alClient;

PanelDiscount->Align = alClient;

PanelOrderMenu->Align = alClient;

LookPanel->Align = alClient;

PanelTable->Align = alClient;

PanelReport->Align = alClient;

}

void \_\_fastcall TFAdmin::buttonViewPersonalClick(TObject \*Sender)

{

this->HidePanel(Sender);

PanelPersonal->Show();

}

void \_\_fastcall TFAdmin::buttonAddPersonalClick(TObject \*Sender)

{

FEditPersonal->Show();

}

void \_\_fastcall TFAdmin::buttonAddCategoryClick(TObject \*Sender)

{

FEditCategory->Show();

}

void \_\_fastcall TFAdmin::Button8Click(TObject \*Sender)

{

FEditFood->Show();

}

void \_\_fastcall TFAdmin::buttonViewCategoryClick(TObject \*Sender)

{

this->HidePanel(Sender);

PanelCategory->Show();

}

void \_\_fastcall TFAdmin::buttonViewFoodClick(TObject \*Sender)

{

this->HidePanel(Sender);

PanelFood->Show();

}

void \_\_fastcall TFAdmin::buttonViewOrderMenuClick(TObject \*Sender)

{

this->HidePanel(Sender);

PanelOrderMenu->Show();

}

void \_\_fastcall TFAdmin::buttonViewDiscountClick(TObject \*Sender)

{

this->HidePanel(Sender);

PanelDiscount->Show();

}

void TFAdmin::HidePanel(TObject \*Sender){

PanelDiscount->Hide();

PanelPersonal->Hide();

PanelFood->Hide();

PanelCategory->Hide();

PanelOrderMenu->Hide();

LookPanel->Hide();

PanelTable->Hide();

PanelReport->Hide();

}

void \_\_fastcall TFAdmin::DeleteCategoyClick(TObject \*Sender)

{

try {

if (IDYES == MessageBox(NULL, L"Âè âïåâíåí³, ùî õî÷åòå âèäàëèòè ?", L"Ï³äâåðäæåííÿ!", MB\_YESNO |MB\_ICONQUESTION))

{

TCategory category;

category.setId\_category(DM->ATCategory->FieldByName("id\_category")->AsInteger);

category.DeleteDBCategory();

DM->OpenDB();

}

}

catch(...){

MessageBox(NULL, L"Äàí³, ïîâ'ÿçàíí³ ç äàíîþ êàòåãîð³þº. Íå ìîæëèâî âèäàëèòè!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFAdmin::AddCategoryClick(TObject \*Sender)

{

FEditCategory->Show();

}

void \_\_fastcall TFAdmin::EditCategoryClick(TObject \*Sender)

{

FEditCategory->Show();

FEditCategory->EditID->Text = DM->ATCategory->FieldByName("id\_category")->AsAnsiString;

FEditCategory->edit\_name\_category->Text = DM->ATCategory->FieldByName("name\_category")->AsAnsiString;

}

void \_\_fastcall TFAdmin::EditSearchCategoryChange(TObject \*Sender)

{

AnsiString tmp = EditSearchCategory->Text;

DM->ATCategory->Filtered = false;

if (tmp.Length()!=0){

DM->ATCategory->Filter = "name\_category like '%"+tmp+"%'";

DM->ATCategory->Filtered = true;

}

}

void \_\_fastcall TFAdmin::Button2Click(TObject \*Sender)

{

FEditDiscount->Show();

FEditDiscount->add = true;

}

void \_\_fastcall TFAdmin::AddDiscountClick(TObject \*Sender)

{

FEditDiscount->Show();

FEditDiscount->add = true;

}

void \_\_fastcall TFAdmin::EditDiscountClick(TObject \*Sender)

{

FEditDiscount->Show();

FEditDiscount->add = false;

FEditDiscount->edit\_id->Text = DM->ATDiscountid\_discount->Value;

FEditDiscount->edit\_pib\_client->Text = DM->ATDiscountpib\_client->Value;

FEditDiscount->edit\_discount->Text = DM->ATDiscountdiscount->Value;

FEditDiscount->edit\_telefon->Text = DM->ATDiscounttelefon->Value;

FEditDiscount->edit\_adress->Text = DM->ATDiscountadress->Value;

FEditDiscount->date\_birthday->Date = DM->ATDiscount->FieldByName("birthday")->AsDateTime;

FEditDiscount->Button1->Hide();

}

void \_\_fastcall TFAdmin::DeleteDiscountClick(TObject \*Sender)

{

try {

if (IDYES == MessageBox(NULL, L"Âè âïåâíåí³, ùî õî÷åòå âèäàëèòè ?", L"Ï³äâåðäæåííÿ!", MB\_YESNO |MB\_ICONQUESTION))

{

TDiscount discount;

discount.setId\_discount(DM->ATDiscountid\_discount->Value);

discount.DeleteDBDiscount();

DM->OpenDB();

}

}

catch(...){

MessageBox(NULL, L"Äàí³, ïîâ'ÿçàíí³ ç äàíîþ êàòåãîð³þº. Íå ìîæëèâî âèäàëèòè!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFAdmin::EditPersonalClick(TObject \*Sender)

{

FEditPersonal->Show();

FEditPersonal->add = false;

FEditPersonal->ButtonRelease->Show();

FEditPersonal->edit\_date\_release->Show();

FEditPersonal->Label3->Show();

FEditPersonal->EditID->Text = DM->ATPersonal->FieldByName("id\_personal")->AsAnsiString;

FEditPersonal->edit\_login->Text = DM->ATPersonal->FieldByName("logins")->AsAnsiString;

FEditPersonal->edit\_password->Text = DM->ATPersonal->FieldByName("passwords")->AsAnsiString;

FEditPersonal->edit\_pib\_personal->Text = DM->ATPersonal->FieldByName("pib\_personal")->AsAnsiString;

FEditPersonal->edit\_telefon->Text = DM->ATPersonal->FieldByName("telefon")->AsAnsiString;

FEditPersonal->PersonalAccess->Checked = DM->ATPersonal->FieldByName("access")->AsBoolean;

FEditPersonal->PersonalActivity->Checked = DM->ATPersonal->FieldByName("activity")->AsBoolean;

FEditPersonal->date\_work->Date = DM->ATPersonal->FieldByName("data\_of\_work")->AsDateTime;

FEditPersonal->date\_birthday->Date = DM->ATPersonal->FieldByName("birthday")->AsDateTime;

FEditPersonal->edit\_date\_release->Text = DM->ATPersonal->FieldByName("release\_date")->AsAnsiString;

FEditPersonal->edit\_adress->Text = DM->ATPersonal->FieldByName("adress")->AsAnsiString;

}

void \_\_fastcall TFAdmin::AddPersonaClick(TObject \*Sender)

{

FEditPersonal->Show();

}

void \_\_fastcall TFAdmin::EditSearchPersonalChange(TObject \*Sender)

{

AnsiString tmp = EditSearchPersonal->Text;

DM->ATPersonal->Filtered = false;

if (tmp.Length()!=0){

DM->ATPersonal->Filter = "logins like '%"+tmp+"%' or pib\_personal like '%"+tmp+"%'";

DM->ATPersonal->Filtered = true;

}

}

void \_\_fastcall TFAdmin::DeletePersonalClick(TObject \*Sender)

{

try {

if (IDYES == MessageBox(NULL, L"Âè âïåâíåí³, ùî õî÷åòå âèäàëèòè ?", L"Ï³äâåðäæåííÿ!", MB\_YESNO |MB\_ICONQUESTION))

{

if(DM->ATPersonal->FieldByName("id\_personal")->AsInteger != 1){

TPersonal personal;

personal.setId\_personal(DM->ATPersonal->FieldByName("id\_personal")->AsInteger);

personal.DeleteDBPersonal();

DM->OpenDB();

} else MessageBox(NULL, L"Âèäàëåííÿ ñòàíäàðòíîãî àäì³í³ñòðàòîðà çàáîðîíåíî!", L"Â³äìîâà!", MB\_OK |MB\_ICONERROR);

}

}

catch(...){

MessageBox(NULL, L"Äàí³, ïîâ'ÿçàíí³ ç äàíîþ êàòåãîð³þº. Íå ìîæëèâî âèäàëèòè!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFAdmin::N4Click(TObject \*Sender)

{

FAuth->NotUserStatus(Sender);

}

void \_\_fastcall TFAdmin::N5Click(TObject \*Sender)

{

FMainForm->Close();

}

void \_\_fastcall TFAdmin::N2Click(TObject \*Sender)

{

DM->OpenDB();

}

void \_\_fastcall TFAdmin::AddFoodClick(TObject \*Sender)

{

FEditFood->Show();

}

void \_\_fastcall TFAdmin::EditFoodClick(TObject \*Sender)

{

FEditFood->Show();

FEditFood->add = false;

FEditFood->CB\_id\_category->Text = DM->ATFood->FieldByName("kod\_category")->AsString;

FEditFood->CB\_name\_category->Text = DM->ATFood->FieldByName("name\_category")->AsString;

FEditFood->CB\_unit->Text = DM->ATFood->FieldByName("unit\_food")->AsString;

FEditFood->EditID->Text = DM->ATFood->FieldByName("id\_food")->AsString;

FEditFood->Edit\_cost\_price->Text = DM->ATFood->FieldByName("cost\_price\_food")->AsString;

FEditFood->Edit\_mark\_up->Text = DM->ATFood->FieldByName("mark\_up")->AsString;

FEditFood->Edit\_price->Text = DM->ATFood->FieldByName("price\_food")->AsString;

FEditFood->Edit\_name\_food->Text = DM->ATFood->FieldByName("name\_food")->AsString;

FEditFood->Edit\_weight->Text = DM->ATFood->FieldByName("weight\_food")->AsString;

FEditFood->CBvisible->Checked = DM->ATFood->FieldByName("visible")->AsBoolean;

FEditFood->Image1->Picture->Assign(DM->ATFood->FieldByName("picture"));

}

void \_\_fastcall TFAdmin::DeleteFoodClick(TObject \*Sender)

{

try {

if (IDYES == MessageBox(NULL, L"Âè âïåâíåí³, ùî õî÷åòå âèäàëèòè ?", L"Ï³äâåðäæåííÿ!", MB\_YESNO |MB\_ICONQUESTION))

{

TFood food;

food.setIdFood(DM->ATFoodid\_food->Value);

food.DeleteDBFood();

DM->OpenDB();

}

}

catch(...){

MessageBox(NULL, L"Äàí³, ïîâ'ÿçàíí³ ç äàíîþ ³íôîðìàö³ºþ. Íå ìîæëèâî âèäàëèòè!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFAdmin::EditSearchFoodChange(TObject \*Sender)

{

AnsiString tmp = EditSearchFood->Text;

DM->ATFood->Filtered = false;

if (tmp.Length()!=0){

DM->ATFood->Filter = "name\_food like '%"+tmp+"%' or name\_category like '%"+tmp+"%'";

DM->ATFood->Filtered = true;

}

}

void \_\_fastcall TFAdmin::ButtonViewTableClick(TObject \*Sender)

{

this->HidePanel(Sender);

PanelTable->Show();

}

void \_\_fastcall TFAdmin::EditSearchTableChange(TObject \*Sender)

{

AnsiString tmp = EditSearchTable->Text;

DM->ATListTable->Filtered = false;

if (tmp.Length()!=0){

DM->ATListTable->Filter = "name\_table like '%"+tmp+"%'";

DM->ATListTable->Filtered = true;

}

}

void \_\_fastcall TFAdmin::AddTableClick(TObject \*Sender)

{

FEditTable->Show();

}

void \_\_fastcall TFAdmin::DeleteTableClick(TObject \*Sender)

{

try {

if (IDYES == MessageBox(NULL, L"Âè âïåâíåí³, ùî õî÷åòå âèäàëèòè ?", L"Ï³äâåðäæåííÿ!", MB\_YESNO |MB\_ICONQUESTION))

{

TListTable table;

table.setId\_table(DM->ATListTable->FieldByName("id\_table")->AsInteger);

table.DeleteDBListTable();

DM->OpenDB();

}

}

catch(...){

MessageBox(NULL, L"Äàí³, ïîâ'ÿçàíí³ ç äàíîþ ³íôîðìàö³ºþ. Íå ìîæëèâî âèäàëèòè!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFAdmin::ButtonAddListTableClick(TObject \*Sender)

{

FEditTable->Show();

}

void \_\_fastcall TFAdmin::InvOcupTableClick(TObject \*Sender)

{

TListTable table;

table.setId\_table(DM->ATListTable->FieldByName("id\_table")->AsInteger);

table.UpdateInverseOccupation();

DM->RefreshADO(DM->ATListTable);

}

void \_\_fastcall TFAdmin::N7Click(TObject \*Sender)

{

FEditOrderMenu->Show();

}

void \_\_fastcall TFAdmin::AddOrderMenuClick(TObject \*Sender)

{

FEditOrderMenu->Show();

}

void \_\_fastcall TFAdmin::DeleteOrderMenuClick(TObject \*Sender)

{

try {

if (IDYES == MessageBox(NULL, L"Âè âïåâíåí³, ùî õî÷åòå âèäàëèòè ?", L"Ï³äâåðäæåííÿ!", MB\_YESNO |MB\_ICONQUESTION))

{

TOrderMenu order;

order.setIdOrder(DM->ATOrderMenu->FieldByName("id\_order")->AsInteger);

order.DeleteDBOrderMenu();

DM->OpenDB();

}

}

catch(...){

MessageBox(NULL, L"Äàí³, ïîâ'ÿçàíí³ ç äàíîþ ³íôîðìàö³ºþ. Íå ìîæëèâî âèäàëèòè!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFAdmin::ButtonDayReportClick(TObject \*Sender)

{

DM->DoSQL(DM->TSQL,"SELECT SUM(payment - (payment \* discount/100)) AS DayPayment \

FROM ((OrderMenu INNER JOIN Discounts ON id\_discount = kod\_discount)\

INNER JOIN Personal ON id\_personal = kod\_personal)\

INNER JOIN ListTable ON id\_table = kod\_table \

WHERE date\_open\_order >= '"+DateToStr(Date())+" 00:00:00' and date\_open\_order <= '"+DateToStr(Date())+" 23:59:59'");

ShowMessage("Äåíèé çàðîá³òîê - "+DM->TSQL->FieldByName("DayPayment")->AsString+" ãðí.");

}

void \_\_fastcall TFAdmin::buttonViewListOrderMenuClick(TObject \*Sender)

{

FEditOrderMenu->Show();

}

void \_\_fastcall TFAdmin::FormShow(TObject \*Sender)

{

DateFirst->Date = Date();

DateSecond->Date = Date();

TimeFirst->Time = StrToTime("0:00:00");

TimeSecond->Time = StrToTime("23:59:59");

}

void \_\_fastcall TFAdmin::TimeFirstChange(TObject \*Sender)

{

DateFirst->Time = TimeFirst->Time;

if (RBperiod->Checked) RBperiodClick(Sender);

}

void \_\_fastcall TFAdmin::DateFirstChange(TObject \*Sender)

{

DateFirst->Time = TimeFirst->Time;

if (RBperiod->Checked) RBperiodClick(Sender);

}

void \_\_fastcall TFAdmin::DateSecondChange(TObject \*Sender)

{

DateSecond->Time = TimeSecond->Time;

if (RBperiod->Checked) RBperiodClick(Sender);

}

void \_\_fastcall TFAdmin::TimeSecondChange(TObject \*Sender)

{

DateSecond->Time = TimeSecond->Time;

if (RBperiod->Checked) RBperiodClick(Sender);

}

void \_\_fastcall TFAdmin::btnSaleIngredientClick(TObject \*Sender)

{

this->HidePanel(Sender);

DM->DSReport->DataSet = DM->TReportIngredient;

DM->OpenReport();

PanelReport->Show();

}

void \_\_fastcall TFAdmin::btnScoreDiscountClick(TObject \*Sender)

{

this->HidePanel(Sender);

DM->DSReport->DataSet = DM->TReportDiscount;

DM->OpenReport();

PanelReport->Show();

}

void \_\_fastcall TFAdmin::btnScorePersonalClick(TObject \*Sender)

{

this->HidePanel(Sender);

DM->DSReport->DataSet = DM->TReportPersonal;

DM->OpenReport();

PanelReport->Show();

}

void \_\_fastcall TFAdmin::btnShowOrderClick(TObject \*Sender)

{

this->HidePanel(Sender);

PanelReport->Show();

DM->DSReport->DataSet = DM->TReportOrder;

DM->OpenReport();

}

void \_\_fastcall TFAdmin::btnScoreFoodClick(TObject \*Sender)

{

this->HidePanel(Sender);

DM->DSReport->DataSet = DM->TReportFood;

DM->OpenReport();

PanelReport->Show();

}

void \_\_fastcall TFAdmin::btnReportClick(TObject \*Sender)

{

String str = DM->DSReport->DataSet->Name;

if (str == "TReportDiscount") { DM->frxScoreDiscount->ShowReport();}

if (str == "TReportFood") { DM->frxScoreFood->ShowReport(); }

if (str == "TReportIngredient"){ DM->frxSaleIngredient->ShowReport();}

if (str == "TReportPersonal") { DM->frxScorePersonal->ShowReport();}

if (str == "TReportOrder") {DM->frxShowOrder->ShowReport();}

}

void \_\_fastcall TFAdmin::RBallClick(TObject \*Sender)

{

String str = DM->DSReport->DataSet->Name;

if (str == "TReportDiscount") {

DM->DoSQL(DM->TReportDiscount,"SELECT id\_discount,pib\_client,count(id\_order) AS CountBuyOrder,sum(payment - (payment\*discount/100)) AS PaymentAll FROM Discounts INNER JOIN OrderMenu ON kod\_discount = id\_discount GROUP BY id\_discount,pib\_client");

}

if (str == "TReportFood") {

DM->DoSQL(DM->TReportFood,"SELECT name\_food,count(counts) AS CountSale FROM (Food INNER JOIN ListOrderMenu ON kod\_food = id\_food) INNER JOIN OrderMenu ON id\_order = kod\_order GROUP BY name\_food ORDER BY CountSale desc");

}

if (str == "TReportIngredient");{

DM->DoSQL(DM->TReportIngredient,"SELECT name\_food,name\_ingredient,price, sum(ListOrderMenu.counts \* ListIngredientFood.counts) AsCountSaleIngredient,unit,sum(ListIngredientFood.price \* ListOrderMenu.counts) AS SaleFood FROM ((OrderMenu INNER JOIN ListOrderMenu ON id\_order = kod\_order) INNER JOIN Food ON id\_food = ListOrderMenu.kod\_food )INNER JOIN ListIngredientFood ON ListIngredientFood.kod\_food = id\_food GROUP BY name\_food,name\_ingredient,unit,price");

}

if (str == "TReportPersonal") {

DM->DoSQL(DM->TReportPersonal,"SELECT pib\_personal,count(id\_order) AS CountOrder,sum(payment) AS Payment, sum(payment-(payment\*discount/100)) AS PaymentDiscount FROM (Personal INNER JOIN OrderMenu ON id\_personal = kod\_personal) INNER JOIN Discounts ON kod\_discount = id\_discount GROUP BY pib\_personal ORDER BY CountOrder desc");

}

if (str == "TReportOrder") {

DM->DoSQL(DM->TReportOrder,"SELECT id\_order,date\_open\_order,pib\_personal,pib\_client,payment,discount,sum(payment-(payment\*discount/100)) AS PaymentDiscount FROM ((OrderMenu INNER JOIN Personal ON id\_personal = kod\_personal) INNER JOIN Discounts ON id\_discount = kod\_discount) INNER JOIN ListTable ON id\_table = kod\_table GROUP BY id\_order,date\_open\_order,pib\_personal,pib\_client,payment,discount");

}

}

void \_\_fastcall TFAdmin::RBperiodClick(TObject \*Sender)

{

String str = DM->DSReport->DataSet->Name;

String dt = "WHERE date\_open\_order >= '"+DateToStr(DateFirst->Date)+" "+TimeToStr(TimeFirst->Time)+"' \

and date\_open\_order <= '"+DateToStr(DateSecond->Date)+" "+TimeToStr(TimeSecond->Time)+"'";

if (str == "TReportDiscount") {

DM->DoSQL(DM->TReportDiscount,"SELECT id\_discount,pib\_client,count(id\_order) AS CountBuyOrder,sum(payment - (payment\*discount/100)) AS PaymentAll FROM Discounts INNER JOIN OrderMenu ON kod\_discount = id\_discount\

"+dt+" GROUP BY id\_discount,pib\_client");

}

if (str == "TReportFood") {

DM->DoSQL(DM->TReportFood,"SELECT name\_food,count(counts) AS CountSale FROM (Food INNER JOIN ListOrderMenu ON kod\_food = id\_food) INNER JOIN OrderMenu ON id\_order = kod\_order \

"+dt+" GROUP BY name\_food ORDER BY CountSale desc");

}

if (str == "TReportIngredient");{

DM->DoSQL(DM->TReportIngredient,"SELECT name\_food,name\_ingredient,price, sum(ListOrderMenu.counts \* ListIngredientFood.counts) AsCountSaleIngredient,unit,sum(ListIngredientFood.price \* ListOrderMenu.counts) AS SaleFood FROM ((OrderMenu INNER JOIN ListOrderMenu ON id\_order = kod\_order) INNER JOIN Food ON id\_food = ListOrderMenu.kod\_food )INNER JOIN ListIngredientFood ON ListIngredientFood.kod\_food = id\_food \

"+dt+" GROUP BY name\_food,name\_ingredient,unit,price");

}

if (str == "TReportPersonal") {

DM->DoSQL(DM->TReportPersonal,"SELECT pib\_personal,count(id\_order) AS CountOrder,sum(payment) AS Payment, sum(payment-(payment\*discount/100)) AS PaymentDiscount FROM (Personal INNER JOIN OrderMenu ON id\_personal = kod\_personal) INNER JOIN Discounts ON kod\_discount = id\_discount \

"+dt+" GROUP BY pib\_personal ORDER BY CountOrder desc");

}

if (str == "TReportOrder") {

DM->DoSQL(DM->TReportOrder,"SELECT id\_order,date\_open\_order,pib\_personal,pib\_client,payment,discount,sum(payment-(payment\*discount/100)) AS PaymentDiscount FROM ((OrderMenu INNER JOIN Personal ON id\_personal = kod\_personal) INNER JOIN Discounts ON id\_discount = kod\_discount) INNER JOIN ListTable ON id\_table = kod\_table \

"+dt+" GROUP BY id\_order,date\_open\_order,pib\_personal,pib\_client,payment,discount");

}

}

Б.5 Вміст файлу Authorization.cpp

#include <vcl.h>

#pragma hdrstop

#include "Authorization.h"

#include "Admin.h"

#include "DataModule.h"

#include "eMenuPCH1.h"

#include "MainForm.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditIngredient.h"

#include "EditPersonal.h"

#include "EditOrderMenu.h"

#include "EditTabel.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFAuth \*FAuth;

\_\_fastcall TFAuth::TFAuth(TComponent\* Owner)

: TForm(Owner)

{

DM->admin = false;

DM->auth = false;

this->trycount = 0;

}

void TFAuth::AdminStatus(TObject \*Sender){

DM->admin = true;

FMainForm->Menu = DM->MAdmin;

FMainForm->ShowOficiant();

}

void TFAuth::UserStatus(TObject \*Sender){

DM->admin = false;

FMainForm->Menu = DM->MUser;

FMainForm->ShowOficiant();

}

void TFAuth::NotUserStatus(TObject \*Sender){

FAdmin->Hide();

FEditCategory->Hide();

FEditDiscount->Hide();

FEditFood->Hide();

FEditIngredient->Hide();

FEditPersonal->Hide();

FMainForm->Menu = DM->MMain;

FMainForm->HideOficiant();

FEditOrderMenu->Hide();

FEditTable->Hide();

}

void \_\_fastcall TFAuth::FormShow(TObject \*Sender)

{

Login->SetFocus();

Login->Clear();

Password->Clear();

NotUserStatus(Sender);

this->trycount = 0;

}

void TFAuth::LockedUser(String log){

if (this->trycount > 3) {

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.Personal Set activity = 0 where logins = :log and id\_personal <> 1");

DM->TSQL->Parameters->ParamByName("log")->Value = log;

DM->TSQL->ExecSQL();

DM->OpenDB();

MessageBox(NULL, L"Âàø àêàóíò áóâ çàáëîêîâàíèé!", L"Áëîêóâàííÿ!", MB\_OK |MB\_ICONERROR);

this->trycount = 0;

}

}

void \_\_fastcall TFAuth::buttonOpenClick(TObject \*Sender)

{

String login = Login->Text, password = Password->Text;

DM->DoSQL(DM->TSQL,"Select \* FROM eMenu.dbo.Personal WHERE logins = "+QuotedStr(login)+" and passwords = "+QuotedStr(password));

if (DM->TSQL->RecordCount !=0) {

DM->auth = true;

if(DM->TSQL->FieldByName("activity")->AsBoolean){

if (DM->TSQL->FieldByName("access")->AsBoolean) this->AdminStatus(Sender);

else this->UserStatus(Sender);

FAuth->Hide();

DM->id\_user = DM->TSQL->FieldByName("id\_personal")->AsInteger;

}

else MessageBox(NULL, L"Êîðèñòóâà÷ çàáëîêîâàíèé!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

else {

MessageBox(NULL, L"Íåïðàâèëüíèé ëîã³í àáî ïàðîëü!", L"Â³äìîâà!", MB\_OK |MB\_ICONERROR);

this->trycount++;

this->LockedUser(login);

}

DM->RefreshADO(DM->MTShowFood);

}

void \_\_fastcall TFAuth::LoginKeyPress(TObject \*Sender, System::WideChar &Key)

{

if(Key == 13) Password->SetFocus();

}

void \_\_fastcall TFAuth::PasswordKeyPress(TObject \*Sender, System::WideChar &Key)

{

if(Key == 13) buttonOpenClick(Sender);

}

void \_\_fastcall TFAuth::buttonCloseClick(TObject \*Sender)

{

FAuth->Hide();

}

Б.6 Вміст файлуMainForm.cpp

#include <vcl.h>

#pragma hdrstop

#include "MainForm.h"

#include "Admin.h"

#include "DataModule.h"

#include "Authorization.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditIngredient.h"

#include "EditPersonal.h"

#include "eMenuPCH1.h"

#include "EditOrderMenu.h"

#include "EditTabel.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFMainForm \*FMainForm;

\_\_fastcall TFMainForm::TFMainForm(TComponent\* Owner)

: TForm(Owner)

{

this->HideOficiant();

this->Label2->StyleElements = TStyleElements();

}

void TFMainForm::HideOficiant(){

this->DBGridCategory->RowCount = 15;

this->PanelOrderMenu->Hide();

this->AddToOrder->Hide();

}

void TFMainForm::ShowOficiant(){

this->DBGridCategory->RowCount = 7;

this->PanelOrderMenu->Show();

this->AddToOrder->Show();

}

void \_\_fastcall TFMainForm::EditMainSearchFoodChange(TObject \*Sender)

{

String tmp = EditMainSearchFood->Text;

if (tmp.Length()!=0) {

DM->DoSQL(DM->MQShowFood,"SELECT \*, STR(weight\_food, 5, 3)+' '+ unit\_food AS weight\_unit FROM eMenu.dbo.Food WHERE name\_food like '%"+tmp+"%'");

DBCGShowFood->DataSource = DM->MDSQShowFood;

} else {

DBCGShowFood->DataSource = DM->MDSShowFood;

DM->MTShowFood->Refresh();

DM->MQShowFood->Close();

}

}

void \_\_fastcall TFMainForm::AddToOrderMouseDown(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

AddToOrder->Color = RGB(239,118,68);

AddToOrder->Font->Color = clBtnFace;

}

void \_\_fastcall TFMainForm::AddToOrderMouseUp(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

AddToOrder->Color = clBtnFace;

AddToOrder->Font->Color = RGB(78,84,118);

}

void \_\_fastcall TFMainForm::Image1MouseDown(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

AddToOrder->BevelOuter = bvLowered;

}

void \_\_fastcall TFMainForm::Image1MouseUp(TObject \*Sender, TMouseButton Button, TShiftState Shift,

int X, int Y)

{

AddToOrder->BevelOuter = bvRaised;

}

void \_\_fastcall TFMainForm::Button1Click(TObject \*Sender)

{

TOrderMenu order;

order.InsertDBOrderMenu();

}

void \_\_fastcall TFMainForm::ButtonViewOrderClick(TObject \*Sender)

{

FEditOrderMenu->Show();

}

void \_\_fastcall TFMainForm::Image1Click(TObject \*Sender)

{

try{

String count;

if(DM->OTOpenOrder->RecordCount == 0)

MessageBox(NULL, L"Ñòâîð³òü çàìîâëåííÿ!", L"Â³äìîâà!", MB\_OK | MB\_ICONWARNING);

else

if(InputQuery("Âèáåð³òü ê³ëüê³ñòü!","Ê³ëüê³ñòü",count)){

TListOrder list;

list.setIdFood(DM->MTShowFood->FieldByName("id\_food")->AsInteger);

list.setIdOrder(DM->OTOpenOrder->FieldByName("id\_order")->AsInteger);

list.setListCounts(StrToFloat(count));

list.InsertDBListOrder();

DM->OpenDBOficiant();

list.UpdatePayment();

DM->OpenDBOficiant();

}

}

catch(...){

MessageBox(NULL, L"Ïåðåâ³ðüòå ïðàâèëüí³ñòü äàíèõ!", L"Â³äìîâà!", MB\_OK | MB\_ICONWARNING);

}

}

Б.7 Вміст файлу EditCategory.cpp

#include <vcl.h>

#pragma hdrstop

#include "EditCategory.h"

#include "DataModule.h"

#include "eMenuPCH1.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFEditCategory \*FEditCategory;

\_\_fastcall TFEditCategory::TFEditCategory(TComponent\* Owner)

: TForm(Owner)

{

}

void TCategory::setId\_category(int a){

this->id\_category = a;

}

void TCategory::setName\_category(AnsiString a){

this->name\_category =a;

}

int TCategory::getId\_category(){

return this->id\_category;

}

AnsiString TCategory::getName\_category(){

return this->name\_category;

}

void TCategory::InsertDBCategory(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("INSERT INTO eMenu.dbo.Category(name\_category)");

DM->TSQL->SQL->Add("VALUES(:P1)");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->name\_category;

DM->TSQL->ExecSQL();

}

void TCategory::UpdateDBCategory(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.Category SET name\_category = :P1 WHERE id\_category = :id");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->name\_category;

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_category;

DM->TSQL->ExecSQL();

}

void TCategory::DeleteDBCategory(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("DELETE FROM eMenu.dbo.Category");

DM->TSQL->SQL->Add("WHERE id\_category = :id");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_category;

DM->TSQL->ExecSQL();

}

void \_\_fastcall TFEditCategory::ButtonAccceptClick(TObject \*Sender)

{

TCategory category;

category.setName\_category(FEditCategory->edit\_name\_category->Text);

if (FEditCategory->EditID->Text.Length() != 0) {

category.setId\_category(StrToInt(FEditCategory->EditID->Text));

category.UpdateDBCategory();

} else category.InsertDBCategory();

DM->OpenDB();

Hide();

}

void \_\_fastcall TFEditCategory::FormShow(TObject \*Sender)

{

EditID->Clear();

edit\_name\_category->Clear();

}

void \_\_fastcall TFEditCategory::ButtonCancelClick(TObject \*Sender)

{

Hide();

}

Б.8 Вміст файлу EditDiscount.cpp

#include <vcl.h>

#pragma hdrstop

#include "EditDiscount.h"

#include "DataModule.h"

#include "eMenuPCH1.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFEditDiscount \*FEditDiscount;

\_\_fastcall TFEditDiscount::TFEditDiscount(TComponent\* Owner)

: TForm(Owner)

{

this->add = true;

}

//-----------------------------------------------------------

void TDiscount::setId\_discount(int a){

this->id\_discount = a;

}

void TDiscount::setInfoDiscount(float disc,String pib,String adres,String telefon,TDateTime birth){

this->discount = disc;

this->pib\_client = pib;

this->adress = adres;

this->telefon = telefon;

this->birthday = birth;

}

int TDiscount::getId\_discount(){

return this->id\_discount;

}

void TDiscount::InsertDBDiscount(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("INSERT INTO eMenu.dbo.Discounts(id\_discount,pib\_client,discount,birthday,adress,telefon)");

DM->TSQL->SQL->Add("VALUES(:id,:pib,:disc,:birt,:adre,:telef)");

DM->TSQL->Parameters->ParamByName("pib")->Value = this->pib\_client;

DM->TSQL->Parameters->ParamByName("disc")->Value = this->discount;

DM->TSQL->Parameters->ParamByName("birt")->Value = this->birthday;

DM->TSQL->Parameters->ParamByName("adre")->Value = this->adress;

DM->TSQL->Parameters->ParamByName("telef")->Value = this->telefon;

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_discount;

DM->TSQL->ExecSQL();

}

void TDiscount::UpdateDBDiscount(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.Discounts SET pib\_client = :pib,discount = :disc,birthday = :birt,adress = :adre,telefon = :telef");

DM->TSQL->SQL->Add("WHERE id\_discount = :id");

DM->TSQL->Parameters->ParamByName("pib")->Value = this->pib\_client;

DM->TSQL->Parameters->ParamByName("disc")->Value = this->discount;

DM->TSQL->Parameters->ParamByName("birt")->Value = this->birthday;

DM->TSQL->Parameters->ParamByName("adre")->Value = this->adress;

DM->TSQL->Parameters->ParamByName("telef")->Value = this->telefon;

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_discount;

DM->TSQL->ExecSQL();

}

void TDiscount::DeleteDBDiscount(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("DELETE FROM eMenu.dbo.Discounts");

DM->TSQL->SQL->Add("WHERE id\_discount = :id and id\_discount <> 0");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_discount;

DM->TSQL->ExecSQL();

}

void \_\_fastcall TFEditDiscount::FormShow(TObject \*Sender)

{

edit\_id->Clear();

edit\_discount->Text = 0;

edit\_pib\_client->Clear();

edit\_telefon->Clear();

edit\_adress->Clear();

date\_birthday->Date = StrToDate("01.01.1970");

this->add = true;

Button1->Show();

}

(float disc,String pib,String adres,String telefon,TDate birth)

void \_\_fastcall TFEditDiscount::ButtonAcceptClick(TObject \*Sender)

{

try{

TDiscount discount;

discount.setId\_discount(StrToInt(edit\_id->Text));

discount.setInfoDiscount(StrToFloat(edit\_discount->Text),edit\_pib\_client->Text,edit\_adress->Text,edit\_telefon->Text,date\_birthday->Date);

if(add){

discount.InsertDBDiscount();

}

else {

discount.UpdateDBDiscount();

}

DM->OpenDB();

Hide();

}

catch(...){

MessageBox(NULL, L"Íå â³ðíî ââåäåí³ äàí³!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFEditDiscount::Button1Click(TObject \*Sender)

{

srand(time(NULL));

long int rands = random(rand()+999999999999);

this->edit\_id->Text = rands;

}

void \_\_fastcall TFEditDiscount::edit\_idKeyPress(TObject \*Sender, System::WideChar &Key)

{

if( Key == VK\_BACK )

return;

if( (Key < L'0') || (Key > L'9') ) Key = 0;

}

void \_\_fastcall TFEditDiscount::edit\_discountKeyPress(TObject \*Sender, System::WideChar &Key)

{

if( Key == VK\_BACK) return;

if( (Key < L'0') || (Key > L'9') ) Key = 0;

}

void \_\_fastcall TFEditDiscount::edit\_discountChange(TObject \*Sender)

{

if (!edit\_discount->Text.IsEmpty())

if(StrToInt(edit\_discount->Text) > 100)

{ ShowMessage("Çíèæêà ìàº áóòè ìåíøå àáî ð³âíà 100");

edit\_discount->Text = 0;

}

}

void \_\_fastcall TFEditDiscount::ButtonCancelClick(TObject \*Sender)

{

Hide();

}

Б.9 Вміст файлу EditFood.cpp

#include <vcl.h>

#pragma hdrstop

#include "EditFood.h"

#include "DataModule.h"

#include "Admin.h"

#include "Authorization.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditIngredient.h"

#include "EditPersonal.h"

#include "eMenuPCH1.h"

#include "MainForm.h"

#include "Math.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFEditFood \*FEditFood;

\_\_fastcall TFEditFood::TFEditFood(TComponent\* Owner)

: TForm(Owner)

{

add = true;

}

void TFood::setIdFood(int a){ this->id\_food = a;}

void TFood::setWeigh\_Food(float a){ this->weight\_food = a;}

void TFood::setPicture(TMemoryStream\* a){ this->picture = a;}

void TFood::setVisible(bool b){ this->visible = b;}

void TFood::setInfoFood(AnsiString name,AnsiString data, AnsiString unit){

this->name\_food = name;

this->data\_food = data;

this->unit\_food = unit;

}

void TFood::setCostFood(float cost,float mark,float price){

this->cost\_price\_food = cost;

this->mark\_up = mark;

this->price\_food = price;

}

int TFood::getIdFood(){ return this->id\_food;}

void TFood::InsertDBFood(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("INSERT INTO eMenu.dbo.Food(kod\_category,name\_food,data\_food,unit\_food,cost\_price\_food,mark\_up,price\_food,weight\_food,picture,visible)");

DM->TSQL->SQL->Add("VALUES(:p1,:p2,:p3,:p4,:p5,:p6,:p7,:p8,:pict,:vis)");

DM->TSQL->Parameters->ParamByName("p1")->Value = this->getId\_category();

DM->TSQL->Parameters->ParamByName("p2")->Value = this->name\_food;

DM->TSQL->Parameters->ParamByName("p3")->Value = this->data\_food;

DM->TSQL->Parameters->ParamByName("p4")->Value = this->unit\_food;

DM->TSQL->Parameters->ParamByName("p5")->Value = this->cost\_price\_food;

DM->TSQL->Parameters->ParamByName("p6")->Value = this->mark\_up;

DM->TSQL->Parameters->ParamByName("p7")->Value = this->price\_food;

DM->TSQL->Parameters->ParamByName("p8")->Value = this->weight\_food;

DM->TSQL->Parameters->ParamByName("vis")->Value = this->visible;

DM->TSQL->Parameters->ParamByName("pict")->DataType = ftBlob;

DM->TSQL->Parameters->ParamByName("pict")->LoadFromStream(this->picture,ftBlob);

DM->TSQL->ExecSQL();

}

void TFood::UpdateDBFood(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.Food SET visible = :p9, kod\_category = :p1, name\_food =:p2, data\_food =:p3,unit\_food =:p4,cost\_price\_food =:p5, mark\_up =:p6, price\_food =:p7, weight\_food =:p8");

DM->TSQL->SQL->Add("WHERE id\_food = :id");

DM->TSQL->Parameters->ParamByName("p1")->Value = this->getId\_category();

DM->TSQL->Parameters->ParamByName("p2")->Value = this->name\_food;

DM->TSQL->Parameters->ParamByName("p3")->Value = this->data\_food;

DM->TSQL->Parameters->ParamByName("p4")->Value = this->unit\_food;

DM->TSQL->Parameters->ParamByName("p5")->Value = this->cost\_price\_food;

DM->TSQL->Parameters->ParamByName("p6")->Value = this->mark\_up;

DM->TSQL->Parameters->ParamByName("p7")->Value = this->price\_food;

DM->TSQL->Parameters->ParamByName("p8")->Value = this->weight\_food;

DM->TSQL->Parameters->ParamByName("p9")->Value = this->visible;

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_food;

DM->TSQL->ExecSQL();

}

void TFood::UpdateFotoDBFood(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.Food Set picture = :P1 WHERE id\_food = :id");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_food;

DM->TSQL->Parameters->ParamByName("P1")->DataType = ftBlob;

TMemoryStream \*str = new TMemoryStream;

FEditFood->Image1->Picture->SaveToStream(str);

DM->TSQL->Parameters->ParamByName("P1")->LoadFromStream(str,ftBlob);

DM->TSQL->ExecSQL();

}

void TFood::DeleteDBFood(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("DELETE FROM eMenu.dbo.Food");

DM->TSQL->SQL->Add("WHERE id\_food = :id");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_food;

DM->TSQL->ExecSQL();

}

void \_\_fastcall TFEditFood::ButtonAcceptClick(TObject \*Sender)

{

try{

TFood food;

food.setInfoFood(Edit\_name\_food->Text,\

"",\

CB\_unit->Text);

food.setCostFood(StrToFloat(Edit\_cost\_price->Text),\

StrToFloat(Edit\_mark\_up->Text),\

StrToFloat(Edit\_price->Text));

food.setWeigh\_Food(StrToFloat(Edit\_weight->Text));

food.setId\_category(StrToInt(CB\_id\_category->Text));

food.setVisible(CBvisible->Checked);

TMemoryStream \*str = new TMemoryStream;

Image1->Picture->SaveToStream(str);

food.setPicture(str);

if(this->add)

{

food.InsertDBFood();

}

else

{

food.setIdFood(StrToInt(EditID->Text));

food.UpdateDBFood();

food.UpdateFotoDBFood();

}

DM->OpenDB();

FEditFood->Hide();

}

catch(...){

MessageBox(NULL, L"Íå â³ðíî ââåäåí³ äàí³!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFEditFood::EditImageClick(TObject \*Sender)

{

if(OPD->Execute())Image1->Picture->LoadFromFile(OPD->FileName);

}

void \_\_fastcall TFEditFood::FormShow(TObject \*Sender)

{

add = true;

CB\_id\_category->Clear();

CB\_name\_category->Clear();

CB\_unit->ItemIndex = 0;

EditID->Clear();

Edit\_cost\_price->Text = 0;

Edit\_mark\_up->Text = 0;

Edit\_price->Text = 0;

Edit\_name\_food->Clear();

Edit\_weight->Text = 0;

CBvisible->Checked = true;

Image1->Picture->Graphic = new Graphics::TBitmap;

DM->ATCategory->First();

for (int i = 0; i < DM->ATCategory->RecordCount; i++) {

CB\_id\_category->Items->Add(DM->ATCategory->FieldByName("id\_category")->AsString);

CB\_name\_category->Items->Add(DM->ATCategory->FieldByName("name\_category")->AsString);

DM->ATCategory->Next();

}

CB\_id\_category->ItemIndex = 0;

}

void \_\_fastcall TFEditFood::EditIDChange(TObject \*Sender)

{

if (EditID->Text.IsEmpty()) {

TabIngredient->TabVisible = false;

} else {

TabIngredient->TabVisible = true;

}

}

void \_\_fastcall TFEditFood::CB\_id\_categoryChange(TObject \*Sender)

{

CB\_name\_category->ItemIndex = CB\_id\_category->ItemIndex;

}

void \_\_fastcall TFEditFood::CB\_name\_categoryChange(TObject \*Sender)

{

CB\_id\_category->ItemIndex = CB\_name\_category->ItemIndex;

}

void TFEditFood::CalculationPrice(){

if (!Edit\_cost\_price->Text.IsEmpty() && !Edit\_mark\_up->Text.IsEmpty()) {

float price = StrToFloat(Edit\_cost\_price->Text), mark =StrToFloat(Edit\_mark\_up->Text);

Edit\_price->Text = price + (price \*(mark/100));

} else Edit\_price->Text = 0;

}

void \_\_fastcall TFEditFood::Edit\_cost\_priceChange(TObject \*Sender)

{

CalculationPrice();

}

void \_\_fastcall TFEditFood::Edit\_mark\_upChange(TObject \*Sender)

{

CalculationPrice();

}

void \_\_fastcall TFEditFood::ToolAddClick(TObject \*Sender)

{

FEditIngredient->Show();

}

void \_\_fastcall TFEditFood::ToolDeleteClick(TObject \*Sender)

{

try {

if (IDYES == MessageBox(NULL, L"Âè âïåâíåí³, ùî õî÷åòå âèäàëèòè ?", L"Ï³äâåðäæåííÿ!", MB\_YESNO |MB\_ICONQUESTION))

{

TIngredient ingredient;

ingredient.setId\_ingredient(DM->ATFoodIngredient->FieldByName("id\_listingrfood")->AsInteger);

ingredient.DeleteDBIngredient();

DM->RefreshADO(DM->ATFoodIngredient);

}

}

catch(...){

MessageBox(NULL, L"Äàí³, ïîâ'ÿçàíí³ ç äàíîþ ³íôîðìàö³ºþ. Íå ìîæëèâî âèäàëèòè!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFEditFood::ButtonCalculationPriceClick(TObject \*Sender)

{

float price = 0;

DM->ATFoodIngredient->First();

for (int i = 0; i < DM->ATFoodIngredient->RecordCount ; i++) {

float prc = DM->ATFoodIngredient->FieldByName("price")->AsFloat;

float cnt = DM->ATFoodIngredient->FieldByName("counts")->AsFloat;

price += prc \* cnt;

DM->ATFoodIngredient->Next();

}

ShowMessage("Ñîá³âàðò³ñòü - "+FloatToStrF(price,ffFixed,10,2)+" ãðí");

Edit\_cost\_price->Text = FloatToStrF((price\*100)/100,ffFixed,10,2);

}

void \_\_fastcall TFEditFood::ButtonCalculationWeightClick(TObject \*Sender)

{

float weight = 0;

DM->ATFoodIngredient->First();

for (int i = 0; i < DM->ATFoodIngredient->RecordCount ; i++) {

if(DM->ATFoodIngredient->FieldByName("unit")->AsString != "øò."){

float prc = DM->ATFoodIngredient->FieldByName("counts")->AsFloat;

weight += prc; }

DM->ATFoodIngredient->Next();

}

ShowMessage("Ïðèáëèçíà âàãà - "+FloatToStrF(weight,ffFixed,10,3)+" êã(ë). \nÇàóâàæåííÿ!!! Øòóêè íå âðàõîâóºòüñÿ!");

Edit\_weight->Text = FloatToStrF(weight,ffFixed,10,3);

}

void \_\_fastcall TFEditFood::Edit\_weightKeyPress(TObject \*Sender, System::WideChar &Key)

{

if( Key == VK\_BACK || (Key == ',')) return;

if( (Key < L'0') || (Key > L'9') ) Key = 0;

}

void \_\_fastcall TFEditFood::Edit\_cost\_priceKeyPress(TObject \*Sender, System::WideChar &Key)

{

if( Key == VK\_BACK || (Key == ',')) return;

if( (Key < L'0') || (Key > L'9') ) Key = 0;

}

void \_\_fastcall TFEditFood::Edit\_mark\_upKeyPress(TObject \*Sender, System::WideChar &Key)

{

if( Key == VK\_BACK) return;

if( (Key < L'0') || (Key > L'9') ) Key = 0;

}

void \_\_fastcall TFEditFood::Edit\_priceKeyPress(TObject \*Sender, System::WideChar &Key)

{

if( Key == VK\_BACK || (Key == ',')) return;

if( (Key < L'0') || (Key > L'9') ) Key = 0;

}

void \_\_fastcall TFEditFood::ButtonCancelClick(TObject \*Sender)

{

FEditFood->Hide();

}

Б.10 Вміст файлу EditIngrient.cpp

#include <vcl.h>

#pragma hdrstop

#include "EditIngredient.h"

#include "DataModule.h"

#include "Admin.h"

#include "Authorization.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditPersonal.h"

#include "eMenuPCH1.h"

#include "MainForm.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFEditIngredient \*FEditIngredient;

\_\_fastcall TFEditIngredient::TFEditIngredient(TComponent\* Owner)

: TForm(Owner)

{

}

void TIngredient::setId\_ingredient(int a ){ this->id\_ingredient = a;}

int TIngredient::getId\_ingredient(){ return this->id\_ingredient;}

void TIngredient::setInfoIngredient(AnsiString name,AnsiString unt,float prc,float cnt){

this->name\_ingredient = name;

this->unit = unt;

this->price = prc;

this->count\_unit= cnt;

}

void TIngredient::InsertDBIngredient(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("INSERT INTO eMenu.dbo.ListIngredientFood(kod\_food,name\_ingredient,counts,unit,price)");

DM->TSQL->SQL->Add("VALUES(:id,:name,:cnt,:unt,:prc)");

DM->TSQL->Parameters->ParamByName("id")->Value = this->getIdFood();

DM->TSQL->Parameters->ParamByName("name")->Value = this->name\_ingredient;

DM->TSQL->Parameters->ParamByName("cnt")->Value = this->count\_unit;

DM->TSQL->Parameters->ParamByName("unt")->Value = this->unit;

DM->TSQL->Parameters->ParamByName("prc")->Value = this->price;

DM->TSQL->ExecSQL();

}

void TIngredient::DeleteDBIngredient(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("DELETE FROM eMenu.dbo.ListIngredientFood");

DM->TSQL->SQL->Add("WHERE id\_listingrfood = :id");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_ingredient;

DM->TSQL->ExecSQL();

}

void \_\_fastcall TFEditIngredient::ButtonAccceptClick(TObject \*Sender)

{

try{

TIngredient ingredient;

ingredient.setIdFood(StrToInt(FEditFood->EditID->Text));

ingredient.setInfoIngredient(Edit\_name\_ingredient->Text,\

CB\_unit->Text,\

StrToFloat(Edit\_price->Text),\

StrToFloat(Edit\_count->Text));

ingredient.InsertDBIngredient();

DM->RefreshADO(DM->ATFoodIngredient);

FEditIngredient->Hide();

}

catch(...){

MessageBox(NULL, L"Íå â³ðíî ââåäåí³ äàí³!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFEditIngredient::FormShow(TObject \*Sender)

{

Edit\_name\_ingredient->Clear();

Edit\_count->Text = 0;

Edit\_price->Text = 0;

}

void \_\_fastcall TFEditIngredient::ButtonCancelClick(TObject \*Sender)

{

FEditIngredient->Hide();

}

void \_\_fastcall TFEditIngredient::Edit\_countKeyPress(TObject \*Sender, System::WideChar &Key)

{

if( Key == VK\_BACK || (Key == ',')) return;

if( (Key < L'0') || (Key > L'9') ) Key = 0;

}

void \_\_fastcall TFEditIngredient::Edit\_priceKeyPress(TObject \*Sender, System::WideChar &Key)

{

if( Key == VK\_BACK || (Key == ',')) return;

if( (Key < L'0') || (Key > L'9') ) Key = 0;

}

Б.11 Вміст файлуEditOrderMenu.cpp

#include <vcl.h>

#pragma hdrstop

#include "EditOrderMenu.h"

#include "Admin.h"

#include "Authorization.h"

#include "DataModule.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditIngredient.h"

#include "EditPersonal.h"

#include "EditTabel.h"

#include "eMenuPCH1.h"

#include "MainForm.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFEditOrderMenu \*FEditOrderMenu;

\_\_fastcall TFEditOrderMenu::TFEditOrderMenu(TComponent\* Owner)

: TForm(Owner)

{

}

void TOrderMenu::setIdOrder(int a){ this->id\_order = a;}

int TOrderMenu::getIdOrder(){ return this->id\_order;}

void TOrderMenu::setDateOrder(TDateTime open,TDateTime close) {

this->date\_open\_order = open;

this->date\_close\_order = close;

}

void TOrderMenu::setCloseOrder(bool b){ this->close\_order = b;}

void TOrderMenu::setPayment(float f){ this->payment = f;}

void TOrderMenu::CloseOrderMenu(){

DM->DoSQLExec(DM->TSQL,"UPDATE eMenu.dbo.OrderMenu SET date\_close\_order = GETDATE(), close\_order = 1 WHERE id\_order = "+IntToStr(this->getIdOrder()));

}

void TOrderMenu::UpdatePayment(){

DM->DoSQL(DM->TSQL,"SELECT sum(counts\*price\_food) as price\_all\

FROM dbo.ListOrderMenu INNER JOIN dbo.Food ON id\_food = kod\_food WHERE kod\_order = "+IntToStr(this->id\_order));

float val = DM->TSQL->FieldByName("price\_all")->AsFloat;

if(DM->TSQL->RecordCount!=0){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.OrderMenu SET payment = :pay WHERE id\_order = :id");

DM->TSQL->Parameters->ParamByName("pay")->Value = val;

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_order;

DM->TSQL->ExecSQL();

}

else{

DM->DoSQLExec(DM->TSQL,"UPDATE eMenu.dbo.OrderMenu SET payment = 0 WHERE id\_order ="+IntToStr(this->id\_order));

}

}

void TOrderMenu::InsertDBOrderMenu(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("INSERT INTO eMenu.dbo.OrderMenu(kod\_table,kod\_personal,kod\_discount)");

DM->TSQL->SQL->Add("VALUES(:t,:p,:d)");

DM->TSQL->Parameters->ParamByName("t")->Value = this->getId\_table();

DM->TSQL->Parameters->ParamByName("p")->Value = this->getId\_personal();

DM->TSQL->Parameters->ParamByName("d")->Value = this->getId\_discount();

DM->TSQL->ExecSQL();

}

void TOrderMenu::UpdateDBOrderMenu(){

}

void TOrderMenu::DeleteDBOrderMenu(){

DM->DoSQLExec(DM->TSQL,"DELETE FROM eMenu.dbo.ListOrderMenu WHERE kod\_order ="+IntToStr(this->id\_order));

DM->DoSQLExec(DM->TSQL,"DELETE FROM eMenu.dbo.OrderMenu WHERE id\_order ="+IntToStr(this->id\_order));

}

void TOrderMenu::UpdateDiscount(){

DM->DoSQL(DM->TSQL,"Select \* FROM eMenu.dbo.Discounts WHERE id\_discount = "+IntToStr(this->getId\_discount()));

if(DM->TSQL->RecordCount!=0){

DM->DoSQLExec(DM->TSQL,\

"UPDATE eMenu.dbo.OrderMenu SET kod\_discount ="+IntToStr(this->getId\_discount())+\

"WHERE id\_order = "+IntToStr(this->id\_order));

} else MessageBox(NULL, L"Íåìàº ³íôîðìàö³¿ ïîâ'ÿçîíî¿ ç äàííîþ êàðòêîþ", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

void TListOrder::setIdListingrfood(int i){ this->id\_listingrfood = i;}

int TListOrder::getIdListingrfood(){return this->id\_listingrfood;}

void TListOrder::setListCounts(float i){this->counts = i;}

void TListOrder::InsertDBListOrder(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("INSERT INTO eMenu.dbo.ListOrderMenu(kod\_order,kod\_food,counts)");

DM->TSQL->SQL->Add("VALUES(:t,:p,:d)");

DM->TSQL->Parameters->ParamByName("t")->Value = this->getIdOrder();

DM->TSQL->Parameters->ParamByName("p")->Value = this->getIdFood();

DM->TSQL->Parameters->ParamByName("d")->Value = this->counts;

DM->TSQL->ExecSQL();

}

void TListOrder::DeleteDBListOrder(){

try{

DM->DoSQLExec(DM->TSQL,\

"DELETE FROM eMenu.dbo.ListOrderMenu where id\_listordermenu ="+IntToStr(this->id\_listingrfood));

} catch(...){MessageBox(NULL, L"Íå âäàëîñÿ âèäàëèòè!", L"Â³äìîâà!", MB\_OK | MB\_ICONERROR); }

}

void \_\_fastcall TFEditOrderMenu::FormClose(TObject \*Sender, TCloseAction &Action)

{

FMainForm->Show();

}

void \_\_fastcall TFEditOrderMenu::btnRemovClick(TObject \*Sender)

{

try {

if (IDYES == MessageBox(NULL, L"Âè âïåâíåí³, ùî õî÷åòå âèäàëèòè ?", L"Ï³äâåðäæåííÿ!", MB\_YESNO |MB\_ICONQUESTION))

{

TListOrder list;

list.setIdListingrfood(DM->OTListOpenOrder->FieldByName("id\_listordermenu")->AsInteger);

list.setIdOrder(DM->OTListOpenOrder->FieldByName("kod\_order")->AsInteger);

list.DeleteDBListOrder();

list.UpdatePayment();

DM->OpenDBOficiant();

}

}

catch(...){

//MessageBox(NULL, L"Äàí³, ïîâ'ÿçàíí³ ç äàíîþ ³íôîðìàö³ºþ. Íå ìîæëèâî âèäàëèòè!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFEditOrderMenu::btnAddToOrderClick(TObject \*Sender)

{

FEditOrderMenu->Hide();

FMainForm->Show();

}

void \_\_fastcall TFEditOrderMenu::btnDiscountClick(TObject \*Sender)

{

try{

String str;

if (InputQuery("Ââåä³òü íîìåð äèñêîíòíî¿ êàðòêè","Êàðòêà",str)) {

TOrderMenu order;

order.setIdOrder(DM->OTOpenOrder->FieldByName("id\_order")->AsInteger);

order.setId\_discount(StrToInt(str));

order.UpdateDiscount();

DM->OpenDBOficiant();

}

} catch(...){

MessageBox(NULL, L"Ïåðåâ³ðüòå ïðàâèëüí³ñòü äàíèõ!", L"Â³äìîâà!", MB\_OK | MB\_ICONWARNING);

}

}

void \_\_fastcall TFEditOrderMenu::btnReportDayClick(TObject \*Sender)

{

DM->DoSQL(DM->TSQL,"SELECT SUM(payment - (payment \* discount/100)) AS DayPayment \

FROM ((OrderMenu INNER JOIN Discounts ON id\_discount = kod\_discount)\

INNER JOIN Personal ON id\_personal = kod\_personal)\

INNER JOIN ListTable ON id\_table = kod\_table \

WHERE date\_open\_order >= '"+DateToStr(Date())+" 00:00:00' and date\_open\_order <= '"+DateToStr(Date())+" 23:59:59'");

ShowMessage("Äåíèé çàðîá³òîê - "+DM->TSQL->FieldByName("DayPayment")->AsString+" ãðí.");

}

void \_\_fastcall TFEditOrderMenu::btnOpenOrderClick(TObject \*Sender)

{

String str = DM->OTNotOcupTable->FieldByName("id\_table")->AsString;

if(!str.IsEmpty()){

TOrderMenu order;

order.setId\_personal(DM->id\_user);

order.setId\_discount(0);

order.setId\_table(DM->OTNotOcupTable->FieldByName("id\_table")->AsInteger);

order.UpdateInverseOccupation();

order.InsertDBOrderMenu();

DM->OpenDB();

DM->OTOpenOrder->Last();

}

else {

MessageBox(NULL, L"Íåìàº â³ëüíèõ ì³ñöü!", L"Â³äìîâà!", MB\_OK | MB\_ICONWARNING);

DM->OpenDB();

}

}

void \_\_fastcall TFEditOrderMenu::btnCloseOrderClick(TObject \*Sender)

{

try{

if (!CB\_id\_order->Text.IsEmpty()){

int id = DM->OTOpenOrder->FieldByName("id\_order")->AsInteger;

TOrderMenu order;

order.setIdOrder(id);

order.setId\_table(DM->OTOpenOrder->FieldByName("kod\_table")->AsInteger);

order.setOccupListTable(false);

order.UpdateOccupation();

order.CloseOrderMenu();

DM->OpenDB();

DM->TSQLCheck->Close();

DM->TSQLCheck->Open();

DM->TSQLCheck->Filtered = false;

DM->TSQLCheck->Filter = "id\_order = "+IntToStr(id);

DM->TSQLCheck->Filtered = true;

DM->frxCheck->ShowReport();

ShowMessage("Çàìîâëåííÿ ¹"+IntToStr(id)+" çàêðèòî");

} else

{

MessageBox(NULL, L"Âèáåð³òü çàìîâëåííÿ!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

DM->OpenDB();

}

} catch(...){MessageBox(NULL, L"Ïåðåâ³ðüòå ïðàâèëüí³ñòü äàíèõ!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);}

}

void \_\_fastcall TFEditOrderMenu::Button1Click(TObject \*Sender)

{

ShowMessage(DM->OTNotOcupTable->FieldByName("id\_table")->AsString) ;

}

void \_\_fastcall TFEditOrderMenu::btnPrintClick(TObject \*Sender)

{

DM->TSQLCheck->Close();

DM->TSQLCheck->Open();

DM->TSQLCheck->Filtered = false;

if (DM->OTOpenOrder->RecordCount !=0) {

DM->TSQLCheck->Filter = "id\_order ="+DM->OTOpenOrder->FieldByName("id\_order")->AsString;

DM->TSQLCheck->Filtered = true;

DM->frxCheck->ShowReport();

} else MessageBox(NULL, L"Âèáåð³òü çàìîâëåííÿ!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

Б.12 Вміст файлу EditPersonal.cpp

#include <vcl.h>

#pragma hdrstop

#include "EditPersonal.h"

#include "DataModule.h"

#include "Admin.h"

#include "Authorization.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditIngredient.h"

#include "eMenuPCH1.h"

#include "MainForm.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFEditPersonal \*FEditPersonal;

\_\_fastcall TFEditPersonal::TFEditPersonal(TComponent\* Owner)

: TForm(Owner)

{

}

void \_\_fastcall TFEditPersonal::ButtonCancelClick(TObject \*Sender)

{

Hide();

}

void \_\_fastcall TFEditPersonal::FormShow(TObject \*Sender)

{

this->add = true;

EditID->Clear();

date\_birthday->Date = StrToDate("01.01.1970");

edit\_date\_release->Text = "";

date\_work->Date = Date();

edit\_telefon->Clear();

edit\_pib\_personal->Clear();

edit\_login->Clear();

edit\_adress->Clear();

edit\_password->Clear();

PersonalAccess->Checked = false;

PersonalActivity->Checked = true;

ButtonRelease->Hide();

edit\_date\_release->Hide();

Label3->Hide();

}

void TPersonal::setId\_personal(int a){

this->id\_personal = a;

}

int TPersonal::getId\_personal(){

return this->id\_personal;

}

void TPersonal::setAccess(bool a){

this->access = a;

}

void TPersonal::setActivity(bool a){

this->activity = a;

}

void TPersonal::setLoginPassword(AnsiString log,AnsiString pass){

this->login = log;

this->password = pass;

}

void TPersonal::setInformationPersonal(AnsiString pib,AnsiString tel,AnsiString adr,TDateTime dat){

this->pib\_personal = pib;

this->telefon = tel;

this->adress = adr;

this->birthday = dat;

}

void TPersonal::setDateWork(TDateTime a){

this->date\_work = a;

}

void TPersonal::setDateRelease(AnsiString a){

this->date\_release = a;

}

AnsiString TPersonal::getLogin(){

return this->login;

}

AnsiString TPersonal::getPassword(){

return this->password;

}

bool TPersonal::getAccess(){

return this->access;

}

bool TPersonal::getActivity(){

return this->activity;

}

void TPersonal::InsertDBPersonal(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("INSERT INTO eMenu.dbo.Personal(logins,passwords,access,activity,pib\_personal,telefon,adress,birthday,data\_of\_work)");

DM->TSQL->SQL->Add("VALUES(:log,:pass,:acs,:act,:pib,:tel,:adr,:birt,:datework)");

DM->TSQL->Parameters->ParamByName("log")->Value = this->login;

DM->TSQL->Parameters->ParamByName("pass")->Value = this->password;

DM->TSQL->Parameters->ParamByName("acs")->Value = this->access;

DM->TSQL->Parameters->ParamByName("act")->Value = this->activity;

DM->TSQL->Parameters->ParamByName("pib")->Value = this->pib\_personal;

DM->TSQL->Parameters->ParamByName("tel")->Value = this->telefon;

DM->TSQL->Parameters->ParamByName("adr")->Value = this->adress;

DM->TSQL->Parameters->ParamByName("birt")->Value = this->birthday;

DM->TSQL->Parameters->ParamByName("datework")->Value = this->date\_work;

DM->TSQL->ExecSQL();

}

void TPersonal::UpdateDBPersonal(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.Personal SET \

logins = :log,passwords = :pass,access = :acs,activity = :act,\

pib\_personal = :pib,telefon = :tel,adress = :adr,birthday = :birt,\

data\_of\_work = :datework");

DM->TSQL->SQL->Add("WHERE id\_personal = :id");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_personal;

DM->TSQL->Parameters->ParamByName("log")->Value = this->login;

DM->TSQL->Parameters->ParamByName("pass")->Value = this->password;

DM->TSQL->Parameters->ParamByName("acs")->Value = this->access;

DM->TSQL->Parameters->ParamByName("act")->Value = this->activity;

DM->TSQL->Parameters->ParamByName("pib")->Value = this->pib\_personal;

DM->TSQL->Parameters->ParamByName("tel")->Value = this->telefon;

DM->TSQL->Parameters->ParamByName("adr")->Value = this->adress;

DM->TSQL->Parameters->ParamByName("birt")->Value = this->birthday;

DM->TSQL->Parameters->ParamByName("datework")->Value = this->date\_work;

DM->TSQL->ExecSQL();

}

void TPersonal::DeleteDBPersonal(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("DELETE FROM eMenu.dbo.Personal");

DM->TSQL->SQL->Add("WHERE id\_personal = :id");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_personal;

DM->TSQL->ExecSQL();

}

void TPersonal::UpdateDBPersonalRelease(){

try{

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.Personal SET release\_date = :data");

DM->TSQL->SQL->Add("WHERE id\_personal = :id");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_personal;

DM->TSQL->Parameters->ParamByName("data")->Value = StrToDate(this->date\_release);

DM->TSQL->ExecSQL();

DM->OpenDB();

}

catch(...){

ShowMessage("Íåêîðåêòíà äàòà!");

}

}

void \_\_fastcall TFEditPersonal::ButtonAcceptClick(TObject \*Sender)

{

try{

TPersonal personal;

personal.setLoginPassword(edit\_login->Text,\

edit\_password->Text);

personal.setInformationPersonal(edit\_pib\_personal->Text,\

edit\_telefon->Text,\

edit\_adress->Text,\

date\_birthday->Date);

personal.setAccess(PersonalAccess->Checked);

personal.setActivity(PersonalActivity->Checked);

personal.setDateWork(date\_work->Date);

personal.setDateRelease(edit\_date\_release->Text);

if (EditID->Text == 1){

personal.setAccess(True);

personal.setActivity(True);

}

if(add){

personal.InsertDBPersonal();

} else {

personal.setId\_personal(StrToInt(EditID->Text));

personal.UpdateDBPersonal();

}

DM->OpenDB();

FEditPersonal->Hide();

}

catch(...){

MessageBox(NULL, L"Íå â³ðíî ââåäåí³ äàí³!", L"Â³äìîâà!", MB\_OK |MB\_ICONWARNING);

}

}

void \_\_fastcall TFEditPersonal::ButtonReleaseClick(TObject \*Sender)

{

TPersonal personal;

personal.setId\_personal(StrToInt(EditID->Text));

personal.setDateRelease(edit\_date\_release->Text);

personal.UpdateDBPersonalRelease();

}

Б.13 Вміст файлу EditTable.cpp

#include <vcl.h>

#pragma hdrstop

#include "EditTabel.h"

#include "Admin.h"

#include "Authorization.h"

#include "DataModule.h"

#include "EditCategory.h"

#include "EditDiscount.h"

#include "EditFood.h"

#include "EditIngredient.h"

#include "EditPersonal.h"

#include "eMenuPCH1.h"

#include "MainForm.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFEditTable \*FEditTable;

\_\_fastcall TFEditTable::TFEditTable(TComponent\* Owner)

: TForm(Owner)

{

}

void \_\_fastcall TFEditTable::ButtonCancelClick(TObject \*Sender)

{

Hide();

}

void TListTable::setId\_table(int a){ this->id\_table = a;}

int TListTable::getId\_table(){return this->id\_table;}

void TListTable::setNameListTable(AnsiString a) {this->name = a;}

void TListTable::setOccupListTable(bool b){ this->occupation = b;}

void TListTable::InsertDBListTable(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("INSERT INTO eMenu.dbo.ListTable(name\_table)");

DM->TSQL->SQL->Add("VALUES(:P1)");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->name;

DM->TSQL->ExecSQL();

}

void TListTable::UpdateOccupation(){

DM->TSQL->SQL->Clear();

DM->TSQL->SQL->Add("UPDATE eMenu.dbo.ListTable SET occupation = :ocup");

DM->TSQL->SQL->Add("WHERE id\_table = :id");

DM->TSQL->Parameters->ParamByName("id")->Value = this->id\_table;

DM->TSQL->Parameters->ParamByName("ocup")->Value = this->occupation;

DM->TSQL->ExecSQL();

}

void TListTable::UpdateInverseOccupation(){

DM->DoSQLExec(DM->TSQL,"Update eMenu.dbo.ListTable SET occupation = occupation ^1 WHERE id\_table ="+IntToStr(this->id\_table));

}

void TListTable::DeleteDBListTable(){

DM->DoSQLExec(DM->TSQL,"DELETE FROM eMenu.dbo.ListTable WHERE id\_table ="+IntToStr(this->id\_table));

}

void \_\_fastcall TFEditTable::ButtonAccceptClick(TObject \*Sender)

{

TListTable table;

table.setNameListTable(Edit\_table->Text);

table.InsertDBListTable();

DM->OpenDB();

Hide();

}