Державний вищий навчальний заклад

«Чернівецький політехнічний коледж»

Циклова комісія інженерії програмного забезпечення

**Пояснювальна записка**

до комплексного курсового проекту

на тему: Електронне меню

Студента IV курсу 541 групи

спеціальності 5.05010301

«Розробка програмного забезпечення»

Гудан Ю. В.

(прізвище та ініціали)

Керівник Серьогіна І.В.

Оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.І. Фесік

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.М. Гуменна

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ І.В. Серьогіна

(підпис)

м. Чернівці – 2018 рік

**Зміст**

[Вступ 3](#_Toc508820909)

[розділ 1.Обґрунтування необхідності розробки 4](#_Toc508820910)

[1.1 Аналіз предметної області 4](#_Toc508820911)

[1.2 Аналіз сучасного стану та перспективи розвитку програмного забезпечення 5](#_Toc508820912)

[1.3 Аналіз існуючого програмного забезпечення 6](#_Toc508820913)

[1.4 Постановка задачі 8](#_Toc508820914)

[Розділ 2. Технічний проект 10](#_Toc508820915)

[2.1 Інформаційна модель предметної області 10](#_Toc508820916)

[2.2 Концептуальна модель предметної області 10](#_Toc508820917)

[2.3 Моделювання даних 12](#_Toc508820918)

[2.4 Представлення реалізації 16](#_Toc508820919)

[Розділ 3. Робочий проект 17](#_Toc508820920)

[3.1 Засоби розробки 17](#_Toc508820921)

[3.2 Інтерфейс програмного забезпечення 18](#_Toc508820922)

[3.3 Розробка документів на супроводження програмного забезпечення 20](#_Toc508820923)

[Розділ 4. Тестування програмного забезпечення 25](#_Toc508820926)

[4.1 Тестування адміністраторської частини 25](#_Toc508820927)

[4.2Тестування клієнтської частини 25](#_Toc508820928)

[4.3 Тестування користувацького інтерфейсу 26](#_Toc508820929)

[4.4 Тестування роботи бази даних 27](#_Toc508820930)

[Висновки 28](#_Toc508820931)

[Перелікджерел 29](#_Toc508820932)

[додатки 30](#_Toc508820933)

# 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 відвідувачів даних закладів.

Мета комплексного курсового проекту розробка програмного продукту для автоматизації роботи в закладах громадського харчування.

Завдання комплексного курсового проекту є розробки програмного забезпечення для автоматизації заданої предметної області.

Об’єктом дослідження є ресторанне господарство .

Предметом дослідження є методології та засоби реалізації функцій адміністратора та офіціанта. В залежності від рівня доступу, для офіціанта є доступ лише для обліку замовлення, для адміністратора є повний доступ по обліку замовлення, перегляду та обробки інформації про страви, розрахунку продукції .

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 RAD Studio XE7 Builder C++ в зв’язку з його відповідним функціоналом для створення даного програмного забезпечення.

У першому розділі представлено перед проектне дослідження та проведення моделювання предметної області;

У другому розділі наведено обґрунтування вибору середовища та етапи розробки, надані інструкції програміста та користувача;

У третьому розділі наведено інструкцію програміста та користувача для користування даним програмним забезпечення.

У четвертому розділі проведено тестування програмного забезпечення.

# розділ 1.Обґрунтування необхідності розробки

## 1.1 Аналіз предметної області

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

Типи закладів громадського харчування:

* ресторани;
* бари;
* кафе, кафетерії;
* їдальні у навчальних закладах тощо;
* закусочні;
* буфети, магазини кулінарних виробів;
* фабрики-кухні, фабрики-заготівельні.

Ресторан – заклад громадського харчування, де подаються страви та напої.

Електронне меню – це інтерактивне меню, що реалізується за допомогою сенсорного дисплею, який розміщують біля столу або на столі, за яким сидить гість, або на робочих місцях офіціантів.

Однією із основних вимог сучасної індустрії ресторанного бізнесу є швидке і якісне обслуговування споживачів. 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. 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.

З цим програмним забезпеченням робота в сфері ресторанного господарства прискориться в багато раз.

## 1.2 Аналіз сучасного стану та перспективи розвитку програмного забезпечення

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 конкурентоспроможного, високоякісного, здатного максимально задовольнити потреби споживачів ресторанного господарства, 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 закладу відчути себе комфортно у виборі страв і при оформленні замовлення.

Статистика по закладах, що ввели дану інновацію показує, що результатом покращення якості обслуговування клієнтів за допомогою електронного 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 %;
* істотне збільшення кількості клієнтів (за рахунок підвищення популярності та впізнаваності);
* 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 (рис. 1.1).

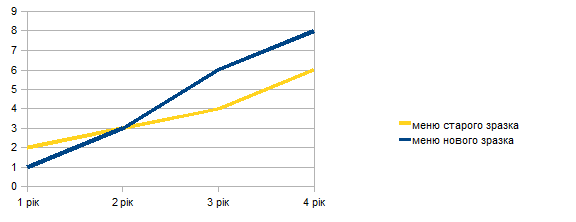


Рисунок 1.1 – Рівень доходу закладу при використанні електронного меню та меню паперового

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.

## 1.3 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 1.2.



Рисунок 1.2 – 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 істотно поліпшити систему обслуговування клієнтів.

Інтерфейс програмного забезпечення зображено на рисунку 1.3.



Рисунок 1.3 – 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.

## 1.4 Постановка задачі

Завданням комплексного курсового проекту є створити програмне забезпечення «Електронне меню» для обліку страв, інгредієнтів, напоїв, замовлень, цінових характеристик, розрахунку вартості замовлень, виведення меню в електронному вигляді, замовлення, страви і т.д. Звернути увагу, що кількість інгредієнта вимірюється в різних одиницях виміру (штуки, грами і т.п.).

Дана система повинна мати можливості:

* обробка даних в залежності від рівня доступу;
* пошук інформації за ключем;
* розрахунок вартості замовлення;
* перегляд та оформлення замовлення в електронному вигляді;
* створення рахунку.

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, наявність, похибка;
* інформація про замовлення: дата замовлення, прізвище та ініціали офіціанта, номер стола, оплата замовлення.

Вихідні дані: інформація про збір страви, рахунок відвідувача (чек).

Функціональна специфікація:

* доступ до інформації в залежності від рівня доступу до системи;
* пошук інформації по ключу;
* формування вихідних даних;
* розрахунок ціни оплати за замовлення;
* реплікація бази даних.

Розроблена специфікація вимог описує необхідні вхідні дані, які мають бути введені, та вихідні дані для виводу відповідної інформації.

Специфікація користувача:

* навички роботи з Windows-додатком;
* знання роботи в сфері діяльності адміністратора та офіціанта.

Системна специфікація:

* операційна система (ОС) Windows 7 та вище на якій програма може працювати;
* пакет Microsoft Office Access для підтримки бази даних;
* процесор (CPU) з частотою 1.8 Ггц та вище;
* оперативно запам’ятовуючий пристрій (ОЗП) на 512 мб та вище;
* вільне місце на жорсткому диску в межах 200 мб.

# Розділ 2. Технічний проект

## 2.1 Інформаційна модель предметної області

Алгоритм роботи програмного забезпечення зображено на рисунку 2.1.

Вхідна інформація

Інтерфейс програмного забезпечення

Сервер бази даних

Вихідна інформація

Системи управління бази даних

Дані бази даних

Запит на отримання даних

Передача

нових даних

Отримання даних

Передача параметрів

Передача даних

Рисунок 2.1 – Схема функціонування програмного забезпечення

## 2.2 Концептуальна модель предметної області

2.2.1 Діаграма прецедентів

Діаграма прецедентів показує структуру майбутнього програмного забезпечення, взаємодію основних сутностей. Діаграма прецедентів зображена на рисунку 2.2.

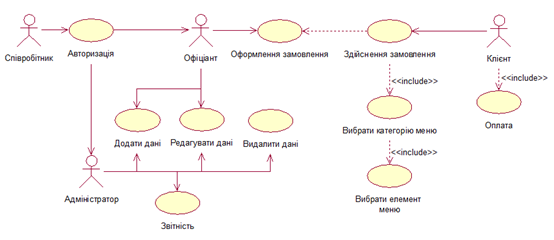


Рисунок 2.2 – Діаграма прецедентів

Опис діаграми прецедентів:

1. Актор «Офіціант» – даний актор при взаємодії з програмним забезпеченням має доступ до: додавання інформації про замовлення. Також даний актор взаємодіє з актором «Клієнт» під час оформлення замовлення.
2. Актор «Адміністратор» – даний актор при взаємодії з програмним забезпеченням має повний доступ до оброблення інформації та створення звітності.
3. Актор «Клієнт» – взаємодії з актором «Офіціант» під час здійснення замовлення та оплати замовлення.

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

2.2.2 Діаграма взаємодій

На рисунок 2.3 зображено діаграму взаємодій, яка описує дії під час оформлення замовлення до його оплати.



Рисунок 2.3 – Діаграма взаємодії

2.2.3 Діаграма послідовності

На рисунку 2.4 зображено діаграму послідовності, в якій описується послідовність дій під час оформлення замовлення. Коли відвідувач вибирає пункти меню, офіціант оформляє замовлення, подає готове замовлення на кухню, очікує оплату, після чого офіціант віддає чек та його замовлення відвідувачу.



Рисунок 2.4 – Діаграма послідовності

## 2.3 Моделювання даних

2.3.1 Концептуальна модель

Концептуальна модель зображена на рисунку 2.5.

Користувач

Замовлення

Страви та напої

Категорії

Інгредієнти

Журнал замовлень

Містить

Містить

Рисунок 2.5 – Концептуальна модель

В даній концептуальній модель описується взаємозв’язок «Користувача», «Замовлення», «Страви та напої», «Інгредієнти», «Категорії». А саме інформація про замовлення яка пов’язана з інформацією про страви та напої, також їм характерна інформація про інгредієнти страви та категорії страви та напою

2.3.2 Фізична модель даних

Для опису предметної області розроблені сім таблиці, в яких зберігається необхідна інформація для майбутньої системи.

Таблиця «Strava» призначена для зберігання інформації про страви та напої. Структура таблиці «Strava» зображена на рисунку 2.6.

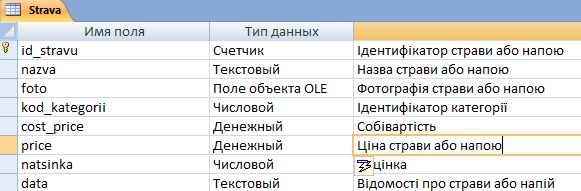


Рисунок 2.6 – Структура таблиць «Strava»

Таблиця «Sklad\_stravy» призначена для зберігання інформації яка зв’язує страву з інгредієнтами. Структура таблиці «Sklad\_stravy» зображена на рисунку 2.7.

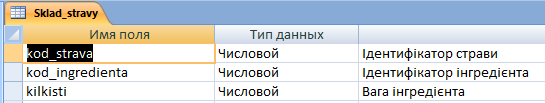


Рисунок 2.7 – Структура таблиць «Sklad\_stravy»

Таблиця «Ingredienty» призначена для зберігання інформації про інгредієнти. Структура таблиці «Ingredienty» зображена на рисунку 2.8.

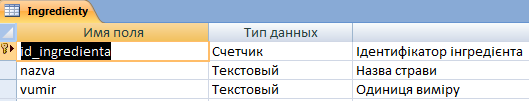


Рисунок 2.8 – Структура таблиць «Ingredienty»

Таблиця «Kategorii» призначена для зберігання інформації про категорії. Структура таблиці «Kategorii» зображена на рисунку 2.9.

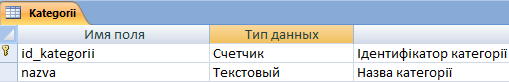


Рисунок 2.9 – Структура таблиць «Kategorii»

Таблиця «Personal» призначена для зберігання інформації про авторизацію користувача. Структура таблиці «Personal» зображена на рисунку 2.10.

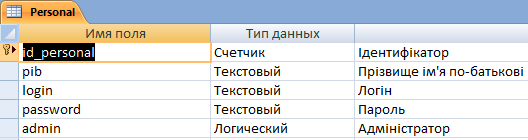


Рисунок 2.10 – Структура таблиць «Personal»

Таблиця «Zakaz\_menu» призначена для зберігання інформації про кількість страв або напоїв, які є в замовлені. Структура таблиці «Zakaz\_menu» зображена на рисунку 2.11.

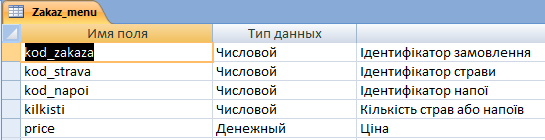


Рисунок 2.11 – Структура таблиць «Zakaz\_menu»

Таблиця «Zakaz» призначена для зберігання інформації про замовлення. Структура таблиці «Zakaz» зображена на рисунку 2.12.

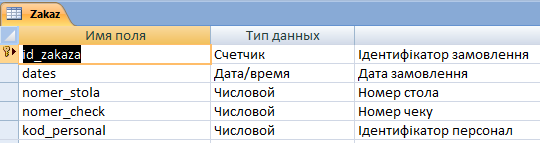


Рисунок 2.12 – Структура таблиць «Zakaz»

2.3.3 Логічна модель даних

Логічна модель даних представлена на рисунку 2.13, в який відображені зв`язких між таблицям.

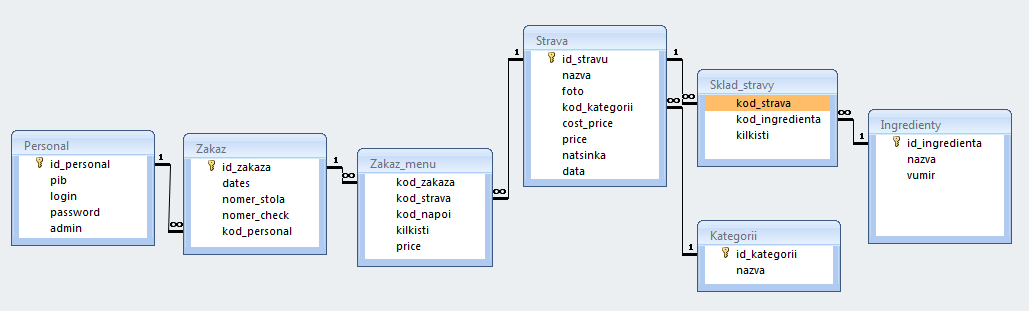


Рисунок 2.13 – Логічна модель даних

2.3.4 Діаграма класів

Для реалізації завдання було розроблено 5 класів. Діаграма класів, представлена на рисунку 2.14.

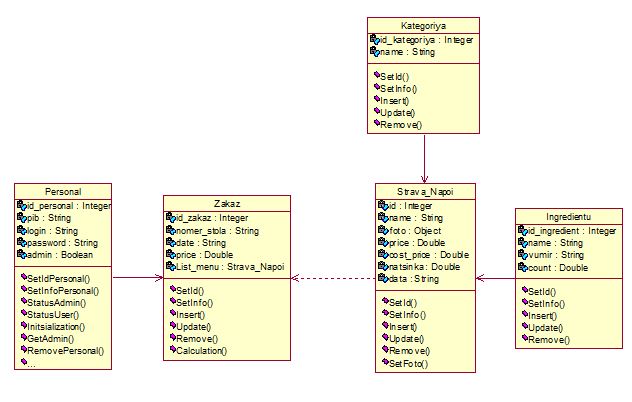


Рисунок 2.14 – Діаграма класів

Для роботи з інформацією в базі даних, використовується такі класи: Zakaz, Personal, Strava\_Napoi, Kategoriya, Ingredientu. Головним класом є Zakaz, даний клас наслідує методи та атрибути класів Personal та Strava\_Napoi, за допомогою цього класу можна обробляти інформацію про замовлення, фіксувати хто користується програмним забезпеченням під час оформлення замовлення.

Клас «Personal» пов’язаний з таблицею «Personal» в локальній базі даних, методами даного класу можна надавати різний доступ до програмного забезпечення.

Клас «Strava\_Napoi» пов’язаний з таблицею «Strava» в локальній базі даних, методами даного можна додавати, редагувати та видаляти інформацію про страви та напої. Даний клас наслідує методи та атрибути класів «Ingredientu» та «Kategoriya».

## 2.4 Представлення реалізації

На рисунку 2.15 зображено діаграму пакетів, завдяки користувацького інтерфейсу користувач зможе отримати доступ до обробки даних завдяки модулю даних, який дозволяє згрупувати та передати дані в зручному форматі, які зберігаються в базі даних.

# 

Рисунок 2.15 – Діаграма пакетів

В даному розділі було представлено реалізацію діаграм послідовності, стану, класів, взаємодії, описано кожну дію. Також розроблені логічна та фізичні моделі даних, яка описує інформацію про базу даних.

# Розділ 3. Робочий проект

## 3.1 Засоби розробки

Як середовище для розробки додатку була обрана програма RAD Studio XE7 Builder C++. Вона заснована компанією Embarcadero. Існує безліч середовищ розробки, але дана була обрана в зв’язку з її зр**у**чним графічним інтерфейсом і засобами налагодження. Основу RAD Studio становить не тільки саму мову, але і RAD (Rapid Application Development) – середовище швидкої розробки програм. Завдяки візуальному програмуванню, а також досить велика бібліотека візуальних RAD Studio XE7 Builder C++ дозволяє створювати програми найбільш швидко і ефективно, приймаючи на себе основну роботу, і залишаючи програмісту творчий процес. Зрозуміло, можливість швидкого створення професійних додатків для Windows робить RAD Studio XE7 Builder C++ – програмістів затребуваними у всіх галузях людської діяльності.

Головні можливості середовища розробки, які були відзначені в ході виконання курсової роботи:

* включені всі «інтелектуальні» можливості по редагуванню коду – авто доповнення, рефакторинг і аналіз коду;
* є можливість візуального перегляду майбутнього програми;
* збірка проекту працює швидше, ніж в альтернативних середовищах розробки;
* зручний конструктор інтерфейсів;
* зручне і інтуїтивно зрозуміле легування проекту;
* управління локалізацією і перекладами;
* управління різними базами даними;
* моніторинг використовуваної пам’яті;
* високопродуктивний компілятор в машинний код;
* об’єктно-орієнтована модель компонентів;
* візуальна (а, отже, і швидкісний) побудова додатків з програмних прототипів;
* масштабовані засоби для побудови баз даних.

## 3.2 Інтерфейс програмного забезпечення

Для зручності роботи з програмним забезпеченням при проектуванні інтерфейсу врахована, що адміністратор та офіціант мають різні можливості

Інтерфейс вікна «Головне вікно» програмного забезпечення зображено на рисунку 3.1. Дане вікно виводить інформацію про зміст поточного замовлення, страви та напої які є в меню, дану інформацію можна фільтрувати за допомогою вибору категорії. Також є можливість видача рахунку після проведення замовлення.

Даний інтерфейс є достатньо гнучкий та простий у використанні, всі елементи меню розміщені компактно для швидкого доступу до них.

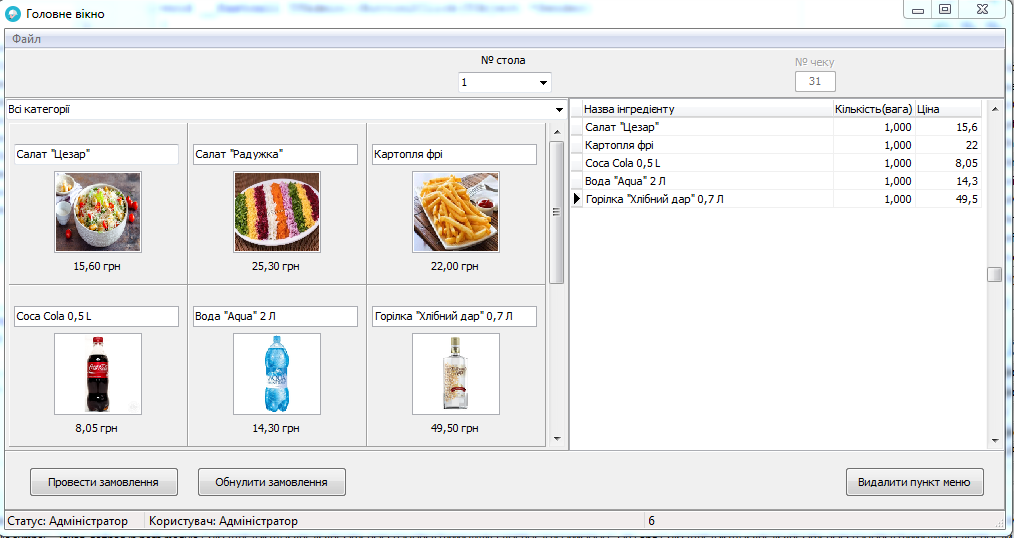


Рисунок 3.1 – Вікно «Головне вікно»

Інтерфейс вікна «Адміністрування» зображено на рисунку 3.2, дане вікно призначена для адміністратора. У даному вікні адміністратор може додавати, редагувати, видаляти інформацію про замовлення, страви та напої, категорії, інгредієнти страв, також можна переглядати інформацію в таблицях, які компактно розміщені в лівій частині вікна. Також адміністратор має доступ для управління користувачами даного програмного продукту. Інтерфейс вікна достатньо зрозумілий та сприятливий для роботи з ним.

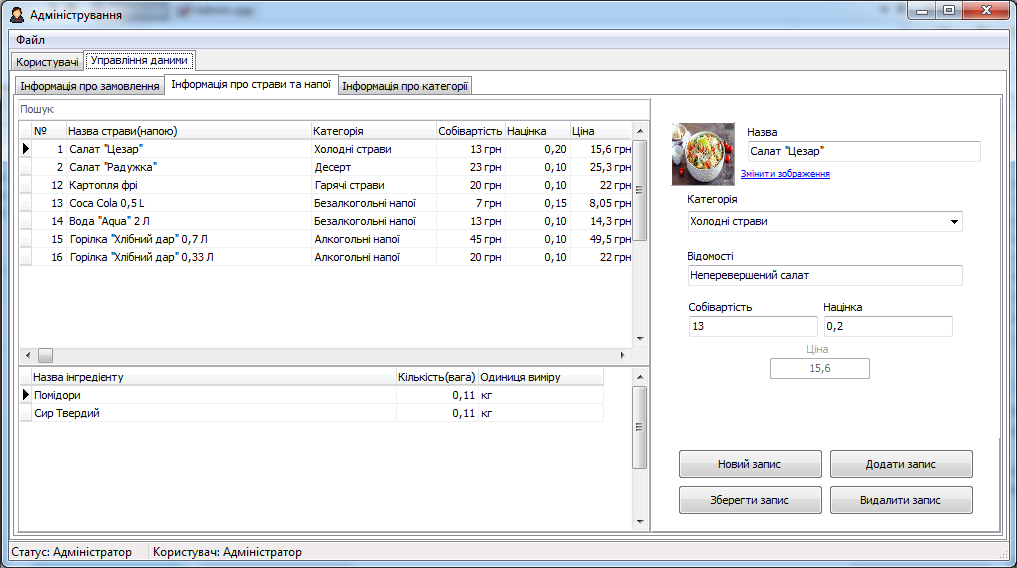


Рисунок 3.2 – Вікно «Адміністрування»

Для авторизації користувача розроблено вікно «Авторизація» зображено на рисунку 3.3. Дане вікно призначена для вводу особистих даних та вибору необхідної дії.

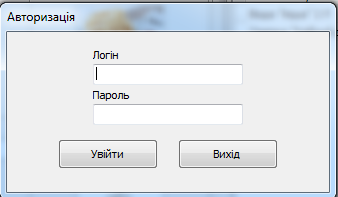


Рисунок 3.3 – Вікно «Авторизація»

Інтерфейс вікна «Додати інгредієнт до страви» зображено на рисунку 3.4, дане вікно призначена для додавання інформації про інгредієнт страви.

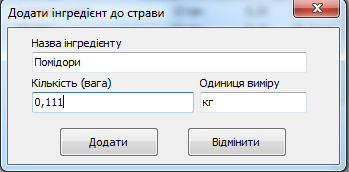


Рисунок 3.4 – Вікно «Додати інгредієнт до страви»

## 3.3 Розробка документів на супроводження програмного забезпечення

### 3.3.1 Інструкція програмісту

Даний проект складається з 4 форм та модулю з не візуальними компонентами для обробки даних (3.5).

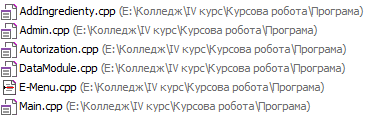


Рисунок 3.5 – Список форм

**DataModule.cpp, DataModule.h, HiderFile**

Модуль з не візуалізованими компонентами. Даний модуль містить такі компоненти для з’єднання проекту з базою даних:

* ADOConnection – для з’єднання з локальною базою даних;
* DBPersonal – для з’єднання з таблицею «Personal» в базі даних;
* DBGolovna – для з’єднання з таблицею «Страва» в базі даних, для вибору страви або напою з меню;
* DBStrava – для з’єднання з таблицею «Страва» в базі даних, для обробки даних;
* DBIngredienty – для з’єднання з таблицею «Склад страви» в базі даних, для перегляду та обробки інформації про інгредієнти, які відносяться до певної страви;
* DBIngred – для з’єднання з таблицею «Ingredienty» в базі даних;
* Check – для з’єднання з таблицею «Чек» в базі даних, для створення рахунку замовлення;
* TSQL – компонент який з’єднаний з локальною базою даних, для виконання певних запитів на додавання, оновлення, видалення;
* DBZakaz\_check – для з’єднання з таблицею «Склад заказу страви» в базі даних, для перегляду вмістимості рахунку замовлення;
* DBZakaz – для з’єднання з таблицею «Замовлення» в базі даних, для перегляду та обробки інформації про замовлення;
* DBKategorii – для з’єднання з таблицею «Kategoriy» в базі даних, для перегляду та обробки інформації про замовлення;
* MM\_Main – меню для головної сторінки;
* MM\_Admin – меню для сторінки адміністратора;
* PMAddIngredienty – спливаюче меню для виклику додавання та видалення інформації про категорію;
* Компоненти класу TDataSource – призначені для з’єднання компонентів TADOTable з компонентами класу TDBGrid. Всі компоненти, що розпочинаються з DB\*, є компонентами класу TADOTable.

Методи класу TDataModule:

* onovleny() – метод який обновляє дані в усіх таблицях бази даних;
* ActiveDB – метод який активує з’єднання з базою даних;
* OnCreate() – метод який при запуску програму підключається шлях до бази даних.

Базовий клас Zakaz містить атрибути: dates (дата замовлення), nomer\_stola (номер стола), price\_zakaz (ціна заказу), List\_Menu (об’єкт класу Strava\_Napoi). Методи базового класу :

* сеттери (для введення даних): SetId(),SetInfo();
* методи для обробки даних в базі даних: Insert(), Remove(), Update(), Calculation().

Дочірній клас Personal містить атрибут: PIB(Прізвище та ініціали),login (логін), password (пароль), admin (статус). Також даний клас містить методи:

* сеттери (для введення даних): setId(), setInfo();
* геттери (для отримання даних): getAdmin(), Initsialisation();
* методи для обробки даних в базі даних: Insert(), Remove(), Update(), StatusAdmin(), StatusUser();

Клас Strava\_Napoi містить такі атрибути: nazva(назва напою), foto (фотографія напою(страви)), cost\_price(собівартість), price(ціна), natsinka(націнка), data(додаткові відомості). Методи класу :

* сеттери (для введення даних): setId(), setFoto(), setInfo();
* методи для обробки даних в базі даних: Insert(), Remove(), Update().

Дочірній клас Kategorii класів Napoi\_Strava, містить атрибут: nazva (назва категорії). Методи класу:

* сеттери (для введення даних): setId(), setInfo();
* методи для обробки даних в базі даних: Insert(), Remove(), Update().

Дочірній клас Ingredientu класу Strava\_Napoi містить атрибути: count (кількість інгредієнтів), vumir (собівартість), name( назва інгредієнту). Методи класу:

* сеттери (для введення даних): setId(), setInfo();
* методи для обробки даних в базі даних: Insert(), Remove(), Update().

### 3.4.2 Інструкція користувачеві

Для використання даного програмного забезпечення (ПЗ), є різний рівень доступ такий як доступ для офіціанта, він має лише можливість оформляти замовлення, та доступ для адміністратор, він має повний доступ до програмного забезпечення.

Для запуски програмного забезпечення потрібно запустити файл «E-menu.exe»

Для входу в ПЗ користувач має ввести логін та пароль, вікно «Авторизація» зображено на рисунку 3.6

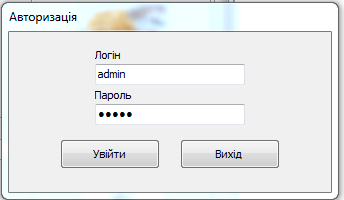


Рисунок 3.6 – Вікно «Авторизація»

Офіціант працює з головним вікном яка зображена на рисунку 3.7. При оформлення замовлення, офіціант для додавання пункту меню повинен натиснути з лівої сторони на страву стільки разів скільки має. Для зручності пошуку страви реалізована фільтрації за категорією страви, для цього треба вибрати категорію з спливаючого меню. бути в замовлені. При відмінні певного пункту замовлення, офіціант видаляє його натискаючи на кнопку «Видалити пункт меню». Також після того як замовлення було здійснено офіціант має провести чек натиснувши на кнопку «Провести замовлення», після чого роздрукується рахунок замовлення, у випадку якщо відвідувач відмовився від замовлення очищує замовлення натискаючи на кнопку «Обнулити замовлення».

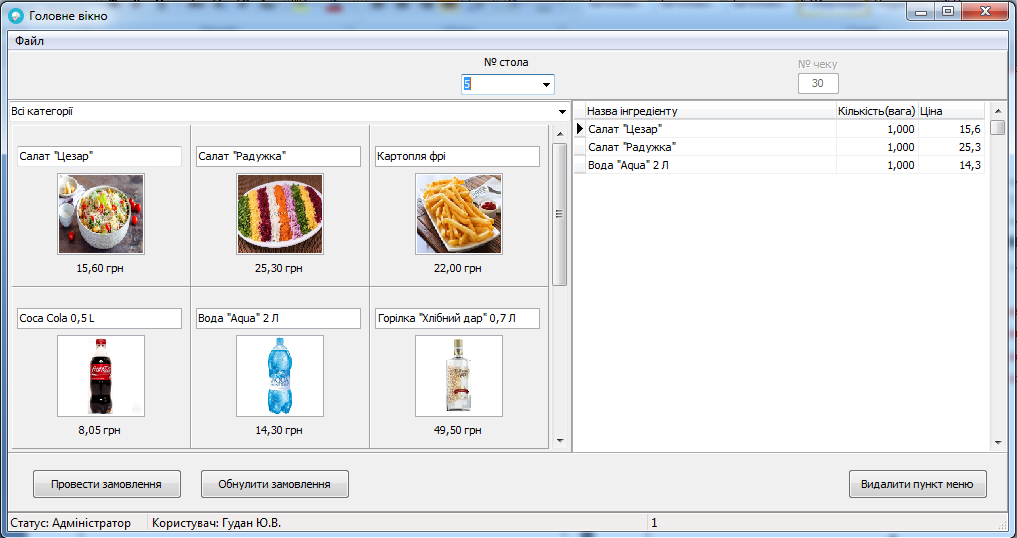


Рисунок 3.7 – Головне вікно

Адміністратор може працювати так само з головним вікном і з вікном «Адміністрування». На вікні «Адміністрування», адміністратор може обробляти всі дані, тобто додавати, редагувати та видаляти. Для того щоб добавити нові дані, в залежності які, йому треба натиснути кнопку «Новий запис», після чого він вводить потрібно йому інформацію в текстові поля та натискає кнопку «Добавити запис», вікно додавання інформації про інгредієнт страви зображений на рисунку 3.8, для редагування інформації адміністратор змінює в текстових полях потрібну йому інформацію та натискає кнопку «Зберегти запис», для видалення запису адміністратор вибирає, який запис йому потрібно видалити та натискає кнопку «Видалити запис». Адміністратор може переглянути вміст старого замовлення, натискаючи на кнопку «Перегляд чеку», також він має можливість управляти користувачами даного ПЗ, даючи їм певний доступ до програми.

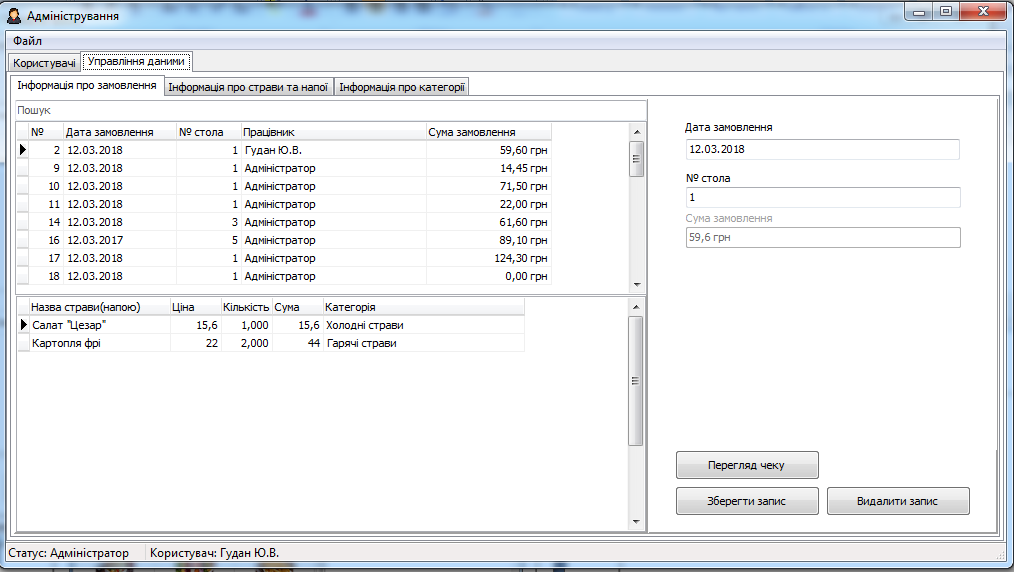


Рисунок 3.7 – Вікно «Адміністрування»

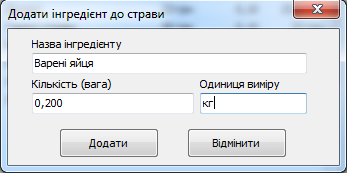


Рисунок 3.8 – Вікно «Додати інгредієнт до страви»

В даному розділі описано розгорнуто інструкцію програміста та користувача, інтерфейс програмного забезпечення та засоби розробки для створення даного програмного забезпечення.

# Розділ 4. Тестування програмного забезпечення

В даному розділі проведено тестування програмного продукту, на коректність його роботи. Протестуємо програмне забезпечення адміністраторської та користувацької частини. Нижче наведено приклади тестування.

## 4.1 Тестування адміністраторської частини

Для перевірки коректного входу в програмне забезпечення для адміністратора, увійдемо у систему під логіном «admin» та паролем «admin». Результат входу можна побачити на рисунку 4.1.

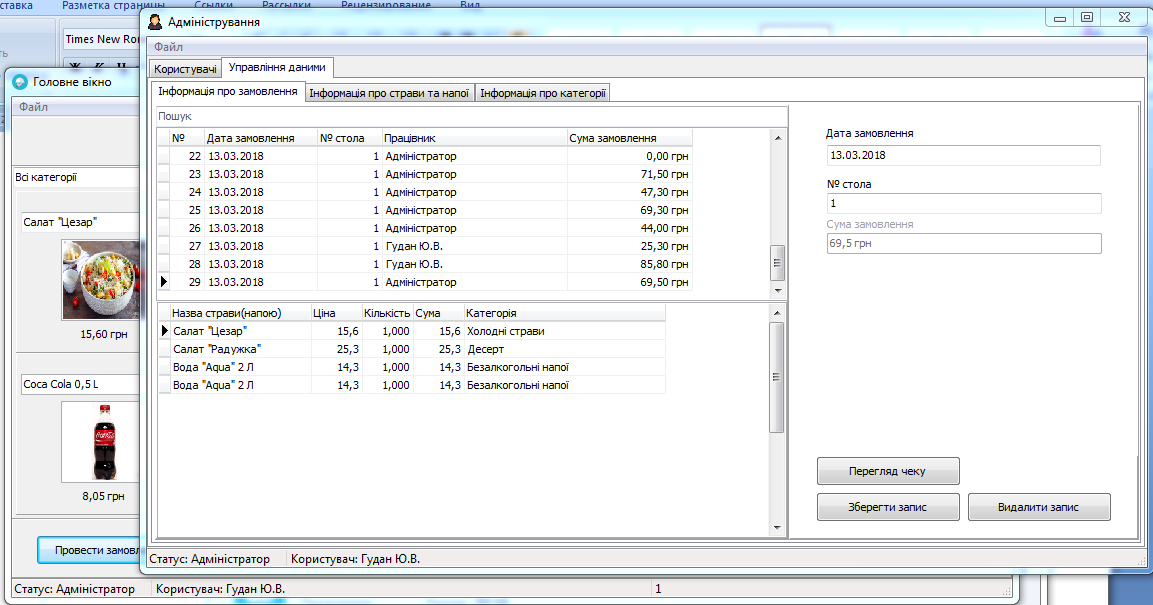


Рисунок 4.1 – Вхід в програмне забезпечення

В результаті перевірки, можна побачити в інформаційній панелі статус «Адміністратор», тобто надався доступ до повної обробки даних.

## 4.2Тестування клієнтської частини

Для перевірки коректної роботи програми під час оформлення замовлення, додамо в чек пару страв або напоїв та подивимось чи інформація з рахунку замовлення співпала з інформацією поточного замовлення. Результат тестування можна побачити на рисунку 4.2.

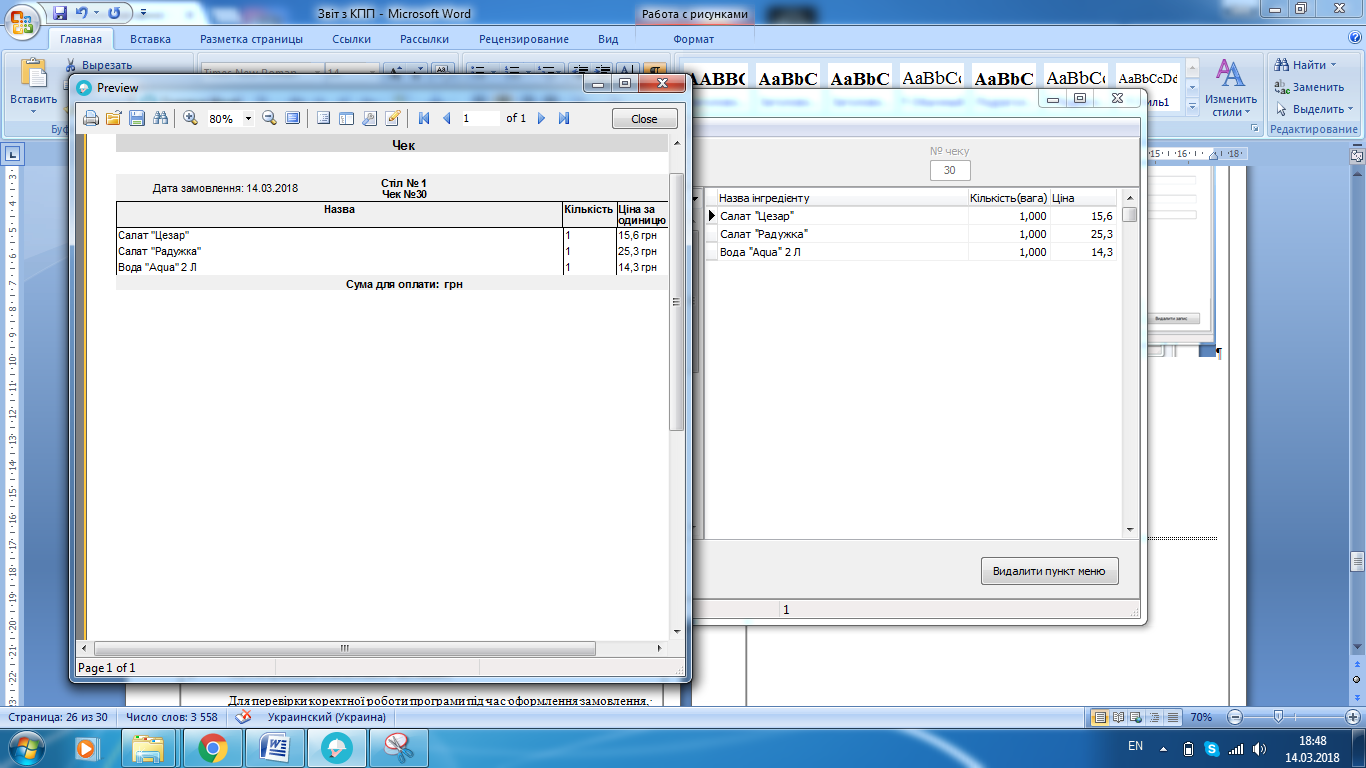


Рисунок 4.2 – Перевірка схожості

В результаті перевірки бачимо, що дані співпадають

## 4.3 Тестування користувацького інтерфейсу

Перевіримо на правильність переходу між вікнами, для цього з головного вікна перейдемо на вікно «Адміністрування». Результат тестування можна побачити на рисунку 4.3.

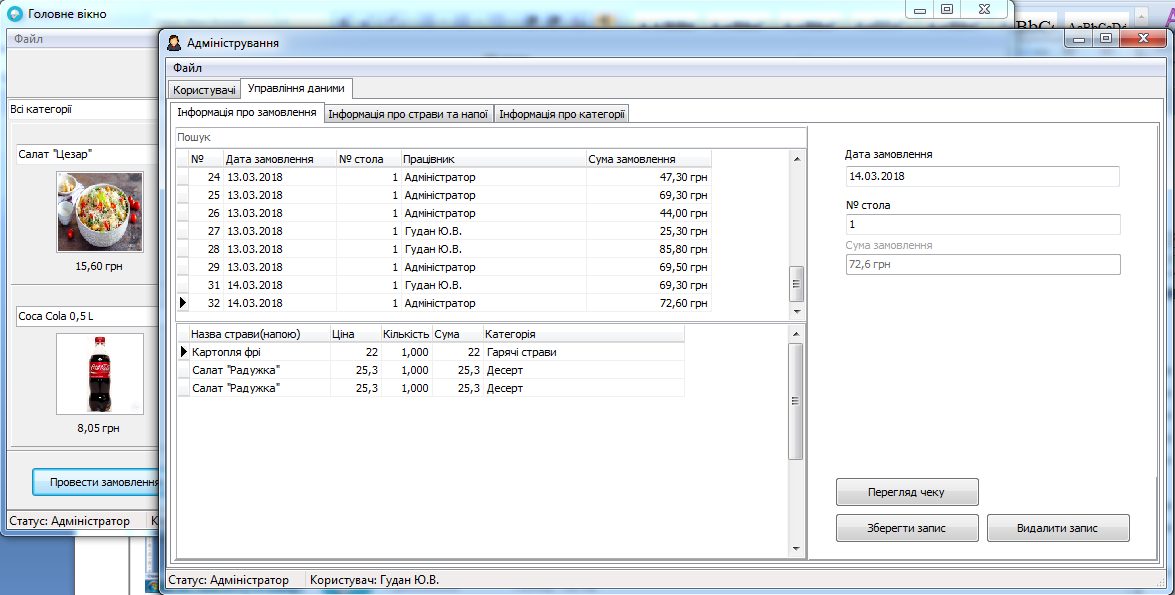


Рисунок 4.3 – Перевірка переходу між формами

## 4.4 Тестування роботи бази даних

Для перевірки чи додається правильно інформація в базу даних, додамо нову категорію з назвою «Гарніри». Результати тестування можна побачити на рисунку 4.4.

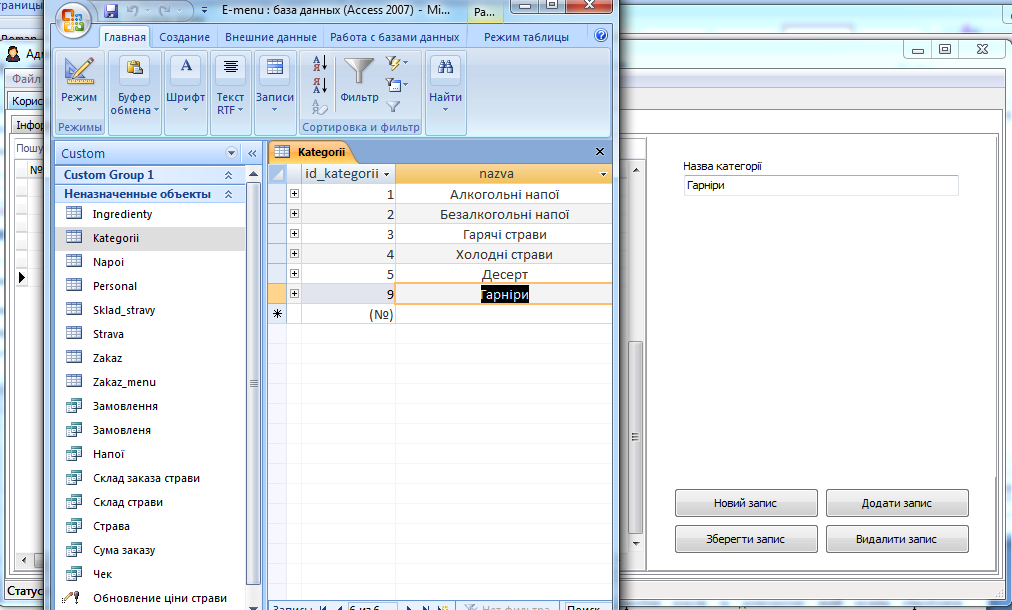


Рисунок 4.4 – Перевірка на додавання в базу даних

В даному розділі було проведено тестування над програмним забезпеченням, в результаті чого програма працює коректно. Отже, даний програмний продукт можна використовувати.

# Висновки

Результатом роботи над комплексним курсовим продуктом є розроблена програмного забезпечення, для автоматизування функції роботи з інформацією адміністратора та офіціанта в закладах громадського харчування. Наведено детальну інструкцію користувача по експлуатації даної автоматизованої інформаційної-системи, описано методику проведення випробувань, яка показує можливість її введення в експлуатацію.

Для розробки автоматизована інформаційна системи було використана середовища Embarcadero RAD Studio Delphi XE7. Структура бази даних була розроблена в Microsoft Access 2007.

Робота над завданням курсового проекту дала змогу розв’язати задачу наближену до реальної.

В результаті розробки програмного забезпечення, набув навичок у розробки програмного забезпечення з використанням принципу «Об’єктно-орієнтованого програмування», розробки класів за допомогою який можна обробляти інформацію в локальній базі даних, вдосконалив свої знання у реалізації структури бази даних.

# Перелікджерел

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 RAD STUDIO XE5 С++BUILDER <http://docwiki.embarcadero.com/Libraries/XE5/en/Main_Page>
4. Короткий довідник sql запитів <http://databases.about.com/od/development/>
5. Форум з зразками sql та с++ функцій http://www.cyberforum.ru
6. Практичне використання реактивного програмування в програмуванні систем <http://habrahabr.ru/post/140719/>.

# додатки

**Додаток А**

**Тексти програмного коду**

А.1 Main.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Main.h"

#include "Admin.h"

#include "Autorization.h"

#include "DataModule.h"

#include "Hiderfile.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFMain \*FMain;

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

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

: TForm(Owner)

{

}

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

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

{

FAutorization->Show();

FAutorization->LELogin->SetFocus();

DM->DBKategorii->First();

ComboBox1->Items->Add("Всі категорії");

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

ComboBox1->Items->Add(DM->DBKategorii->FieldByName("nazva")->AsString);

DM->DBKategorii->Next();

}

DM->DBKategorii->First();

ComboBox1->ItemIndex = -1;

DM->DBZakaz->Last();

LEChecknumber->Text = DM->DBZakaz->FieldByName("id\_zakaza")->AsInteger + 1;

}

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

void \_\_fastcall TFMain::ComboBox1Change(TObject \*Sender)

{

if (ComboBox1->Text != "Всі категорії") {

DM->DBGolovna->Filter = "Kategorii.nazva like '"+ComboBox1->Text+"'";

DM->DBGolovna->Filtered = true;

} else DM->DBGolovna->Filtered = false;

}

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

void \_\_fastcall TFMain::LEChecknumberChange(TObject \*Sender)

{

if(LEChecknumber->Text != ""){

DM->DBZakaz\_check->Filter = "kod\_zakaza = "+LEChecknumber->Text;

DM->DBZakaz\_check->Filtered = true;

} else DM->DBZakaz\_check->Filtered = false;

}

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

void \_\_fastcall TFMain::DBImage1DblClick(TObject \*Sender)

{

Zakaz.SetIdStrava(DM->DBGolovnaid\_stravu->Value);

Zakaz.SetIdZakaz(StrToInt(LEChecknumber->Text));

Zakaz.SetCount(1);

Zakaz.InsertPunktZakaz();

}

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

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

{

Zakaz.SetInfoZakaz(StrToInt(ComboBox2->Text),Date());

Zakaz.SetIdPersonal(StrToInt(StatusBar1->Panels->Items[2]->Text));

Zakaz.InsertZakaz();

Zakaz.CalculationPrice();

DM->Check->Filtered = false;

DM->Check->Filter = "id\_zakaza = "+ LEChecknumber->Text;

DM->Check->Filtered = true;

DM->DBZakaz->Filtered;

DM->DBZakaz->Filter = "id\_zakaza ="+LEChecknumber->Text;

DM->DBZakaz->Filtered = true;

DM->frxReport1->ShowReport();

DM->DBZakaz->Last();

LEChecknumber->Text = DM->DBZakaz->FieldByName("id\_zakaza")->AsInteger + 1;

DM->DBZakaz->Filtered = false;

}

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

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

{

Zakaz.SetIdZakaz(StrToInt(LEChecknumber->Text));

Zakaz.RemovePunktuZakaz();

}

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

void \_\_fastcall TFMain::Button3Click(TObject \*Sender)

{

Zakaz.SetIdZakaz(DM->DBZakaz\_checkkod\_zakaza->Value);

Zakaz.SetIdStrava(DM->DBZakaz\_checkkod\_strava->Value);

Zakaz.RemovePunktZakaz();

}

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

A.2 Admin.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Admin.h"

#include "Autorization.h"

#include "DataModule.h"

#include "Hiderfile.h"

#include "Main.h"

#include <Math.h>

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFAdmin \*FAdmin;

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

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

: TForm(Owner)

{

}

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

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

{

Kategoriya.SetInfoKategoriya(K\_Kategoriya\_name->Text);

Kategoriya.InsertKategoriya();

}

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

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

{

Kategoriya.SetIdKategoriya(DM->DBKategoriiid\_kategorii->Value);

Kategoriya.SetInfoKategoriya(K\_Kategoriya\_name->Text);

Kategoriya.UpdateKategoriya();

}

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

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

{

K\_Kategoriya\_name->Clear();

K\_Kategoriya\_name->SetFocus();

Kategoriya.SetIdKategoriya(0);

}

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

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

{

Kategoriya.SetIdKategoriya(DM->DBKategoriiid\_kategorii->Value);

Kategoriya.DeleteKategoriya();

}

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

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

{

K\_Personal\_pib->Clear();

K\_Personal\_login->Clear();

K\_Personal\_password->Clear();

K\_Personal\_Admin->Checked = false;

K\_Personal\_pib->SetFocus();

}

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

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

{

Personal.SetIdPersonal(DM->DBPersonalid\_personal->Value);

Personal.SetInfoPersonal(K\_Personal\_pib->Text,K\_Personal\_login->Text,K\_Personal\_password->Text,K\_Personal\_Admin->Checked);

Personal.InsertPersonal();

}

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

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

{

Personal.SetIdPersonal(DM->DBPersonalid\_personal->Value);

Personal.SetInfoPersonal(K\_Personal\_pib->Text,K\_Personal\_login->Text,K\_Personal\_password->Text,K\_Personal\_Admin->Checked);

Personal.UpdatePersonal();

}

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

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

{

Personal.SetIdPersonal(DM->DBPersonalid\_personal->Value);

Personal.RemovePersonal();

}

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

void \_\_fastcall TFAdmin::DownNapoiFotoMouseDown(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

DownNapoiFoto->Font->Color = clPurple;

}

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

void \_\_fastcall TFAdmin::DownNapoiFotoMouseUp(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

DownNapoiFoto->Font->Color = clBlue;

}

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

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

{

if(OpenPictureDialog1->Execute()) ImageNapoiFoto->Picture->LoadFromFile(OpenPictureDialog1->FileName);

}

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

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

{

if(OpenPictureDialog1->Execute()) ImageStravaFoto->Picture->LoadFromFile(OpenPictureDialog1->FileName);

/\*TBitmap \*tmpimg = new TBitmap;

tmpimg->LoadFromFile(OpenPictureDialog1->FileName);

DM->DBStrava->Edit();

DM->DBStrava->FieldByName("foto")->Assign(tmpimg);

DM->DBStrava->Post(); \*/

}

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

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

{

K\_Napoi\_name->Clear();

K\_Napoi\_Kategoriya->Clear();

K\_Napoi\_Kategoriya\_Id->Clear();

K\_Napoi\_Data->Clear();

K\_Napoi\_cost\_price->Text = "0";

K\_Napoi\_price->Text = "0";

K\_Napoi\_natsinka->Text = "0";

DM->DBKategorii->First();

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

FAdmin->K\_Napoi\_Kategoriya->Items->Add(DM->DBKategoriinazva->Value);

FAdmin->K\_Napoi\_Kategoriya\_Id->Items->Add(DM->DBKategorii->FieldByName("id\_kategorii")->AsString);

DM->DBKategorii->Next();

}

FAdmin->K\_Napoi\_Kategoriya->ItemIndex = 0;

FAdmin->K\_Napoi\_Kategoriya\_Id->ItemIndex = 0;

ImageStravaFoto->Picture->LoadFromFile(GetCurrentDir()+"/Image/Picture\_Foto.bmp");

}

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

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

{

K\_Strava\_name->Clear();

K\_Strava\_Kategoriya->Clear();

K\_Strava\_Kategoriya\_Id->Clear();

K\_Strava\_Data->Clear();

K\_Strava\_cost\_price->Text = "0";

K\_Strava\_price->Text = "0";

K\_Strava\_natsinka->Text = "0";

DM->DBKategorii->First();

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

FAdmin->K\_Strava\_Kategoriya->Items->Add(DM->DBKategoriinazva->Value);

FAdmin->K\_Strava\_Kategoriya\_Id->Items->Add(DM->DBKategorii->FieldByName("id\_kategorii")->AsString);

DM->DBKategorii->Next();

}

FAdmin->K\_Strava\_Kategoriya->ItemIndex = 0;

FAdmin->K\_Strava\_Kategoriya\_Id->ItemIndex = 0;

ImageStravaFoto->Picture->LoadFromFile(GetCurrentDir()+"/Image/Picture\_Foto.bmp");

}

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

void \_\_fastcall TFAdmin::K\_Napoi\_KategoriyaChange(TObject \*Sender)

{

K\_Napoi\_Kategoriya\_Id->ItemIndex = K\_Napoi\_Kategoriya->ItemIndex;

}

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

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

{

Napiy.SetInfoNapiy(K\_Napoi\_name->Text,K\_Napoi\_Data->Text,StrToFloat(K\_Napoi\_cost\_price->Text),StrToFloat(K\_Napoi\_price->Text),StrToFloat(K\_Napoi\_natsinka->Text));

Napiy.SetIdKategoriya(StrToInt(K\_Napoi\_Kategoriya\_Id->Text));

Napiy.SetFotoNapiy(ImageNapoiFoto->Picture->Bitmap);

Napiy.InsertNapiy();

}

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

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

{

Strava.SetIdStrava(DM->DBStravaid\_stravu->Value);

Strava.SetIdKategoriya(StrToInt(K\_Strava\_Kategoriya\_Id->Text));

Strava.SetFotoStrava(ImageStravaFoto->Picture->Bitmap);

Strava.SetInfoStrava(K\_Strava\_name->Text,K\_Strava\_Data->Text,StrToFloat(K\_Strava\_cost\_price->Text),StrToFloat(K\_Strava\_price->Text),StrToFloat(K\_Strava\_natsinka->Text));

Strava.UpdateStrava();

}

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

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

{

Strava.SetInfoStrava(K\_Strava\_name->Text,K\_Strava\_Data->Text,StrToFloat(K\_Strava\_cost\_price->Text),StrToFloat(K\_Strava\_price->Text),StrToFloat(K\_Strava\_natsinka->Text));

Strava.SetIdKategoriya(StrToInt(K\_Strava\_Kategoriya\_Id->Text));

Strava.SetFotoStrava(ImageStravaFoto->Picture->Bitmap);

Strava.InsertStrava();

}

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

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

{

Strava.SetIdStrava(DM->DBStravaid\_stravu->Value);

Strava.RemoveStrava();

}

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

void \_\_fastcall TFAdmin::K\_Strava\_KategoriyaChange(TObject \*Sender)

{

K\_Strava\_Kategoriya\_Id->ItemIndex = K\_Strava\_Kategoriya->ItemIndex;

}

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

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

{

DM->DBKategorii->First();

DM->DBStrava->First();

DM->DBPersonal->First();

}

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

void \_\_fastcall TFAdmin::K\_Strava\_cost\_priceChange(TObject \*Sender)

{

try {

if (K\_Strava\_cost\_price->Text != "" && K\_Strava\_natsinka->Text !="") {

float cost = StrToFloat(K\_Strava\_cost\_price->Text),natsinka = StrToFloat(K\_Strava\_natsinka->Text);

float rezult = 0;

rezult = cost + (cost \* natsinka);

K\_Strava\_price->Text = rezult;

} else K\_Strava\_price->Text = 0;

} catch (...) {

}

}

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

void \_\_fastcall TFAdmin::K\_Strava\_natsinkaChange(TObject \*Sender)

{

K\_Strava\_cost\_priceChange(Sender);

}

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

void \_\_fastcall TFAdmin::K\_Napoi\_cost\_priceChange(TObject \*Sender)

{

try {

if (K\_Napoi\_cost\_price->Text != "" && K\_Napoi\_natsinka->Text !="") {

float cost = StrToFloat(K\_Napoi\_cost\_price->Text),natsinka = StrToFloat(K\_Napoi\_natsinka->Text);

float rezult = 0;

rezult = cost + (cost \* natsinka);

K\_Napoi\_price->Text = rezult;

} else K\_Napoi\_price->Text = 0;

} catch (...) {

}

}

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

void \_\_fastcall TFAdmin::K\_Napoi\_natsinkaChange(TObject \*Sender)

{

K\_Napoi\_cost\_priceChange(Sender);

}

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

void \_\_fastcall TFAdmin::ImageStravaFotoMouseDown(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

Panel16->Width = 256;

Panel16->Height = 256;

}

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

void \_\_fastcall TFAdmin::ImageStravaFotoMouseUp(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

Panel16->Width = 64;

Panel16->Height = 64;

}

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

void \_\_fastcall TFAdmin::Button4Click(TObject \*Sender)

{

Zakaz.SetIdZakaz(DM->DBZakazid\_zakaza->Value);

Zakaz.SetInfoZakaz(StrToInt(K\_Zakaz\_nomer\_stola->Text),StrToDate(K\_Zakaz\_Date->Text));

Zakaz.UpdateZakaz();

}

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

void \_\_fastcall TFAdmin::Button3Click(TObject \*Sender)

{

Zakaz.SetIdZakaz(DM->DBZakazid\_zakaza->Value);

Zakaz.RemoveZakaz();

Zakaz.RemovePunktZakaz();

}

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

void \_\_fastcall TFAdmin::Edit3Change(TObject \*Sender)

{

String tmp = Edit3->Text;

DM->DBKategorii->Filtered = false;

if (Edit3->Text!="") {

DM->DBKategorii->Filter="nazva like '%"+tmp+"%'";

DM->DBKategorii->Filtered = true;

}

}

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

void \_\_fastcall TFAdmin::Edit1Change(TObject \*Sender)

{

String tmp = Edit1->Text;

DM->DBStrava->Filtered = false;

if (Edit1->Text!="") {

DM->DBStrava->Filter="Strava.nazva like '%"+tmp+"%' or Kategorii.nazva like '%"+tmp+"%'";

DM->DBStrava->Filtered = true;

}

}

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

void \_\_fastcall TFAdmin::Edit4Change(TObject \*Sender)

{

String tmp = Edit4->Text;

DM->DBZakaz->Filtered = false;

if (Edit4->Text!="") {

DM->DBZakaz->Filter="pib like '%"+tmp+"%'";

DM->DBZakaz->Filtered = true;

}

}

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

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

{

String tmp = EditPersonal->Text;

DM->DBPersonal->Filtered = false;

if (EditPersonal->Text!="") {

DM->DBPersonal->Filter="pib like '%"+tmp+"%' or login like '%"+tmp+"%'";

DM->DBPersonal->Filtered = true;

}

}

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

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

{

DM->Check->Filtered = false;

DM->Check->Filter = "id\_zakaza = "+ DM->DBZakaz->FieldByName("id\_zakaza")->AsString;

DM->Check->Filtered = true;

DM->frxReport1->ShowReport();

}

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

A.3 Autorization.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Autorization.h"

#include "Admin.h"

#include "DataModule.h"

#include "Hiderfile.h"

#include "Main.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFAutorization \*FAutorization;

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

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

: TForm(Owner)

{

}

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

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

{

FMain->Close();

}

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

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

{

if (Zakaz.Initsialization(LELogin->Text,LEPassword->Text)) {

FAutorization->Hide();

FMain->Enabled = true;

FMain->Show();

} else { ShowMessage("Неправильний логін або пароль"); FMain->Enabled = false; LELogin->SetFocus();}

LELogin->Clear();

LEPassword->Clear();

}

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

void \_\_fastcall TFAutorization::LEPasswordKeyPress(TObject \*Sender, System::WideChar &Key)

{

if (Key == 13) Button1Click(Sender);

}

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

A.4 AddIngredienty.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "AddIngredienty.h"

#include "Admin.h"

#include "Autorization.h"

#include "DataModule.h"

#include "Hiderfile.h"

#include "Main.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFAddIngredienty \*FAddIngredienty;

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

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

: TForm(Owner)

{

}

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

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

{

K\_Ingredient\_kilkisti->Text = 0;

K\_Ingredient\_name->Clear();

K\_Ingredient\_vumir->Clear();

FAddIngredienty->Hide();

}

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

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

{

Button2Click(Sender);

}

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

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

{

Ingredienty.SetInfoIngredienty(K\_Ingredient\_name->Text,K\_Ingredient\_vumir->Text,StrToFloat(K\_Ingredient\_kilkisti->Text));

Ingredienty.InsertIngredienty();

K\_Ingredient\_kilkisti->Text = 0;

K\_Ingredient\_name->Clear();

K\_Ingredient\_vumir->Clear();

FAddIngredienty->Hide();

}

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

A.5 DataModule.cpp

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

#pragma hdrstop

#include "DataModule.h"

#include "Admin.h"

#include "Autorization.h"

#include "Hiderfile.h"

#include "Main.h"

#include "AddIngredienty.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)

{

}

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

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

{

FMain->Close();

}

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

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

{

FAutorization->Show();

FMain->Enabled = false;

FAdmin->Enabled = false;

FAutorization->LELogin->SetFocus();

}

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

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

{

ADOConnection->Connected = false;

ADOConnection->ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source="+GetCurrentDir()+"/E-menu.accdb;Persist Security Info=False";

ADOConnection->Connected = true;

onovlenya();

}

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

//-------------------------Оновлення-----------------------------------------

void TDM::onovlenya(){

ActiveDB(DBPersonal);

ActiveDB(DBKategorii);

ActiveDB(DBStrava);

ActiveDB(DBIngredienty);

ActiveDB(DBGolovna);

ActiveDB(DBZakaz);

ActiveDB(DBZakaz\_strava);

ActiveDB(DBZakaz\_check);

ActiveDB(DBIngred);

ActiveDB(Check);

}

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

//-------------------Клас персонал-------------------------------------------

TPersonal::TPersonal(){

this->id\_personal = 0;

this->login = "Login";

this->password = "Password";

this->admin = false;

}

void TPersonal::SetIdPersonal(int id){

this->id\_personal = id;

}

void TPersonal::SetInfoPersonal(String \_pib, String \_login, String \_password,bool \_admin){

this->pib = \_pib;

this->login = \_login;

this->password = \_password;

this->admin = \_admin;

}

bool TPersonal::GetAdmin(){

return this->admin;

}

int TPersonal::GetIdPersonal(){

return this->id\_personal;

}

bool TPersonal::Initsialization(String lg,String ps){

DM->DBPersonal->First();

bool f = false;

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

if (lg == DM->DBPersonallogin->AsString && ps == DM->DBPersonalpassword->AsString) {

f = true;

this->SetIdPersonal(DM->DBPersonalid\_personal->Value);

this->SetInfoPersonal(DM->DBPersonalpib->Value,DM->DBPersonallogin->Value,DM->DBPersonalpassword->Value,DM->DBPersonaladmin->Value);

FMain->StatusBar1->Panels->Items[2]->Text = this->id\_personal;

break;

}

DM->DBPersonal->Next();

}

if (this->admin)this->StatusAdmin(); else this->StatusUser();

if (f) return true; else return false;

}

void TPersonal::StatusAdmin(){

DM->NAdmin14->Visible = true;

FAdmin->Enabled = true;

FAdmin->StatusBar1->Panels->Items[0]->Text = "Статус: Адміністратор";

FAdmin->StatusBar1->Panels->Items[1]->Text = "Користувач: "+this->pib;

FMain->StatusBar1->Panels->Items[0]->Text = "Статус: Адміністратор";

FMain->StatusBar1->Panels->Items[1]->Text = "Користувач: "+this->pib;

}

void TPersonal::StatusUser(){

FMain->StatusBar1->Panels->Items[0]->Text = "Статус: Користувач";

FMain->StatusBar1->Panels->Items[1]->Text = "Користувач: "+this->pib;

DM->NAdmin14->Visible = false;

FAdmin->Enabled = false;

FAdmin->Hide();

}

void TPersonal::InsertPersonal(){

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

DM->TSQL->SQL->Add("INSERT INTO Personal( pib, login, [password], admin)");

DM->TSQL->SQL->Add("VALUES(:P1,:P2,:P3,:P4)");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->pib;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->login;

DM->TSQL->Parameters->ParamByName("P3")->Value = this->password;

DM->TSQL->Parameters->ParamByName("P4")->Value = this->admin;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TPersonal::UpdatePersonal(){

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

DM->TSQL->SQL->Add("UPDATE Personal SET Personal.pib = :P1, Personal.login = :P2, Personal.[password] = :P3, Personal.admin = :P4");

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

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

DM->TSQL->Parameters->ParamByName("P1")->Value = this->pib;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->login;

DM->TSQL->Parameters->ParamByName("P3")->Value = this->password;

DM->TSQL->Parameters->ParamByName("P4")->Value = this->admin;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TPersonal::RemovePersonal(){

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

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

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

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

//------------------------------Клас Категорія--------------------------------

TKategoriya::TKategoriya(){

this->id\_kategoriya = 0;

this->name\_kategoriya = "Kategoriya";

}

void TKategoriya::SetIdKategoriya(int \_id){

this->id\_kategoriya = \_id;

}

void TKategoriya::SetInfoKategoriya(String name){

this->name\_kategoriya = name;

}

void TKategoriya::InsertKategoriya(){

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

DM->TSQL->SQL->Add("INSERT INTO Kategorii(nazva)");

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

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TKategoriya::UpdateKategoriya(){

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

DM->TSQL->SQL->Add("UPDATE Kategorii SET Kategorii.nazva = :P2");

DM->TSQL->SQL->Add("WHERE (((Kategorii.id\_kategorii) = :P1));");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->id\_kategoriya;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->name\_kategoriya;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TKategoriya::DeleteKategoriya(){

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

DM->TSQL->SQL->Add("DELETE Kategorii.id\_kategorii FROM Kategorii WHERE (((Kategorii.id\_kategorii)=:id\_delete));");

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

//------------------------------Клас Інгредієнти------------------------------

TIngredienty::TIngredienty(){

this->id\_ingredient = 0;

this->names = "Name";

this->vumir = "Vumir";

this->count = 0;

}

void TIngredienty::SetIdIngredienty(int id){

this->id\_ingredient = id;

}

void TIngredienty::SetInfoIngredienty(String name,String vum,float kil){

this->names = name;

this->vumir = vum;

this->count = kil;

}

void TIngredienty::InsertIngredienty(){

int id\_strava = DM->DBStrava->FieldByName("id\_stravu")->AsInteger;

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

DM->TSQL->SQL->Add("INSERT INTO Ingredienty(nazva,vumir)");

DM->TSQL->SQL->Add("VALUES(:P1,:P2)");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->names;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->vumir;

DM->TSQL->ExecSQL();

DM->onovlenya();

DM->DBIngred->Last();

int id\_ing = DM->DBIngred->FieldByName("id\_ingredienta")->AsInteger;

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

DM->TSQL->SQL->Add("INSERT INTO Sklad\_stravy(kod\_strava,kod\_ingredienta,kilkisti)");

DM->TSQL->SQL->Add("VALUES(:P1,:P2,:P3)");

DM->TSQL->Parameters->ParamByName("P1")->Value = id\_strava;

DM->TSQL->Parameters->ParamByName("P2")->Value = id\_ing;

DM->TSQL->Parameters->ParamByName("P3")->Value = this->count;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TIngredienty::RemoveIngredienty(){

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

DM->TSQL->SQL->Add("DELETE id\_ingredienta FROM Ingredienty WHERE ((id\_ingredienta=:id\_delete));");

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

//------------------------------Клас Напій------------------------------------

TNapiy::TNapiy(){

this->id\_napiy = 0;

this->names\_napiy = "Name napiy";

this->data\_napiy = "Data napiy";

this->cost\_price\_napiy = 0.0;

this->price\_napiy = 0.0;

this->natsinka\_napiy = 0.0;

this->foto\_napiy = new TBitmap;

}

void TNapiy::SetIdNapiy(int id){

this->id\_napiy = id;

}

void TNapiy::SetInfoNapiy(String name,String data,float cost,float price ,float natsinka){

this->names\_napiy = name;

this->data\_napiy = data;

this->cost\_price\_napiy = cost;

this->price\_napiy = price;

this->natsinka\_napiy = natsinka;

}

void TNapiy::InsertNapiy(){

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

DM->TSQL->SQL->Add("INSERT INTO Napoi(nazva,kod\_kategoriy,cost\_price,price,natsinka,data,foto)");

DM->TSQL->SQL->Add("VALUES(:P1,:P2,:P3,:P4,:P5,:P6,:P7)");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->names\_napiy;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->id\_kategoriya;

DM->TSQL->Parameters->ParamByName("P3")->Value = this->cost\_price\_napiy;

DM->TSQL->Parameters->ParamByName("P4")->Value = this->price\_napiy;

DM->TSQL->Parameters->ParamByName("P5")->Value = this->natsinka\_napiy;

DM->TSQL->Parameters->ParamByName("P6")->Value = this->data\_napiy;

DM->TSQL->Parameters->ParamByName("P7")->Assign(this->foto\_napiy);

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TNapiy::UpdateNapiy(){

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

DM->TSQL->SQL->Add("UPDATE Napoi SET nazva = :P2,cost\_price = :P4,price = :P5, natsinka = :P6,data = :P7,foto = :P8, kod\_kategoriy = :P9 ");

DM->TSQL->SQL->Add("WHERE Napoi.id\_napoi = :P1;");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->id\_napiy;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->names\_napiy;

DM->TSQL->Parameters->ParamByName("P4")->Value = this->cost\_price\_napiy;

DM->TSQL->Parameters->ParamByName("P5")->Value = this->price\_napiy;

DM->TSQL->Parameters->ParamByName("P6")->Value = this->natsinka\_napiy;

DM->TSQL->Parameters->ParamByName("P7")->Value = this->data\_napiy;

DM->TSQL->Parameters->ParamByName("P8")->Assign(this->foto\_napiy);

DM->TSQL->Parameters->ParamByName("P9")->Value = this->id\_kategoriya;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TNapiy::RemoveNapiy(){

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

DM->TSQL->SQL->Add("DELETE Napoi.id\_napoi FROM Napoi WHERE (((Napoi.id\_napoi)=:id\_delete));");

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TNapiy::SetFotoNapiy(TBitmap\* tmp){

this->foto\_napiy = tmp;

}

//------------------------------Клас Страва-----------------------------------

TStrava::TStrava(){

this->id\_strava = 0;

this->names\_strava = "Name strava";

this->data\_strava = "Data napiy";

this->cost\_price\_strava = 0.0;

this->price\_strava = 0.0;

this->natsinka\_strava = 0.0;

this->foto\_strava = new TBitmap;

}

void TStrava::SetIdStrava(int id){

this->id\_strava = id;

}

void TStrava::SetInfoStrava(String name,String data,float cost,float price ,float natsinka){

this->names\_strava = name;

this->data\_strava = data;

this->cost\_price\_strava = cost;

this->price\_strava = price;

this->natsinka\_strava = natsinka;

}

void TStrava::InsertStrava(){

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

DM->TSQL->SQL->Add("INSERT INTO Strava(nazva,kod\_kategorii,cost\_price,price,natsinka,data,foto)");

DM->TSQL->SQL->Add("VALUES(:P1,:P2,:P3,:P4,:P5,:P6,:P7)");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->names\_strava;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->id\_kategoriya;

DM->TSQL->Parameters->ParamByName("P3")->Value = this->cost\_price\_strava;

DM->TSQL->Parameters->ParamByName("P4")->Value = this->price\_strava;

DM->TSQL->Parameters->ParamByName("P5")->Value = this->natsinka\_strava;

DM->TSQL->Parameters->ParamByName("P6")->Value = this->data\_strava;

DM->TSQL->Parameters->ParamByName("P7")->Assign(this->foto\_strava);

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TStrava::UpdateStrava(){

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

DM->TSQL->SQL->Add("UPDATE Strava SET kod\_kategorii = :P9, nazva = :P2,cost\_price = :P4,price = :P5, natsinka = :P6,data = :P7, foto = :P8");

DM->TSQL->SQL->Add("WHERE Strava.id\_stravu = :P1;");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->id\_strava;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->names\_strava;

DM->TSQL->Parameters->ParamByName("P4")->Value = this->cost\_price\_strava;

DM->TSQL->Parameters->ParamByName("P5")->Value = this->price\_strava;

DM->TSQL->Parameters->ParamByName("P6")->Value = this->natsinka\_strava;

DM->TSQL->Parameters->ParamByName("P7")->Value = this->data\_strava;

DM->TSQL->Parameters->ParamByName("P8")->Assign(this->foto\_strava);

DM->TSQL->Parameters->ParamByName("P9")->Value = this->id\_kategoriya;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TStrava::RemoveStrava(){

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

DM->TSQL->SQL->Add("DELETE Strava.id\_stravu FROM Strava WHERE (((Strava.id\_stravu)=:id\_delete));");

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TStrava::SetFotoStrava(TBitmap\* tmp){

this->foto\_strava = tmp;

}

//------------------------------клас Замовлення-----------------------------

TZakaz::TZakaz(){

this->id\_zakaz = 0;

this->nomer\_stola = 0;

this->dates = Date();

this->count = 1;

}

void TZakaz::SetIdZakaz(int id){

this->id\_zakaz = id;

}

void TZakaz::SetCount(float con){

this->count = con;

}

void TZakaz::SetInfoZakaz(int stol,TDateTime date){

this->nomer\_stola = stol;

this->dates = date;

}

void TZakaz::InsertZakaz(){

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

DM->TSQL->SQL->Add("INSERT INTO Zakaz(dates,nomer\_stola,kod\_personal)");

DM->TSQL->SQL->Add("VALUES(:P1,:P2,:P3)");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->dates;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->nomer\_stola;

DM->TSQL->Parameters->ParamByName("P3")->Value = this->id\_personal;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TZakaz::UpdateZakaz(){

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

DM->TSQL->SQL->Add("UPDATE Zakaz SET dates = :P2, nomer\_stola = :P3");

DM->TSQL->SQL->Add("WHERE Zakaz.id\_zakaza = :P1;");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->id\_zakaz;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->dates;

DM->TSQL->Parameters->ParamByName("P3")->Value = this->nomer\_stola;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TZakaz::RemoveZakaz(){

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

DM->TSQL->SQL->Add("DELETE Zakaz.id\_zakaza FROM Zakaz WHERE (((Zakaz.id\_zakaza)=:id\_delete));");

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

DM->TSQL->ExecSQL();

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

DM->TSQL->SQL->Add("DELETE Zakaz\_menu.kod\_zakaza FROM Zakaz\_menu WHERE (((Zakaz\_menu.kod\_zakaza)=:id\_delete));");

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TZakaz::RemovePunktZakaz(){

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

DM->TSQL->SQL->Add("DELETE Zakaz\_menu.kod\_zakaza, Zakaz\_menu.kod\_strava FROM Zakaz\_menu");

DM->TSQL->SQL->Add("WHERE (((Zakaz\_menu.kod\_zakaza)=:id\_delete) AND ((Zakaz\_menu.kod\_strava)=:id\_str));");

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

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TZakaz::RemovePunktuZakaz(){

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

DM->TSQL->SQL->Add("DELETE Zakaz\_menu.kod\_zakaza FROM Zakaz\_menu WHERE (((Zakaz\_menu.kod\_zakaza)=:id\_delete));");

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

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TZakaz::InsertPunktZakaz(){

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

DM->TSQL->SQL->Add("INSERT INTO Zakaz\_menu(kod\_zakaza,kod\_strava,kilkisti)");

DM->TSQL->SQL->Add("VALUES(:P1,:P2,:P3)");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->id\_zakaz;

DM->TSQL->Parameters->ParamByName("P2")->Value = this->id\_strava;

DM->TSQL->Parameters->ParamByName("P3")->Value = this->count;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

void TZakaz::CalculationPrice(){

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

DM->TSQL->SQL->Add("UPDATE Strava INNER JOIN (Zakaz INNER JOIN Zakaz\_menu ON Zakaz.id\_zakaza = Zakaz\_menu.kod\_zakaza) ON Strava.id\_stravu = Zakaz\_menu.kod\_strava SET Zakaz\_menu.price = [Strava]![price]\*[Zakaz\_menu]![kilkisti]");

DM->TSQL->SQL->Add("WHERE (((Zakaz\_menu.kod\_zakaza)=:P1));");

DM->TSQL->Parameters->ParamByName("P1")->Value = this->id\_zakaz;

DM->TSQL->ExecSQL();

DM->onovlenya();

}

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

void \_\_fastcall TDM::N6Click(TObject \*Sender)

{

onovlenya();

}

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

void \_\_fastcall TDM::NAdmin14Click(TObject \*Sender)

{

if(FAdmin->Enabled) FAdmin->Show();

}

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

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

{

onovlenya();

}

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

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

{

FMain->Close();

}

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

void \_\_fastcall TDM::DBKategoriiAfterScroll(TDataSet \*DataSet)

{

try {

FAdmin->K\_Kategoriya\_name->Text = DBKategoriinazva->Value;

Kategoriya.SetIdKategoriya(DM->DBKategoriiid\_kategorii->Value);

Kategoriya.SetInfoKategoriya(DBKategoriinazva->Value);

}

catch (...){ }

}

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

void \_\_fastcall TDM::DBPersonalAfterScroll(TDataSet \*DataSet)

{

try {

FAdmin->K\_Personal\_pib->Text = DBPersonalpib->Value;

FAdmin->K\_Personal\_login->Text = DBPersonallogin->Value;

FAdmin->K\_Personal\_password->Text = DBPersonalpassword->Value;

FAdmin->K\_Personal\_Admin->Checked = DBPersonaladmin->Value;

} catch (...) {

}

}

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

void \_\_fastcall TDM::DBStravaAfterScroll(TDataSet \*DataSet)

{

try {

FAdmin->K\_Strava\_name->Text = DBStravaStravanazva->Value;

FAdmin->K\_Strava\_Data->Text = DBStravadata->Value;

FAdmin->K\_Strava\_cost\_price->Text = DBStravacost\_price->Value;

FAdmin->K\_Strava\_natsinka->Text = DBStravanatsinka->Value;

FAdmin->K\_Strava\_price->Text = DBStravaprice->Value;

FAdmin->K\_Strava\_Kategoriya->Clear();

FAdmin->K\_Strava\_Kategoriya\_Id->Clear();

FAdmin->ImageStravaFoto->Picture->Assign(DM->DBStrava->FieldByName("foto"));

DBKategorii->First();

int index = 0;

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

FAdmin->K\_Strava\_Kategoriya->Items->Add(DBKategoriinazva->Value);

FAdmin->K\_Strava\_Kategoriya\_Id->Items->Add(DBKategorii->FieldByName("id\_kategorii")->AsString);

if (DBStravaid\_kategorii->Value == DBKategoriiid\_kategorii->Value) index = i;

DBKategorii->Next();

}

FAdmin->K\_Strava\_Kategoriya->ItemIndex = index;

FAdmin->K\_Strava\_Kategoriya\_Id->ItemIndex = index;

} catch (...) { }

}

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

void \_\_fastcall TDM::N14Click(TObject \*Sender)

{

FAddIngredienty->Show();

FAddIngredienty->K\_Ingredient\_kilkisti->Text = 0;

FAddIngredienty->K\_Ingredient\_name->Clear();

FAddIngredienty->K\_Ingredient\_vumir->Clear();

}

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

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

{

Ingredienty.SetIdIngredienty(DM->DBIngredientykod\_ingredienta->Value);

Ingredienty.RemoveIngredienty();

}

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

void \_\_fastcall TDM::DBZakazAfterScroll(TDataSet \*DataSet)

{

try {

FAdmin->K\_Zakaz\_Date->Text = DM->DBZakazdates->Value;

FAdmin->K\_Zakaz\_nomer\_stola->Text = DM->DBZakaznomer\_stola->Value;

FAdmin->K\_Zakaz\_price\_zakaz->Text = DM->DBZakaz->FieldByName("pricezakaz")->AsString + " грн";

} catch (...) {

}

}

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

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

{

FMain->Show();

}

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

A.6 Hiderfile.h

#include <vcl.h>

#include <tchar.h>

#include <jpeg.hpp>

class TPersonal{

private:

String login,password;

bool admin;

protected:

int id\_personal;

String pib;

public:

TPersonal();

TPersonal(int,String,String,String);

void SetIdPersonal(int);

void SetInfoPersonal(String,String,String,bool);

void StatusAdmin();

void StatusUser();

bool Initsialization(String,String);

bool GetAdmin();

int GetIdPersonal();

void DeleteInfoPersonal();

void InsertPersonal();

void UpdatePersonal();

void RemovePersonal();

}Personal;

class TKategoriya{

protected:

int id\_kategoriya;

String name\_kategoriya;

public:

TKategoriya();

TKategoriya(int,String);

void SetIdKategoriya(int);

void SetInfoKategoriya(String);

void InsertKategoriya();

void UpdateKategoriya();

void DeleteKategoriya();

}Kategoriya;

class TIngredienty{

protected:

int id\_ingredient;

String names,vumir;

float count;

public:

TIngredienty();

TIngredienty(int,String,String,float);

void SetIdIngredienty(int);

void SetInfoIngredienty(String,String,float);

void InsertIngredienty();

void RemoveIngredienty();

}Ingredienty;

class TNapiy: public TKategoriya{

protected:

int id\_napiy;

String names\_napiy,data\_napiy;

float cost\_price\_napiy,price\_napiy,natsinka\_napiy;

TBitmap \*foto\_napiy;

public:

TNapiy();

TNapiy(int,String,String,float,float,float);

void SetIdNapiy(int);

void SetInfoNapiy(String,String,float,float,float);

void SetFotoNapiy(TBitmap\*);

void InsertNapiy();

void UpdateNapiy();

void RemoveNapiy();

}Napiy;

class TStrava: public TKategoriya{

protected:

int id\_strava;

String names\_strava,data\_strava;

float cost\_price\_strava,price\_strava,natsinka\_strava;

TIngredienty mas\_ingredient[50];

TBitmap \*foto\_strava;

public:

TStrava();

TStrava(int,String,String,float,float,float);

void SetIdStrava(int);

void SetInfoStrava(String,String,float,float,float);

void SetFotoStrava(TBitmap\*);

void InsertStrava();

void UpdateStrava();

void RemoveStrava();

}Strava;

class TZakaz:public TPersonal,public TStrava{

protected:

int id\_zakaz,nomer\_stola;

float count;

TDateTime dates;

public:

TZakaz();

void SetIdZakaz(int);

void SetCount(float);

void SetInfoZakaz(int,TDateTime);

void InsertZakaz();

void InsertPunktZakaz();

void UpdateZakaz();

void RemoveZakaz();

void RemovePunktZakaz();

void RemovePunktuZakaz();

void CalculationPrice();

}Zakaz;