МІНІСТЕРСТВО ОСВІТІ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Методичні вказівки до виконання лабораторної роботи за темою «Розробка засобами Microsoft Visual Studio прикладного програмного забезпечення для роботи з базою даних Microsoft SQL Server»

для студентів, що навчаються за спеціальностями
«121 Інженерія програмного забезпечення»
«122 Комп’ютерні науки»
«126 Інформаційні системи та технології»

Затверджено
редакційно-видавничою
радою університету,
протокол № 2 від 16.06.2023 р.

Харків
НТУ ХПІ
2023

Методичні вказівки до виконання лабораторної роботи за темою «Розробка засобами Microsoft Visual Studio прикладного програмного забезпечення для роботи з базою даних Microsoft SQL Server» для студентів, що навчаються за спеціальностями 121 «Інженерія програмного забезпечення», 122 «Комп'ютерні науки», 126 «Інформаційні системи та технології» / Уклад. Орловський Д.Л., Копп А.М. – Харків: НТУ «ХПІ», 2023. – 63 с.

Укладачі Д.Л. Орловський,
А.М. Копп

Рецензент Гринченко М.А.

Кафедра програмної інженерії та інтелектуальних технологій управління

ЗМІСТ

ВСТУП

Сучасні інформаційні системи ґрунтуються на використанні баз даних, в яких накопичується різна інформація. Тому зараз розробляються і значно поширюються методи і засоби роботи з базами даних з метою підвищення ефективності роботи людини у різних галузях діяльності. Ці засоби та методи пов’язані з узагальненням і різними додатковими способами обробки даних. Основні ідеї сучасної інформаційної технології базуються на концепції, згідно з якою дані повинні бути організовані в бази даних з метою адекватного відображення реального світу, що змінюється, і задоволення інформаційних потреб користувачів. Ці бази даних створюються і функціонують під управлінням спеціальних програмних комплексів, які називають системами управління базами даних (СУБД).

Методичні вказівки до лабораторної роботи за темою «Розробка засобами Microsoft Visual Studio прикладного програмного забезпечення для роботи з базою даних Microsoft SQL Server» призначені для студентів, що навчаються за спеціальностями «121 Інженерія програмного забезпечення», «122 Комп’ютерні науки», «126 Інформаційні системи та технології».

Виконання лабораторної роботи повинно забезпечити закріплення теоретичних знань і практичних навичок, отриманих при вивченні лекційної частини дисциплін, пов’язаних із проектуванням, розробкою та застосуванням баз даних.

В методичних вказівках розглянуті основні питання, пов’язані з теоретичним обґрунтуванням та безпосереднім виконанням лабораторної роботи.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

Користувач автоматизованої обчислювальної системи має право очікувати не тільки точних результатів обробки, але і зручності в використанні системи. Тіло людини (користувача) - це механізм, який працює в рамках визначених обмежень і допусків. Очам людини необхідно, щоб образи мали визначений розмір, рівень яскравості, контрастності і розміщувались на зручній відстані. Деякі кольори сприймаються краще інших. Необхідно відмітити і обмеження мозку людини. У людини велика довготривала пам'ять і дуже обмежена короткочасна пам'ять. Подібно буферам обчислювальної системи ця пам'ять може легко перевантажуватись. Люди можуть розширювати цю пам'ять за допомогою записів на листках паперу, застосовуючи нові моделі роботи або пристосовуючись до нового способу роботи. Однак така адаптація може приводити до стресів, а як результат до неприйнятного рівня помилок.

Критерій зручності став визначальним в останні роки в зв'язку з широким застосування персональних комп'ютерів з діалоговими режимами роботи в різних галузях народного господарства, побуті, що привело до зростання кількості користувачів неспеціалістів в області інформатики та обчислювальної техніки. Згідно з класифікацією фірми IBM розрізняють такі типи користувачів:

- системний програміст;
- програміст;
- кінцевий користувач.

Кінцеві користувачі - це люди, які хочуть більш ефективно і економно вирішувати свої завдання, використовуючи комп'ютерні засоби. Частіше всього вони не є спеціалістами в області інформатики. Досвід показує, що ці користувачі можуть ефективно вирішувати на машині свої завдання лише в умовах наявності активної допомоги зі

сторони системи на всіх етапах вирішення завдання і спрощення методів взаємодії з обчислювальною системою.

Зрозуміло, що сучасні обчислювальні системи повинні орієнтуватись на таких користувачів в першу чергу, оскільки більшість користувачів персональних комп'ютерів є кінцевими користувачами. Використовуючи в подальшому термін користувач, ми маємо на увазі кінцевих користувачів.

Таким чином, з точки зору користувача діалогової системи - система є зручною для користувача, якщо інтелектуальні зусилля користувача, необхідні для розуміння дій системи і реакції на них мінімальні. Можливо сформулювати ряд вимог до системи з точки зору зручності:

1. Поведінка системи по відношенню до користувача повинна бути гнучкою, тобто користувач не повинен діяти строго визначеним способом.

2. Система повинна вміти розрізняти користувача і пристосовуватись до нього.

3. Поведінка системи повинна бути зрозумілою користувачеві.

4. Система завжди повинна бути готова допомогти користувачеві.

5. Для використання системи не потрібні спеціальні навики та додаткове навчання.

6. Не потрібно зловживати здатністю людини до навчання під час роботи з системою.

7. Система повинна реагувати на порушення взаємодії з користувачем, обумовлені властивостями людини і приймати запобіжні заходи проти цих порушень.

Інтерфейс користувач-комп'ютер включає всі ті аспекти обчислювальної системи, з якими безпосередньо взаємодіє користувач.

Ефективна робота кінцевих користувачів з базами даних можлива лише за наявності інтерфейсу користувача як обов'язкової складової застосунку, який забезпечує роботу з базою даних.

1.1 Визначення та суть інтерфейсу користувача

Інтерфейс користувача (ІК) - це сукупність засобів, за допомогою яких користувач взаємодіє з різними пристроями (з комп'ютером або побутовою технікою) або іншим складним інструментарієм (системою). Інтерфейс користувача - це такий різновид інтерфейсів, в якому з одного боку - людина, з іншого - машина (пристрій, програмне забезпечення).

ІК часто розуміють лише як зовнішній вигляд програмного забезпечення (ПЗ), але таке розуміння є надто вузьким, оскільки саме за допомогою інтерфейсу користувач сприймає програму в цілому та використовує її функціональність. ІК забезпечує підтримку прийняття рішень у визначеній предметній галузі та визначає порядок використання ПЗ і документації до нього. В дійсності, ІК об'єднує усі елементи і компоненти ПЗ, які здатні впливати на взаємодію користувача з програмним забезпеченням. До таких елементів належать: набір задач, які користувач розв'язує за допомогою ПЗ; використовувана програмним забезпеченням метафора (наприклад, "робочий стіл" у операційній системі Windows); елементи управління ПЗ; навігація між блоками ПЗ; візуальний (і не тільки) дизайн вікон та екранних форм програми та інші складові (рисунок 1.1).

Стиль інтерфейсу користувача - це набір ознак, методів, прийомів діяльності, які характеризують індивідуальність інтерфейсу користувача, а також сукупність прийомів використання інструментів розроблення ПЗ.

Процес проектування ІК – це складний, нелінійний, недетермінований і неортогональний процес. Складність ІК обумовлюється рядом невизначеностей, які суттєво впливають на процес розроблення. Нелінійність проектування ІК полягає у відсутності фіксованого, впорядкованого і прямолінійного алгоритму від початку до кінця проектування. Процес проектування є невизначеним, оскільки не існує рівняння, за яким можна було б одержати однаковий результат при заданих однакових початкових

умовах, більш того, одержати ідентичний результат практично неможливо. Інтерфейс користувача неортогональний у тому сенсі, що будь-який аспект проектного рішення може впливати на інші аспекти, до того ж результат цього впливу не завжди є позитивним та прийнятним.

Рисунок 1.1 - Складові інтерфейсу користувача

1.2 Графічний інтерфейс користувача

Графічний інтерфейс користувача (GUI - Graphical User Interface) - тип інтерфейсу користувача, в якому елементи інтерфейсу (меню, кнопки, значки, списки), представлені на екрані, виконані у вигляді графічних зображень. В графічному інтерфейсі користувача (ГІК) забезпечено довільний доступ (за допомогою пристроїв введення) до

всіх видимих екранних об'єктів (елементів інтерфейсу) та безпосереднє маніпулювання ними. Найчастіше елементи інтерфейсу в ГІК реалізовані на основі метафор та відображають свої призначення та властивості, що полегшує розуміння та засвоєння ПЗ недосвідченими користувачами.

Основною характеристикою графічного інтерфейсу користувача є використання ряду робочих елементів. ГІК є графічним представленням на екрані комп'ютера способу чи функції для інтерактивної взаємодії з програмами, об'єктами і даними. Він забезпечує користувачів необхідним інструментарієм і застосунками, а не просто списком функцій.

Характеристики ГІК:

1) підтримує ідею сумісності між програмами;

2) користувачі можуть бачити графічні зображення і текст на екрані в тому вигляді, в якому вони будуть роздруковані;

3) слідує концепції інтерактивної взаємодії "об'єкт-дія";

4) дозволяє переміщувати інформацію між програмами;

5) надає можливість безпосереднього маніпулювання об'єктами та інформацією на екрані;

6) пропонує стандартні елементи інтерфейсу (меню та діалогові вікна);

7) забезпечує візуальне відображення інформації і об'єктів (іконки і вікна);

8) забезпечує візуальний зворотній зв'язок в процесі виконання користувачами дій та задач;

9) надає візуальне відображення дій користувача/системи, а також режимів (меню, палітри);

10) використовує графічні керуючі елементи, які дозволяють користувачам робити вибір та вводити дані;

11) надає користувачам можливість налагодити та персоналізувати інтерфейс та інтерактивні дії;

12) забезпечує гнучкість переходу від клавіатури до іншого пристрою введення даних.

Під час досліджень зручності застосування ГІК було окреслено коло задач, які найбільш часто розв'язуються користувачами різних рівнів підготовки: запуск програми, виклик підказки, відкриття файлу, збереження файлу, копіювання файлу, зміна кольору робочого столу, пошук файлу, запуск ще одного додатку, видалення файлу/папки, перейменування файлу/папки, вибір принтера, створення папки, перегляд черги завдань принтера, розміщення документу на сервері, закриття програми, відхилення видалення/відновлення файлу, перевірка стану ресурсів системи.

Одним з недоліків ГІК є його орієнтація на застосунки. При роботі з ГІК користувачі бачать інформацію на екрані, в них складається враження, що вони дійсно працюють з об'єктами, однак увага користувачів сконцентрована на додатках. Вони обирають застосунок, вказують файл даних, який використовується, організовують додатки і дані на комп'ютерах у вигляді графічних деревовидних структур, використовуючи диски і папки.

ГІК передбачає проблемно-орієнтований підхід - перш ніж виконувати операції з файлами, користувачі повинні запустити програмне забезпечення, яке працює з даним типом файлів. Користувачі працюють з застосунками у вікнах, які мають панель меню з варіантами маршрутів, відображуваними спадними меню. Такі меню містять варіанти дій і маршрутів, пов'язаних з об'єктами у вікні або додатком, який надає саме вікно.

Будь-який вдало розроблений ІК повинен знижувати навантаження на пам'ять користувача. ГІК здатні запропонувати наочні підказки та необхідні відомості, використовуючи комп'ютерні потужності для зберігання та пошуку інформації. ГІК об'єднують три основних стилі інтерфейсу: командний рядок, меню та пряме маніпулювання.

Методи інтерактивної взаємодії ГІК передбачають використання клавіатури або вказівних пристроїв для переміщення, вибору та маніпулювання інформацією на екрані.

До методів інтерактивної взаємодії ГІК належать:

1) можливість використання клавіатури або миші в залежності від переваг користувача;

2) робота з різними типами меню;

3) два режими редагування: вставка або заміна;

4) технологія drag-and-drop;

5) буфер обміну.

1.3 Принципи побудови «дружнього» інтерфейсу користувача

Поняття "дружнього" інтерфейсу користувача (User-friendly Interface) розшифровується як "інтерактивний програмний інтерфейс, який забезпечує природній для користувача режим взаємодії з обчислювальною машиною". Відтоді, як ІК став одним з найважливіших елементів ПЗ, його якість та "дружність" хвилює кожного розробника і проектувальника.

Думка користувача про те, що деякий ІК якісний, тобто кращий за інші, залежить від невеликої кількості характеристик, які стають очевидними після знайомства з будь-яким додатком незалежно від стилю ІК. Якісний ІК повинен базуватись на 5 головних принципах, які позначають абревіатурою SAPCO (рисунок 1.2).

Простий. Програмні об'єкти забезпечують підвищення продуктивності і нарощення можливостей ПЗ без внесення додаткової складності в ІК. На початкових етапах розробки стилю ІК широко використовується мінімалізм і розбиття ПЗ на множину слабозв'язаних рівнів. Якісний прикладний ІК не потребує довідника або діалогової довідкової системи, щоб почати виконувати задачі за його допомогою. Якісний прикладний ІК зрозумілий на інтуїтивному рівні (користувачу потрібне лише пояснення, як досягти результату).

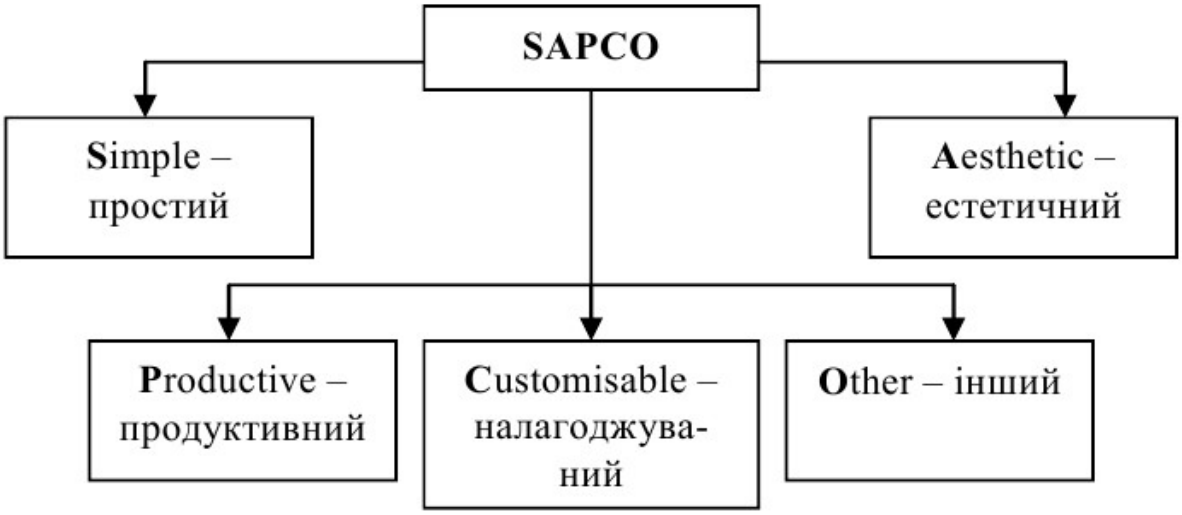


Рисунок 1.2 - Принципи SAPCO проектування якісного ІК

Естетичний. Програмні об’єкти повинні мати естетичну та ергономічну привабливість, при цьому широко використовується графічний дизайн та візуалізація. Якісний прикладний ІК повинен не викликати негативних емоцій у користувача, використовувати метафори реального світу користувача та максимально візуалізувати інформацію.

Продуктивний. Використання програмних продуктів вимагає мінімальних зусиль для вирішення задач користувача. ІК реалізується таким чином, щоб уникнути складної ієрархічності вікон та/або екранів, а також зайвих дій з клавіатурою та мишею.

Налагоджуваний. Програмне забезпечення доступне в різних формах для задоволення індивідуальних потреб. Програмні об’єкти ІК повинні мати можливість налагодження (налагоджуваність). Якісний прикладний ІК дозволяє користувачу обрати метод взаємодії і методи макетування та доступу для оптимізації потреб користувачів.

1.4 Правила проектування інтерфейсу користувача

Розглянемо основні правила проектування інтерфейсу користувача.

Правило 1: "дати контроль користувачу". Основне правило проектування ІК – дати користувачу контроль над системою. Проста аналогія – водій автомобіля та пасажир потягу. Рішення – їхати власним авто або потягом – повинен приймати саме пасажир (в нашому випадку користувач). Розробник інтерфейсу повинен виділити в якості основного такий принцип проектування, при якому продукт буде задовільняти всім вимогам. Досвідчені проектувальники дозволяють користувачам вирішувати деякі задачі на власний розсуд. Так, наприклад, європейські архітектори по завершенні будівництва складного комплексу будівель повинні прокласти між ними доріжки для пішоходів. На майданчиках між будинками ставлять таблички: “Ходіть, будь-ласка, по траві”. Через деякий час будівельники повертаються і лише тоді, згідно волевиявленню населення, асфальтують протоптані доріжки. Це прекрасний приклад надання можливості користувачу контролю над ситуацією і проектування інтерфейсу, який би відповідав потребам та побажанням користувача.

Основними положеннями надання контролю користувачу над інтерфейсом є безрежимність, гнучкість, можливість переривання, корисність, поблажливість, можливість орієнтування, доступність, спрощення користування, пристосовуваність, інтерактивність.

Безрежимність - вибір та використання режимів розважливо. Режими (певні умови роботи, діяльності) - атрибут багатьох програмних інтерфейсів, але застосовувати їх потрібно лише при необхідності. Існує ряд інтерфейсів, які занадто часто перемикаються між режимами. На екрані з’являється діалогове вікно режиму, і користувач стає обмеженим в діях не лише в даній програмі, але й не має можливості переходу в інші програми.

Інтерфейс може працювати в двох режимах, які в багатьох випадках необхідні, однак позбавляють користувача самостійності:

1) режим користувача – обмежує режим роботи і дозволяє користувачу виконувати лише певні дії в певному місці програми. В наш

час проектувальники інтерфейсів все частіше обходяться без перемикання режимів. Можливо, оптимальним варіантом є показ даних в форматі, який відповідає рівню доступу користувача;

2) системний режим – використовується достатньо рідко. Якщо система знаходиться в цьому режимі, йому дозволено працювати лише з поточною програмою. При розробленні повідомлень та інформації про допомогу в такому режимі слід пам'ятати про те, наскільки незручний системний режим для користувача.

При виборі режимів важливо слідувати принципу негайного візуального оберненого зв'язку. Користувач повинен бути постійно впевнений в тому, що він знаходиться в потрібному режимі. Режими інтерфейсу повинні бути настільки природніми, щоб користувачу було комфортно працювати з ними.

Гнучкість - забезпечує користувачу можливість вибору, наприклад, можливість роботи з клавіатурою передбачає використання клавіатури замість миші. Панелі інструментів створені для прискорення роботи при використанні миші, однак при роботі з клавіатурою до них важко дістатись – для подібних випадків передбачені підменю. Більшість користувачів мають звичку працювати як з клавіатурою, так і з мишею.

Можливість переривання - дозвіл користувачу перемикати увагу між різними програмними засобами. Програмні інтерфейси повинні бути спроектовані так, щоб користувач міг в будь-яку хвилину перервати або зберегти результати роботи. Забезпечення контролю користувача над програмою та його підтримка – ось головні принципи розроблення. Не потрібно змушувати користувачів закінчувати виконання початих послідовностей дій. Вони повинні мати вибір – анулювати або зберегти дані і повернутись туди, де вони перервались.

Корисність - демонстрація повідомлень, які допомагатимуть користувачу в роботі. У інтерфейсі потрібно використовувати зрозумілі для користувача терміни. Даний принцип застосовний не лише для

повідомлень, але й для усіх текстів на екрані: запрошень, інструкцій, заголовків, написів на кнопках. Усі текстові аспекти інтерфейсу повинні розроблятися спеціалістами, котрі мають знання з орфографії та лексики. При написанні системної та програмної документації, а також повідомлень слід обрати вірний тон. Невдалі термінологія та тон призводять до того, що користувачі звинувачуватимуть себе у помилках, що виникають при використанні ПЗ.

Поблажливість - створення умов для негайних та зворотніх дій, а також зворотнього зв'язку. Недостатність зворотнього зв'язку в більшості програмних продуктів змушує користувача витрачати багато сил на виконання поставленої задачі. Кожен програмний продукт повинен мати функції відхилення (скасування) та повторення дії. Необхідно інформувати користувача, якщо дана дія не може бути відхилена, і, по можливості, дозволити йому альтернативну дію.

Можливість орієнтування - забезпечення доступу користувачу до будь-якої частини інтерфейсу. Користувач повинен мати можливість вільно орієнтуватись в інтерфейсі, мати доступ до будь-якої його частини, можливість пересуватись вперед і назад по нисхідній та висхідній структурі інтерфейсу, отримувати зручні контекстні підказки там, де вони потрібні. Наприклад, панель задач Windows показує, які програми виконуються, і надає користувачу доступ до них.

Панелі інструментів, меню, палітри та інші елементи інтерфейсу операційних систем, набори програмних продуктів та різні утиліти - все це розроблено, щоб допомогти користувачам орієнтуватись в операційній системі та в просторі жорсткого диску. Системні та програмні "помічники" та "асистенти" теж пропонують допомогу для орієнтації в програмних функціях або задачах.

Доступність - налагодження системи на користувачів з різним рівнем підготовки. Деякі програми пропонують спеціальні інтерфейси, які допомагають обрати рівень складності взаємодії. Наприклад, панель задач та підменю програм можуть бути стандартними або

деталізованими в залежності від вибору користувача та типу виконуваної задачі. Для досвідчених користувачів потрібно передбачити швидкий доступ до функцій ПЗ.

Спрощення користування - створення зрозумілого, "дружнього" ІК. ІК – "міфічна" частина програмного продукту. При вдалому проектуванні користувачі не помічають інтерфейсу. Якщо інтерфейс розроблений невдало, користувачам доведеться докласти чимало зусиль для ефективного використання програмного продукту. "Прозорість" інтерфейсу забезпечується тим, що користувачу надається можливість користуватись об’єктами, відмінними від системних команд.

Пристосовуваність - надання можливості користувачу налагоджувати інтерфейс за своїм смаком. Користувач повинен мати можливість налагодження: способу представлення інформації на свій вибір (кольори, шрифти, розташування елементів), поведінки інтерфейсу (дії за замовчуванням, макроси, кнопки) та інтерфейсних функцій (натискання кнопок чи клавіш, сполучення клавіш для швидкого вибору команд, мнемоніка, розташування кнопок миші для виконання команд).

Інтерактивність - дозвіл користувачу маніпулювати об'єктами інтерфейсу. Всюди, де це можливо, потрібно дозволяти користувачу безпосередньо взаємодіяти з об’єктами на екрані в природньому, натуральному стилі. Користувачі повинні комфортно почуватись при виконанні операцій та знати про передбачуваний результат.

Правило 2: "зменшити навантаження на пам’ять користувача".

Положеннями зменшення навантаження на пам’ять користувача є запам'ятовування, розпізнавання, інформування, терпимість, швидкість, інтуїтивність, перенос, контекст, організація.

Запам’ятовування - не слід завантажувати короткочасну пам'ять. Короткочасна пам’ять допомагає нам зберігати інформацію на протязі невеликого проміжку часу. Користувачі часто працюють над декількома

задачами одночасно, тому не варто ІК завантажувати короткочасну пам’ять користувача в моменти перемикань між ними. Цей принцип проектування часто порушується, що змушує застосовувати зовнішні “засоби” для зберігання інформації (папір, калькулятор).

Функції програм (виділення останньої дії та її повтор) та дії з використанням буферу обміну дозволяють користувачам маніпулювати частинами інформації, необхідними в багатьох місцях, а також всередині задач. Оптимальним є варіант, коли програма в потрібному місці може автоматично зберегти та передати дані, коли користувач зайнятий виконанням інших задач.

Розпізнавання - потрібно опиратись на розпізнавання, а не на повторення. Підтримка інтерфейсом довгочасної пам’яті передбачає розпізнавання інформації користувачем, перш ніж він згадає її. Легше обрати якийсь об’єкт зі списку, ніж згадувати його правильну назву для введення в порожній рядок. Онлайнова допомога, повідомлення, поради щодо користування інструментами, система контекстної допомоги тощо, допомагають користувачу обирати інформацію, а не згадувати її. Слід передбачити списки і меню, що містять об’єкти чи документи, які можна обирати, не змушуючи користувачів вводити інформацію в командному рядку без підтримки системи.

Інформування - наявність візуальних заставок. Необхідний аспект будь-якого графічного ІК та об'єктно-орієнтованого ІК полягає в тому, що користувачі повинні знати, в якому місці ПЗ вони знаходяться, які дії виконують наразі і які дії можуть виконати в подальшому. Візуальна інформація служить нагадуванням для користувачів. Коли користувачі працюють в якомусь режимі або з мишею, це повинно відображатись на екрані за допомогою відповідної індикації. Форма курсора може змінюватись для вказання поточного режиму або дії, а індикатор – вмикатись і вимикатись. Візуальна інформативність продукту полягає в наявності візуальних підказок інтерфейсу, які інформують користувача про його поточні дії.

Терпимість - треба передбачити параметри за замовчуванням, команди відхилення (скасування) та повторення дії. Інтерфейс повинен надавати користувачу можливість зміни установок та налаштувань ПЗ, а також можливість повернення до установок за замовчуванням. Користувачі за своїми перевагами можуть змінити колір, шрифт, властивості прикладної програми або операційної системи, не задумуючись про їх початковий варіант.

При розробленні інтерфейсу потрібно передбачити багаторівневі системи відхилення та повторення команд. Швидкість - слід передбачити "швидкі" шляхи проходження інтерфейсу. Крім комбінованого використання миші та клавіатури існує ряд інших можливостей прискорити роботу з програмою. “Гарячі” клавіші та ярлики зменшують навантаження на пам’ять користувача та доводять виконання операцій до автоматизму.

Є 2 основних способи встановлення ярликів: прискорюючі та мнемонічні. Мнемонічні (або доступні) ярлики - це одиночні буквенно-цифрові символи, які встановлюють курсор на потрібний об’єкт та дозволяють зробити вибір. Вони використовують різні меню (панель, підменю, контекстні) та списки. Мнемонічні символи повинні бути унікальними для кожного роду дій. Наприклад, типове меню вікна використовує стандартні клавіші: F - для файлу, E - для редагування, V - для перегляду, H - для виклику довідкової системи. Наступний рівень меню (підменю) використовують свої налагодження мнемонічних клавіш: N - новий документ, O - відкрити, C - закрити, S - зберегти. Мнемонічні символи прискорюють рух і вибір потрібного меню або списку. Прискорюючі ярлики (або клавіші швидкого доступу) – це клавіші або комбінації клавіш, які користувачі повинні натиснути для здійснення якої-небудь дії. Наприклад, Ctrl+P – прискорюючий ярлик для друку.

Щойно користувачі добре засвоять програмний продукт, вони починають відчувати потребу в прискорюючих ярликах. Не слід ігнорувати цю необхідність, однак при проектуванні потрібно слідувати

стандартам щодо призначення клавіш та їх комбінацій, інакше користувачі робитимуть помилки, використовуючи звичну комбінацію клавіш, яка виконуватиме непередбачувану дію.

Інтуїтивність - потрібно активізувати синтаксис дій з об'єктами. Об'єктно-орієнтований інтерфейс надає ряд переваг від використання об'єктно-орієнтованого синтаксису взаємодії. Користувачам не треба запам’ятовувати, яку дію можна виконати в певний момент часу для об’єкта. При виборі об'єкта користувач бачить у меню дії, які можуть бути виконані над об’єктом. Недопустимі дії виділяються, наприклад, сірим кольором. Об'єктно-орієнтований синтаксис дозволяє користувачу зрозуміти взаємозв’язок між об’єктами та діями в програмному продукті.

Перенос - в інтерфейсі слід використовувати метафори (переноси, які використовують назву об'єкту одного класу для опису об'єкту іншого класу) реального світу, які дозволяють користувачу переносити свої знання з реального світу у світ комп’ютерів. Наприклад, для програми Калькулятор не варто розробляти новий інтерфейс, він повинен співпадати з інтерфейсом звичайного калькулятора, який добре засвоїли користувачі. Однак при виборі та використанні метафор для інтерфейсу слід бути обережними. Обираючи метафору, потрібно зафіксувати її та слідувати їй весь час. Якщо виявиться, що метафора не відповідає своєму призначенню в усьому інтерфейсі, з'явиться необхідність вибору нової метафори.

Контекст - слід застосовувати розкриття та пояснення понять і дій. Користувачі не повинні вважати можливості ПЗ більшими чи іншими, ніж вони є насправді. На кожному рівні інтерфейсу не варто показувати абсолютно всі функції ПЗ, а лише ті, в яких є потреба саме на даному рівні. Деякі програмні продукти надають різні меню для користувачів. Спочатку можна обрати просте меню для щоденних, звичайних потреб. Пізніше, по мірі засвоєння продукту або при виникненні потреби в більш складних властивостях програми, користувачі

зможуть обрати більш складне меню. Це приклад того, як користувач одержує контроль над задачею та програмою. Нові властивості інтерфейсів (Майстри та Порадники) розкривають та пояснюють поняття та дії при вирішенні задач. Майстри допомагають користувачу виконувати задачу крок за кроком, кожен з яких є зрозумілим. Розробник має забезпечити користувачу легкий доступ до часто використовуваних функцій та дій. Можна приховати частину властивостей та функцій і дозволити користувачу викликати їх по мірі необхідності. Не потрібно намагатись відобразити всю інформацію в головному вікні, для цього можна використати вторинні вікна.

Організація - потрібно збільшити візуальну ясність інтерфейсу. Принципи візуального проектування дають можливість полегшити сприйняття інформації шляхом використання групувань об'єктів в меню або списки; нумерації об'єктів; заголовків та запрошень. Деякі програми одночасно надають надто багато інформації на екрані, що викликає відчуття хаосу. Інформація має подаватись у порядку, зрозумілому користувачам, тобто "форма повинна відповідати призначенню".

Правило 3: "зробити інтерфейс сумісним".

Сумісність – ключовий аспект інтерфейсу. Однією з основних переваг сумісності є те, що користувачі можуть перенести свої знання та навички зі старої програми, якою вони користувались раніше, у нову.

Наприклад, при створенні інформаційного повідомлення бажано присвоїти йому ідентифікаційний номер, який надалі слід видавати разом з повідомленням. Це дозволить користувачу зрозуміти, що схожі повідомлення з'являються кожен раз в певній ситуації, незалежно від того, в якому місці інтерфейсу він знаходиться.

Положеннями розроблення сумісного інтерфейсу є: послідовність інтерфейсу, досвід, прогнозування, відношення, передбачуваність.

Послідовність інтерфейсу - проектування послідовного, узгодженого інтерфейсу. Користувачі повинні мати опорні точки інтерфейсу. Це заголовки вікон, навігаційні карти та деревовидні структури. Користувачу також потрібна можливість завершити поставлену задачу без зміни середовища роботи або перемикання між стилями введення інформації. Візуальна допомога підкаже користувачу, що відбудеться при завершенні задачі.

Досвід - спільна сумісність всіх програм. Один з головних аспектів в розробці інтерфейсу – можливість навчання користувача головним концепціям системи та програмного забезпечення, що можуть застосовуватись у нових ситуаціях та інших програмах.

Сумісність - одна з головних властивостей ІК, яка проявляється на трьох рівнях: подання інформації, поведінка програми та техніка взаємодії.

Сумісність в поданні інформації передбачає, що користувачі можуть сприймати інформацію та об’єкти в схожому логічному, візуальному та фізичному вигляді в усьому програмному продукті. Стиль представлення інформації не повинен змінюватись без видимих причин.

Сумісність в поведінці програми передбачає, що об’єкт є однаковим всюди. Поведінка контрольних інструментів, таких як кнопки, списки і меню, не змінюється всередині програми або у різних програмах одного розробника. Користувачі мають бути впевнені в послідовній поведінці об’єктів інтерфейсу.

Сумісність техніки взаємодії також дуже важлива. Однакові сполучення “швидких” клавіш та способи роботи з мишею повинні працювати в схожих за призначенням програмах. Мнемонічні схеми клавіатури теж не повинні змінюватися. Користувачі очікують аналогічних результатів при взаємодії однаковими методами з різними об’єктами. Коли схожі об’єкти поводяться по-різному у схожих ситуаціях, у користувачів відбувається негативний перенос знань. Це

гальмує вивчення інтерфейсу програми і призводить до втрати користувачем впевненості у своїх силах.

Прогнозування - збереження результатів взаємодії. Якщо користувач, виконуючи одні й ті ж дії, одержує різні результати, він більш схильний звинувачувати у цьому себе, ніж програмне забезпечення. Сумісність кроків і дій має витримуватись в усьому програмному продукті. Процеси реєстрації та навігації мають бути послідовними. Стандартні елементи інтерфейсу повинні поводитись однаково. Якщо результати можуть відрізнятись від того, що очікує користувач, слід інформувати його перед виконанням дії, а також надати йому опції виконання, скасування або здійснення іншої дії.

Відношення - естетична привабливість та цілісність. Деякі з сучасних програмних продуктів виглядають так, ніби вони розроблені різними людьми, які жодного разу не спілкувались між собою. Сумніви користувачів щодо цілісності ПЗ виникають, якщо в різних його частинах виявляється непослідовність кольорів, шрифтів, іконок та інших складових інтерфейсу. Приємний для сприйняття інтерфейс не приховує недолік функціональності програмного забезпечення.

Передбачуваність - заохочення вивчення. Задача проектувальників ІК - створити "дружній" інтерфейс, який заохочуватиме користувача досліджувати його складові і властивості без страху зробити щось невірно.

2 ВИКОНАННЯ РОБОТИ

Для початку виконання роботи треба підключити базу даних, яку було створено при виконанні лабораторної роботи за темою «Ознайомлення з основними особливостями СУБД Microsoft SQL Server. Створення бази даних та об'єктів бази даних».

При виконанні лабораторної роботи передбачається використання інтегрованого середовища розробки Microsoft Visual Studio. У зв'язку із цим елементи інтерфейсу можуть мати відмінності порівняно із тими, що наведено далі у вигляді рисунків (особливо у випадку використання версії, локалізованої для використання певної національної мови). Ці відмінності не є принциповими та не впливають на виконання роботи та отримані результати.

2.1 Створення проекту та головної форми

Для створення проекту та головної форми необхідно виконати таку послідовність дій.

1. Запустити Microsoft Visual Studio.

2. Створити новий проект (для цього, наприклад, вибрати в головному меню пункт File, потім вибрати New, потім Project). Під час створення нового проекту вказати тип проекту (рисунок 2.1) та місце розміщення файлів проекту. В результаті на екрані з'явиться інтерфейс Visual Studio (рисунок 2.2). Як видно, застосунок складається поки що з однієї екранної форми. Цю форму будемо використовувати як головну форму програми.

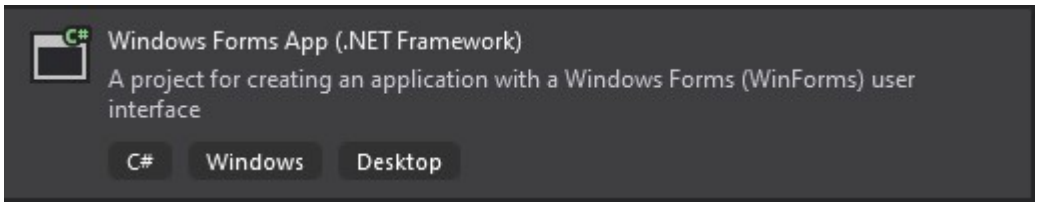


Рисунок 2.1

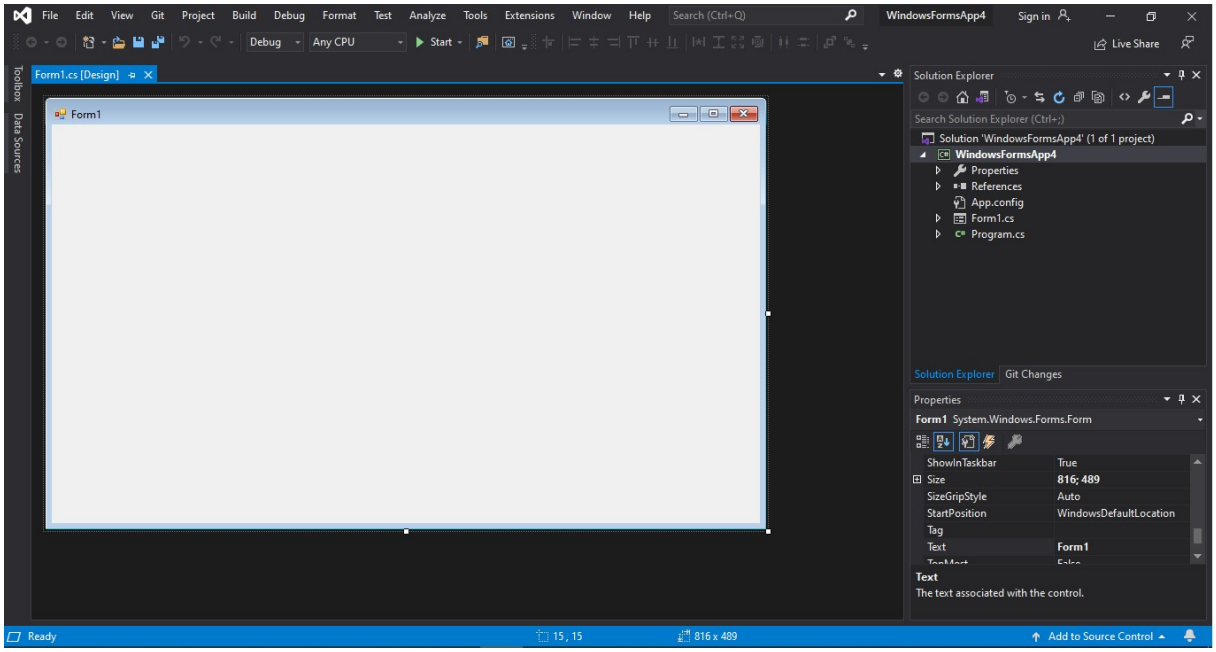


Рисунок 2.2

3. Виконати пробний запуск програми, для чого:

• у головному меню вибрати пункт Debug;

• у вертикальному меню вибрати пункт Start Without Debugging;

• в результаті на екрані з'явиться форма, яка не містить будь-яких елементів керування. Після запуску форму слід закрити.

4. Знайти властивість форми IsMdiContainer та встановити для неї значення True. Для властивості Text встановити значення Supply.

5. Встановити розміри форми. Принципового значення цьому етапі роботи розміри форми немає. Для прикладу можна для наступних властивостей форми встановити значення:

Size: 433; 316; Location: 5; 5

6. Додати у форму головне меню. Для цього в панелі інструментів Toolbox вибрати компонент MenuStrip та перетягнути його на форму. Введіть назви трьох пунктів головного меню: Data, Reports, Exit. Для пункту Data сформувати підменю, що складається з пунктів: Suppliers, Supplied.

7. Запустити форму. Зовнішній вигляд форми може бути приблизно таким, як показано на рисунку 2.3 (геометричні розміри реальної форми будуть більшими).

Рисунок 2.3

8. Для пункту головного меню Exit створити функцію – обробник події, що дозволяє закрити програму. Для введення тексту функції подвійним клацанням миші відкрити вікно програмного коду та ввести текст функції (рисунок 2.4). Після цього запустити програму та перевірити коректність роботи пункту меню Exit.

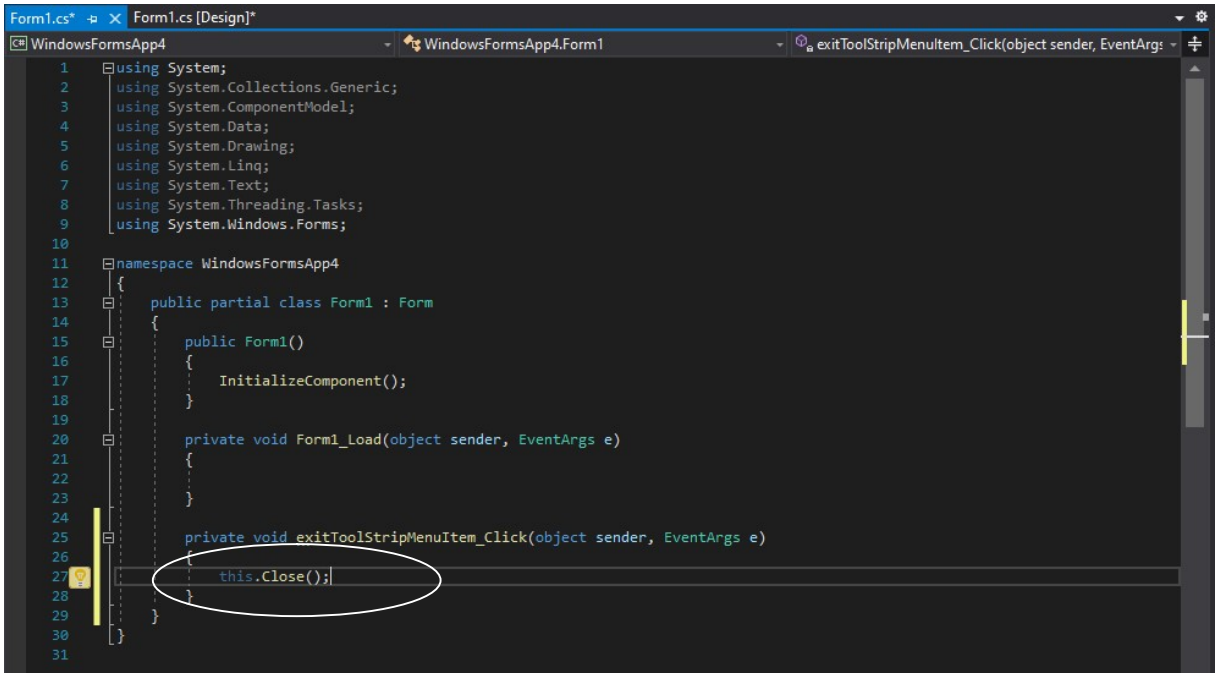


Рисунок 2.4

2.2 Створення найпростіших форм для роботи з даними

Розглянемо процес створення найпростіших форм для роботи з даними на прикладі форм, що забезпечують роботу з таблицями Suppliers, IndividualEntrepreneurs та LegalEntities.

1. Створити нове джерело даних. Для цього відкрийте вікно Data Sources (якщо воно не відкрите, у головному меню вибрати пункт View, Other Windows і в підменю вибрати пункт Data Sources). Клацнути мишею по пункту Add New Data Source і у вікні Data Source Configuration Wizard вибрати Database як тип джерела даних. Натиснути кнопку Next.

2. У наступному вікні натиснути кнопку New Connection і в вікні Add Connection встановити параметри з'єднання (рисунок 8.5). При цьому необхідно врахувати, що як Server name слід вводити дані, що відповідають комп'ютеру, на якому виконується робота. У полі Select or enter a database name слід ввести ім’я бази даних, наприклад, delivery. Після введення параметрів слід перевірити правильність підключення, натиснувши кнопку Test Connection. У разі успішного тестового підключення слід натиснути кнопку ОК. Вікно Add Connection буде закрито і у вікні Data Source Configuration Wizard як активне буде встановлено нове з'єднання. Після цього необхідно натиснути кнопку Next.

3. Створене з'єднання можна зберегти, наприклад, з ім'ям deliveryConnectionString, натиснувши кнопку Next у наступному вікні. Потім потрібно вибрати об'єкти бази даних, які можуть використовуватись під час роботи (рекомендується вибрати усі). Натиснути кнопку Finish.

4. У результаті буде створено джерело даних deliveryDataSet, яке з'явиться у списку джерел даних у вікні Data Sources. У списку об'єктів цього джерела даних повинні бути всі об'єкти бази даних, створені в процесі виконання попередніх лабораторних робіт.

5. Використовуючи створене джерело даних, створимо найпростішу екранну форму, що дозволяє кінцевому користувачеві працювати з таблицею Suppliers. Для цього в головному меню виберемо пункт Project і у вертикальному меню – Add Form (Windows Forms)… . В результаті на

екрані з'явиться вікно, що дозволяє визначити тип нової форми та її назву (рисунок 2.5). Для додавання форми до проекту потрібно натиснути кнопку Add.

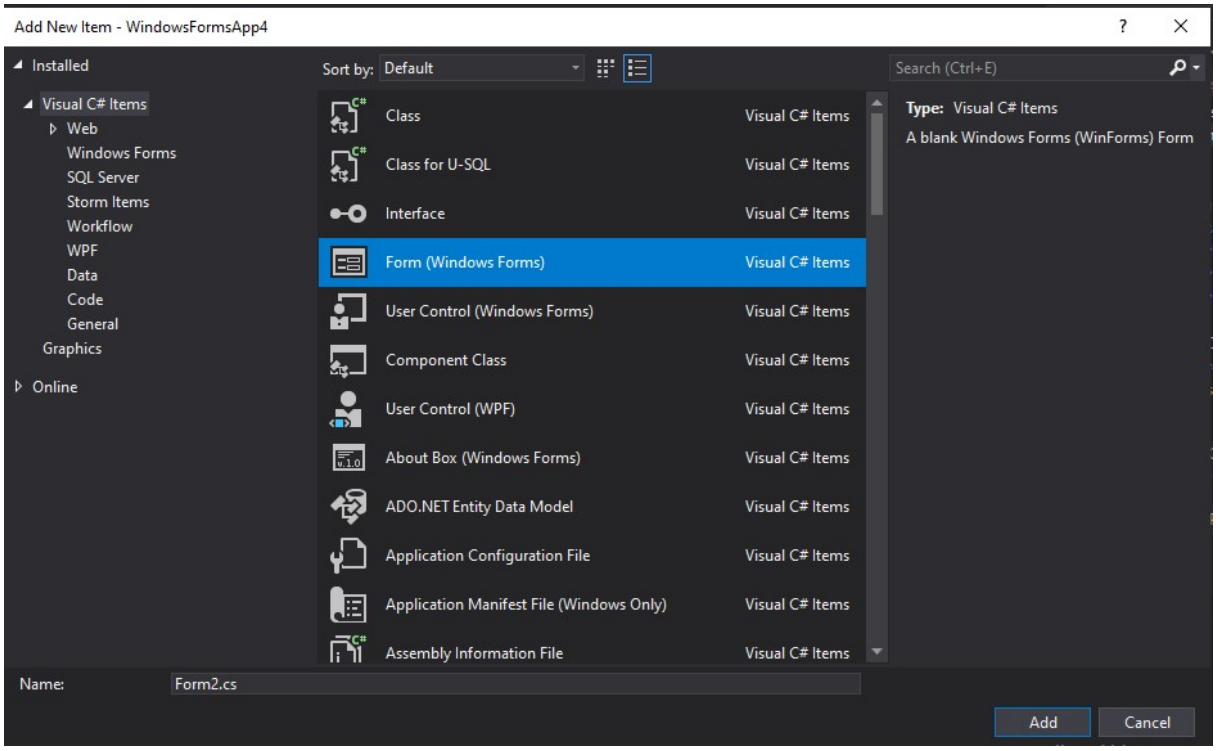


Рисунок 2.5

6. В результаті до проекту буде додано нову форму. Вибравши у списку об'єктів джерела даних таблицю Suppliers, потрібно за допомогою миші перетягнути її на форму. В результаті на формі з'являться об'єкти типу DataGridView та BindingNavigator, за допомогою яких можна переглянути вміст даної таблиці та виконати операції маніпулювання даними (рисунок 2.6).

7. Тепер створену форму потрібно підключити до спільного меню програми. Для цього потрібно перейти на роботу з головною формою (Form1) (в режимі Design) і для пункту меню Suppliers (у вертикальному меню, що відповідає пункту Data головного меню) створити підменю. Перший пункт цього підменю слід назвати General Information (рисунок 2.7). Подвійне натискання миші на цьому пункті меню відкриє вікно коду,

в якому можна ввести текст функції, що забезпечує відкриття форми при виборі цього пункту меню (рисунок 2.8).

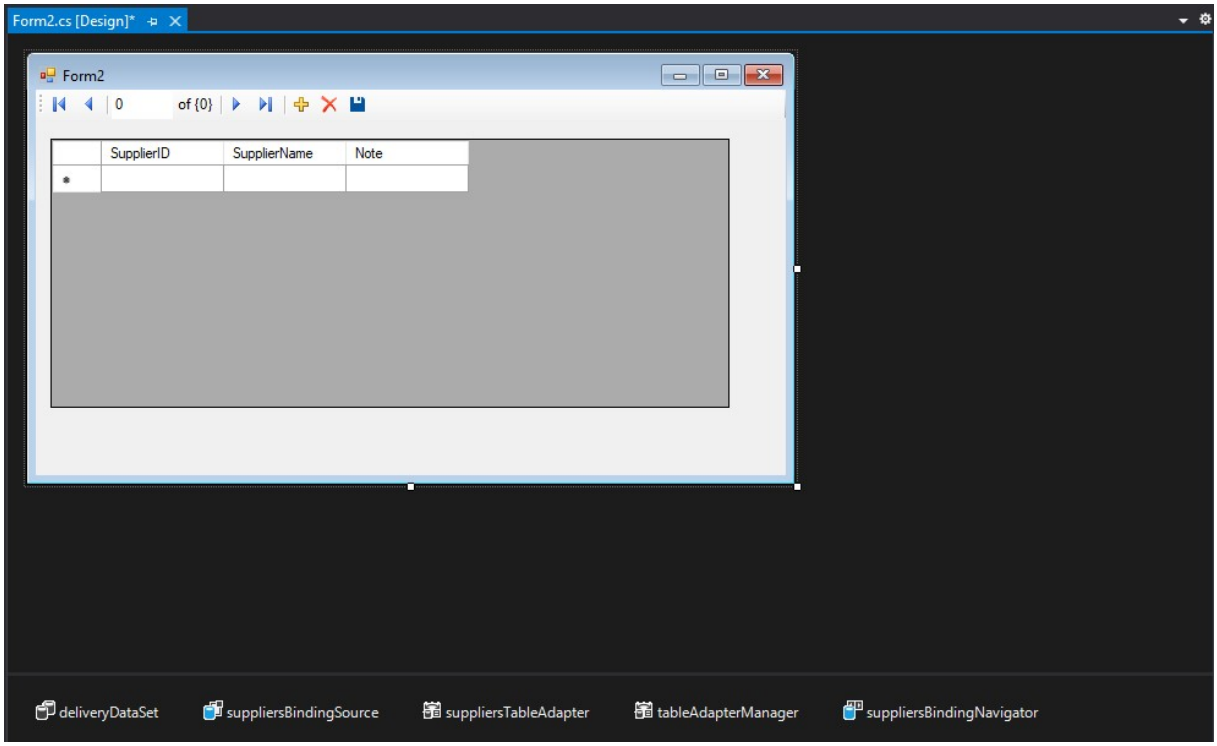


Рисунок 2.6

Рисунок 2.7

```
private void generalInformationToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form2 frm = new Form2();
    frm.Show();
}
```

Рисунок 2.8

8. Потрібно запустити застосунок та, вибравши відповідний пункт меню, вивести на екран форму, що містить загальні дані про постачальників (рисунок 2.9). Перевірити працездатність форми шляхом зміни даних про постачальників (додавання нового постачальника, видалення постачальника, зміна даних постачальника). Зміни обов'язково слід зберігати. Щоб зберегти зміни, потрібно використовувати кнопку Save. Перевірити виконання відповідних операцій можна за допомогою застосунку SQL Server Management Studio, а також шляхом закриття та повторного відкриття цієї форми.

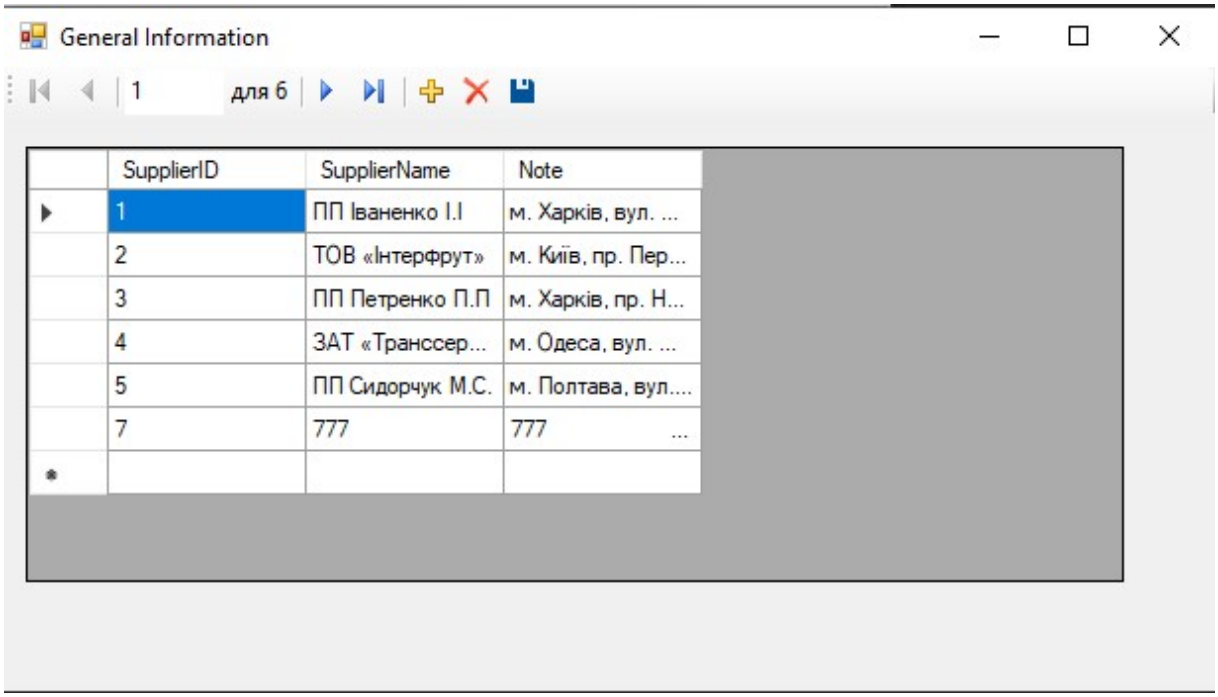


Рисунок 2.9

9. Постачальники як суб'єкти підприємницької діяльності можуть бути як фізичними, так і юридичними особами. Для того щоб працювати з такою інформацією, створимо окремі форми. Розглянемо процес створення такої форми з прикладу форми, призначеної для роботи з даними постачальників – фізичних осіб. Насамперед змінимо меню головної форми, доповнивши його пунктами IndividualEntrepreneurs та LegalEntities

(рисунок 2.10). Для пункту меню IndividualEntrepreneurs зробимо форму, загалом аналогічну до форми, підключеної до пункту General Information.

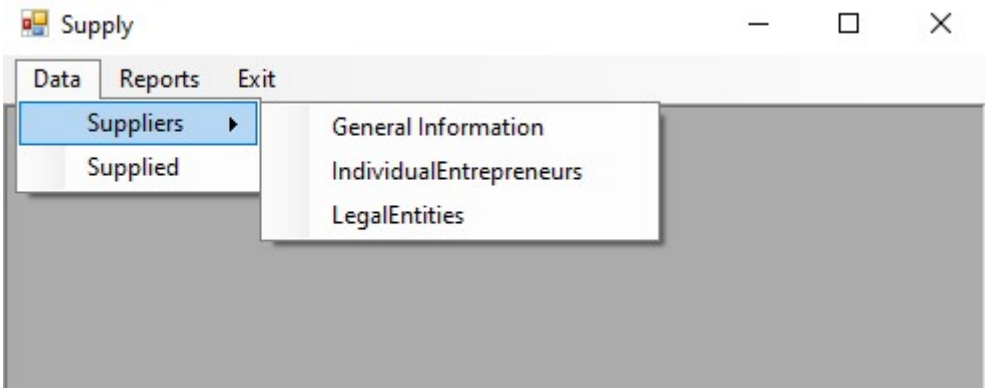


Рисунок 2.10

10. Послідовність дій при створенні такої форми аналогічна послідовності дій при створенні форми для роботи із загальними відомостями про постачальників. Припустимо, що при створенні нової форми їй було присвоєно ім'я Form3.cs. Після створення форми та розміщення на ній об'єктів управління її необхідно підключити до головної форми. Після цього застосунок потрібно запустити та перевірити працездатність нової форми. Її зовнішній вигляд може бути, наприклад, таким, як показано на рисунку 2.11. Очевидно, що користувач, який працює з цією формою, повинен пам'ятати коди відповідних постачальників, що не завжди можливо. У зв'язку з цим модернізуємо форму таким чином, щоб введення коду постачальника здійснювалося не вручну, а шляхом вибору коду зі списку кодів, що вже є в таблиці Suppliers.

11. Треба відкрити цю форму в режимі Design, клацнути по об'єкту DataGridView і у вікні властивостей знайти властивість Columns. Натиснути кнопку «…» для того, щоб отримати доступ до стовпців DataGridView. В результаті на екрані з'явиться вікно, що містить список стовпців та їх властивості (рисунок 2.12). Виберемо стовпець

КодПостачальника та властивість ColumnType (рисунок 2.12). Встановимо для цього стовпця тип DataGridViewComboBoxColumn (рисунок 2.12).

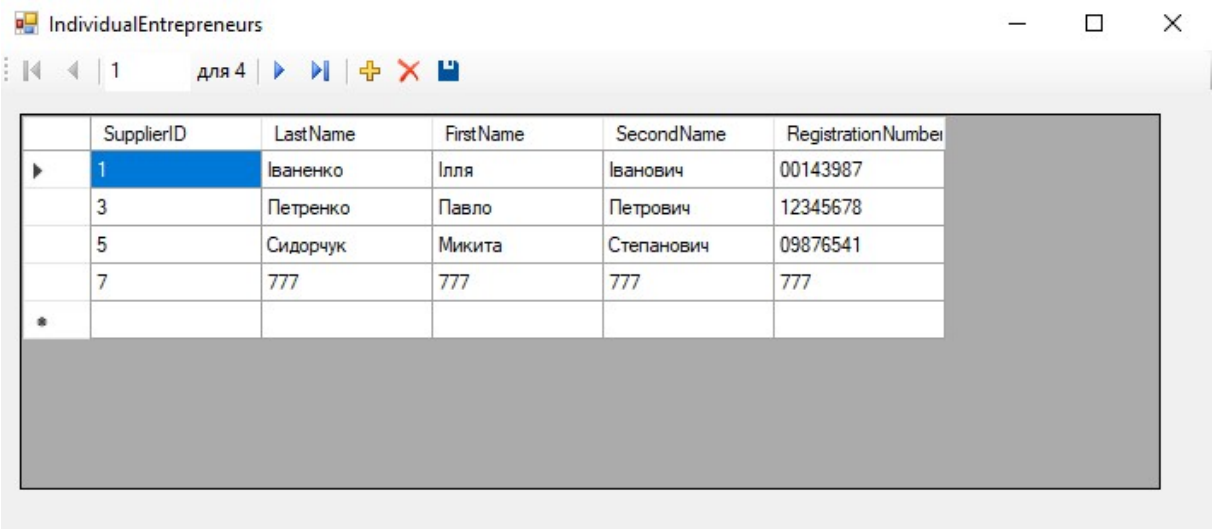


Рисунок 2.11

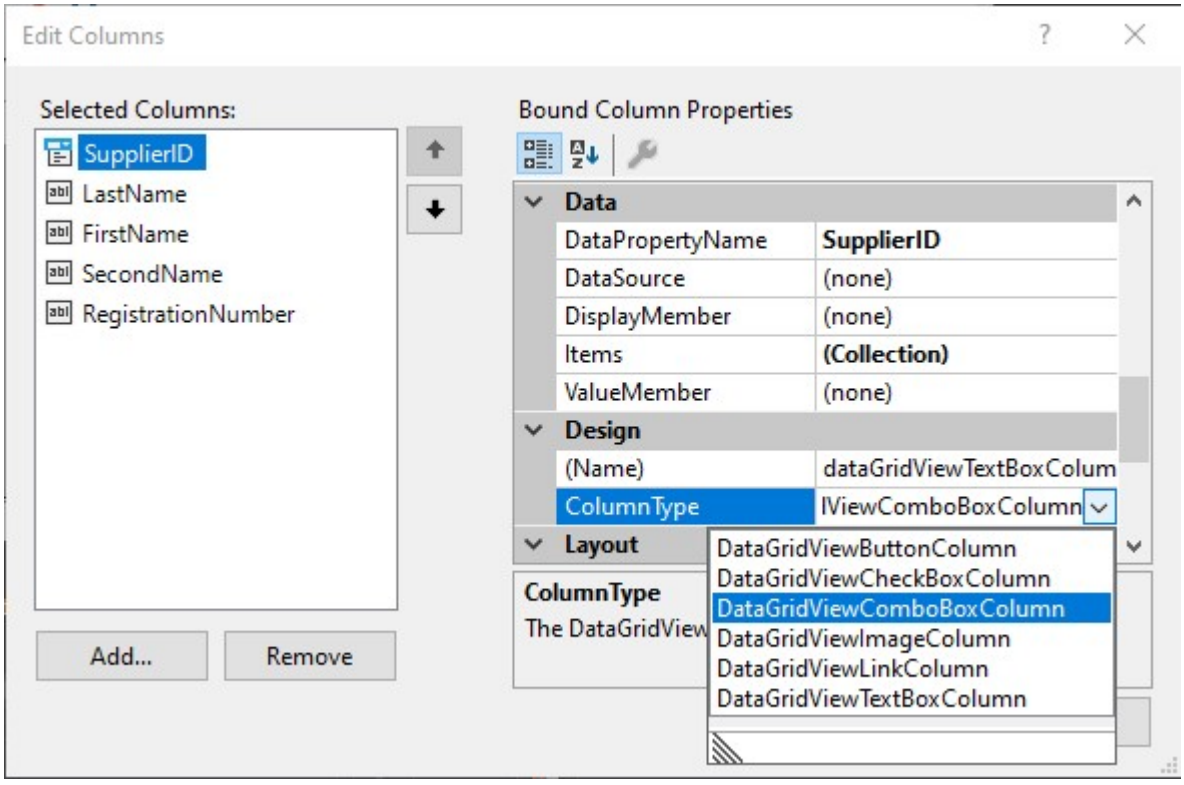


Рисунок 2.12

12. Встановимо для цього стовпця значення властивості DataSource. Для цього викличемо список джерел даних та виберемо джерело даних, яке відповідає таблиці Suppliers (рисунок 2.13).

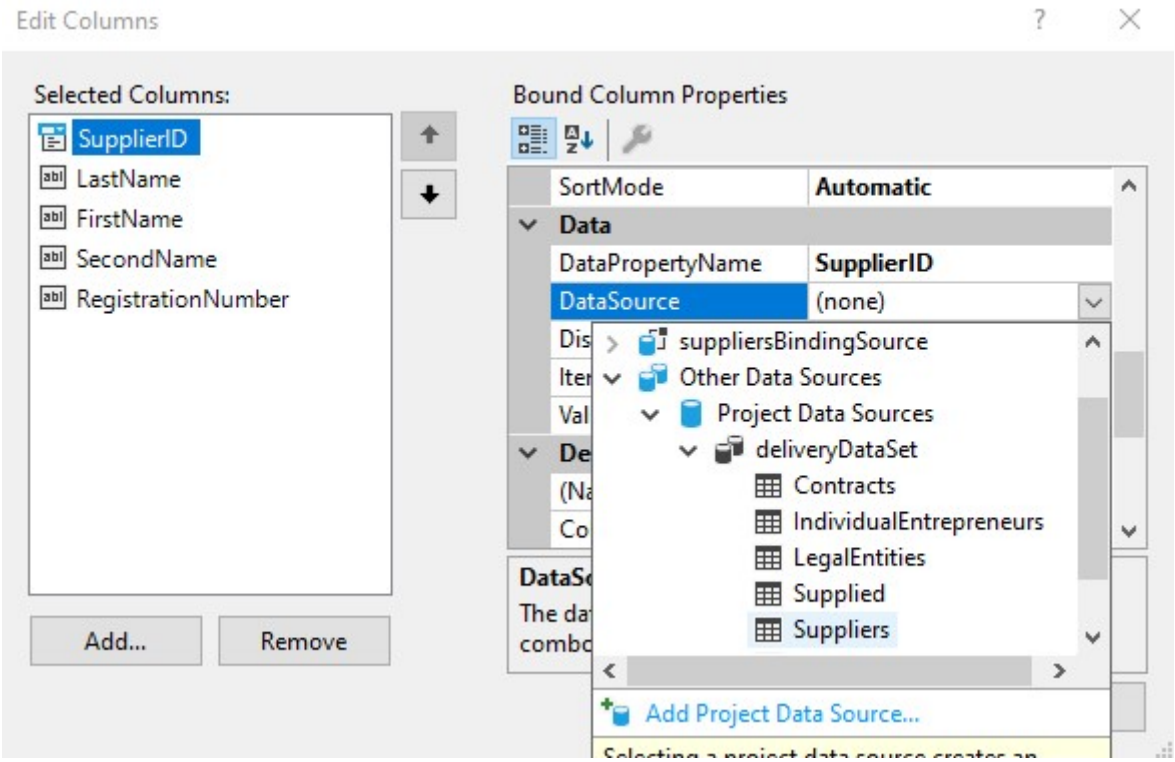


Рисунок 2.13

13. Для властивості DisplayMember встановити значення SupplierID (рисунок 2.14). Після цього вікно Edit Columns закрити натиснувши кнопку ОК. В результаті в стовпці SupplierID має з'явитися кнопка поля зі списком.

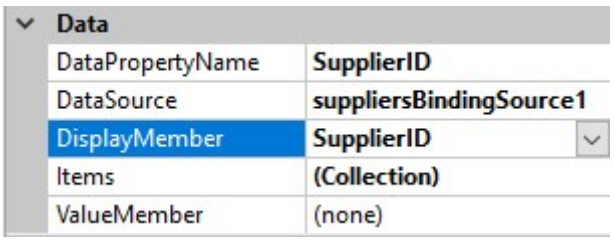


Рисунок 2.14

14. Запустити застосунок та перевірити працездатність зміненої форми. Її зовнішній вигляд може бути, наприклад, таким, як показано на рисунку 2.15.

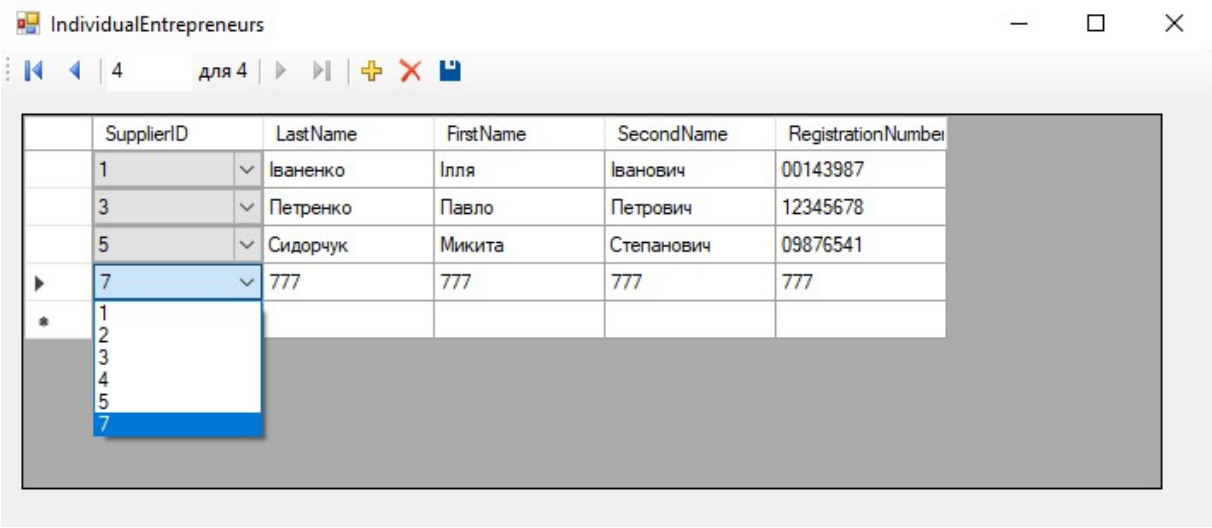


Рисунок 2.15

15. Перевірити працездатність форми під час введення даних. При цьому можна перевірити працездатність не тільки форми, але й раніш створеного тригера. Для цього потрібно спробувати ввести довільні дані про постачальника 2 як про фізичну особу (рисунок 2.16) та спробувати зберегти ці дані. В цьому випадку на екран буде виведено повідомлення, пов'язане з роботою тригера (рисунок 2.17). Новий запис не буде збережено.

16. Аналогічно можна розробити та підключити у застосунок форму, що забезпечує роботу з даними про постачальників – юридичних осіб.

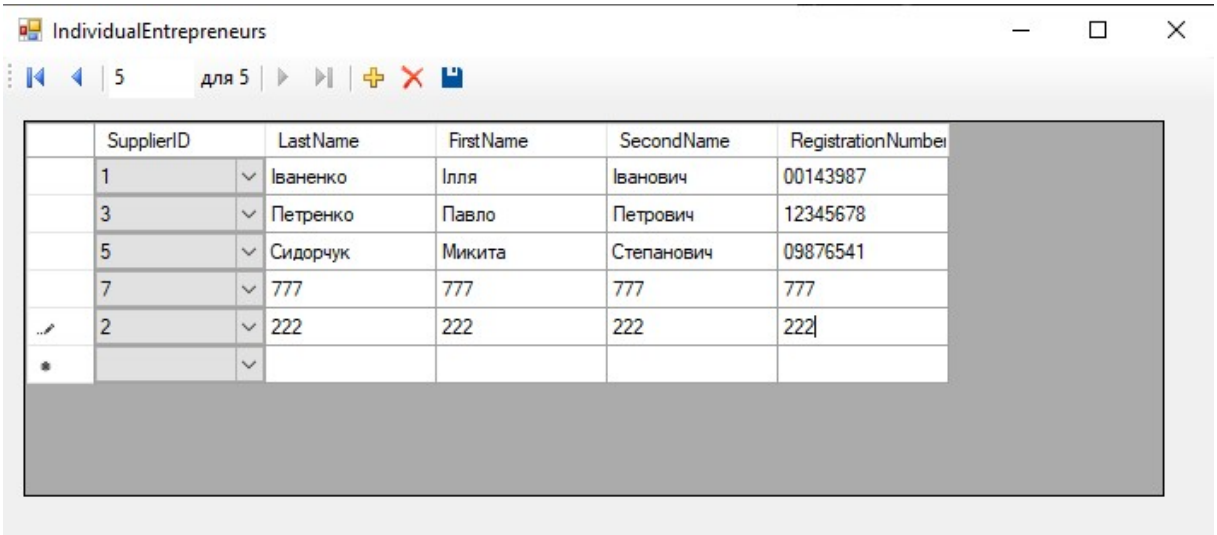


Рисунок 2.16

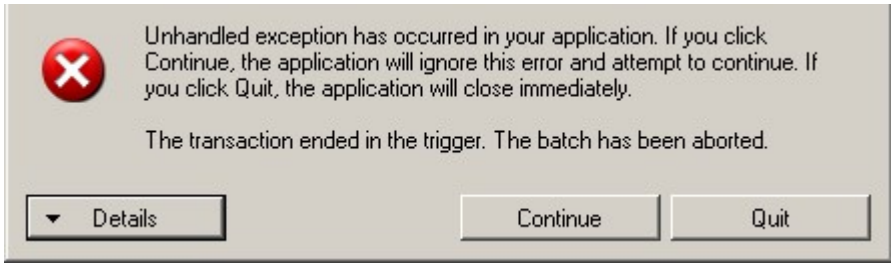


Рисунок 2.17

У цілому розроблені форми є працездатними, але вони забезпечують користувачеві лише мінімальний набір засобів до роботи з даними. Крім того, ці форми мають дуже мало засобів контролю за діями користувача. Наприклад, при натисканні кнопки видалення запису, застосунок не вимагає від користувача підтвердження видалення, отже помилкове видалення може призвести до втрати даних. Також, якщо користувач після вводу нових даних не натисне кнопку зберігання (наприклад, забувши про це), та закриє форму, то нові дані не будуть збережені та ін.

2.3 Створення форми, що забезпечує роботу з кількома таблицями

Розглянемо процес створення форми, за допомогою якої кінцевий користувач буде мати можливість переглядати та модифікувати дані про договори та продукцію, яку було закуплено на підставі цих договорів.

1. Створити нову форму, наприклад, з ім'ям Form5.cs. Для властивості Text встановити значення Supplied.

2. Розмістити на формі об'єкт TabControl (для цього потрібно вибрати цей об'єкт у панелі Toolbox та перетягнути на форму). Для властивості Name встановити значення tabControl1. Об'єкт повинен містити дві вкладки з іменами tabPage1 та tabPage2 відповідно. Для властивості Text цих вкладок встановити значення List of contracts та New contract відповідно. Підключити нову форму до пункту меню Supplied. Текст функції, що забезпечує таке підключення, наведено на рисунку 2.18. Запустити застосунок та перевірити працездатність нової форми. Форма може мати вигляд, який наведено на рисунку 2.19. Закрити програму і повернутися в режим Design для Form5.cs.

```
private void suppliedToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form5 frm = new Form5();
    frm.Show();
}
```

Рисунок 2.18

3. Розмістити на вкладці tabPage1 об'єкт типу DataGridView. В результаті форма (в режимі Design) матиме вигляд, наведений на рисунку 2.20). Для об'єкта встановити ім'я dataGridView1.

4. Клацнути по кнопці у верхньому правому кутку об'єкта dataGridView1 для того, щоб відкрити вікно DataGridView Tasks (рисунок 2.21). За допомогою цього вікна визначити джерело даних для dataGridView1 (рисунок 2.21), вибравши його зі списку джерел даних.

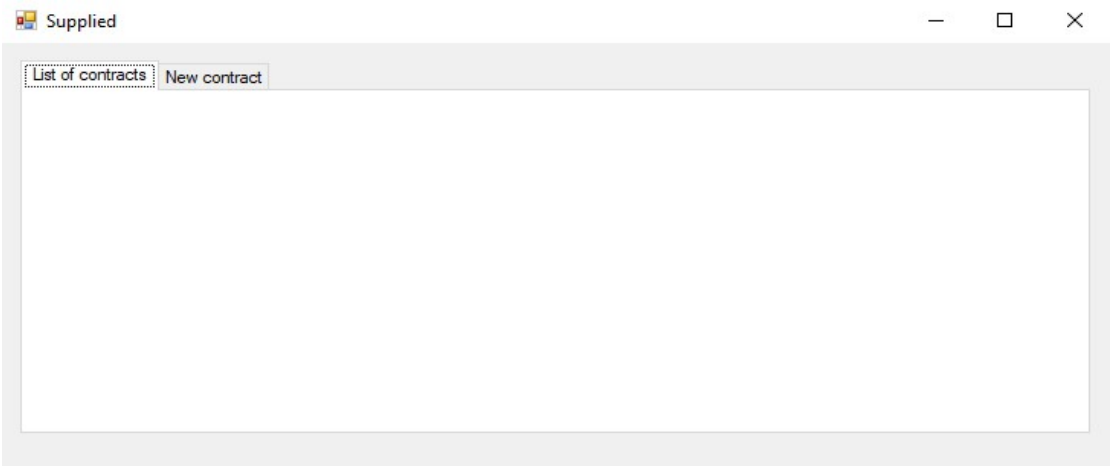


Рисунок 2.19

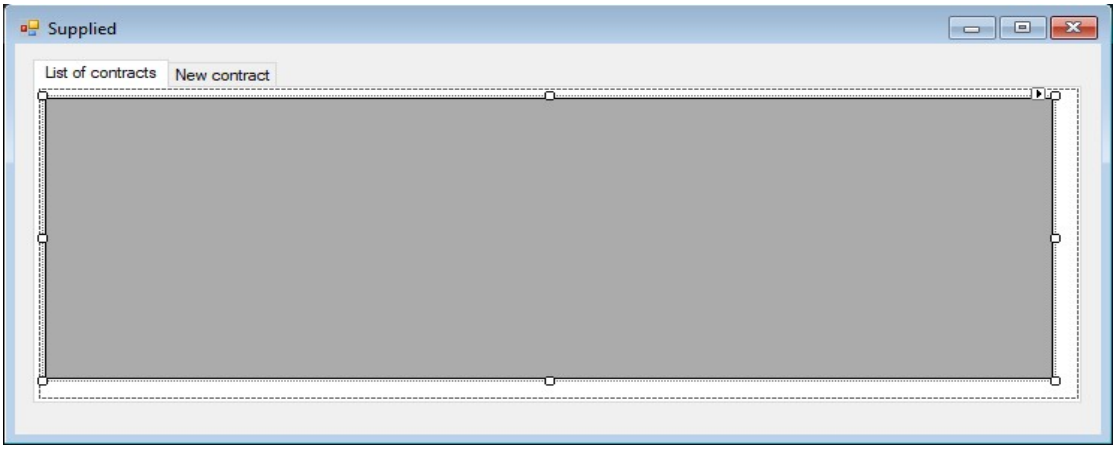


Рисунок 2.20

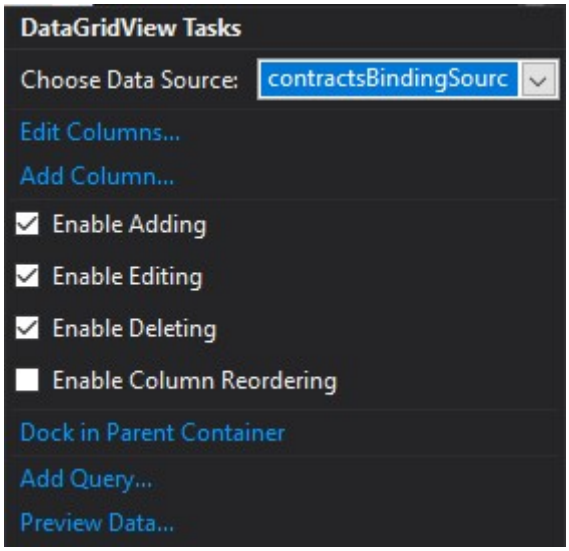


Рисунок 2.21

5. Перевірити працездатність зміненої форми. Форма може мати вигляд, аналогічний наведеному на рисунку 2.22. Закрити застосунок і повернутися в режим Design для Form5.cs. Встановити для властивості об'єкта ReadOnly dataGridView1 значення True.

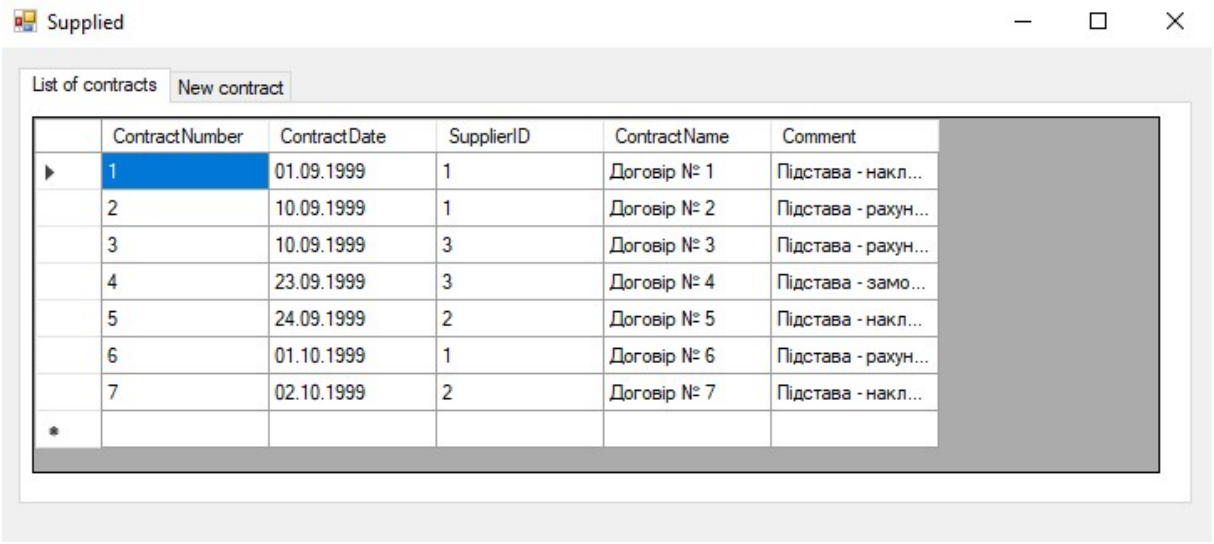


Рисунок 2.22

6. Як видно з рисунку 2.22, створений список договорів не цілком зручний для роботи. Це, зокрема, стосується інформації про постачальника. Припустимо, що від користувачів надійшла вимога, щоб замість коду постачальника у списку договорів виводилася назва постачальника, а поруч, у дужках для постачальників – фізичних осіб має виводитися прізвище, ім’я та по-батькові, а для юридичних осіб – податковий номер. Щоб отримати такі дані, створимо у базі даних представлення, що дозволяє сформувати таку інформацію про постачальників. Текст визначального запиту такого представлення може мати вигляд, наведений на рисунку 2.23. Таке представлення необхідно зробити за допомогою SQL Server Management Studio та зберегти з ім'ям View_3. Потім у вікні Data Sources потрібно клацнути правою кнопкою миші по джерелу даних deliveryDataSet і в меню вибрати пункт Configure DataSet with Wizard ... У вікні потрібно повторно відзначити пункт Views.

В результаті новостворене представлення має з'явитися у списку об'єктів джерела даних. Змінене джерело даних треба зберігти.

```
SELECT      dbo.Suppliers.SupplierID, RTRIM(CAST(dbo.Suppliers.SupplierName AS char(20))) + ' (' +
                      ISNULL(RTRIM(dbo.IndividualEntrepreneurs.LastName) + ' ' + RTRIM(dbo.IndividualEntrepreneurs.FirstName)
                      + ' ' + RTRIM(dbo.IndividualEntrepreneurs.SecondName), RTRIM(dbo.LegalEntities.TaxNumber)) + ')' AS Supplier
FROM        dbo.Suppliers LEFT OUTER JOIN
              dbo.LegalEntities ON dbo.Suppliers.SupplierID = dbo.LegalEntities.SupplierID LEFT OUTER JOIN
              dbo.IndividualEntrepreneurs ON dbo.Suppliers.SupplierID = dbo.IndividualEntrepreneurs.SupplierID
```

Рисунок 2.23

7. Для об'єкта dataGridView1 відкрити вікно DataGridView Tasks та вибрати пункт Edit Columns… Змінити назви стовпців, якщо це потрібно (це можна зробити, змінивши властивість HeaderText для кожного стовпця (рисунок 2.24)). Також можна змінити ширину стовпців (це можна зробити, змінивши властивість Width для кожного стовпця). Ширину шпальт встановити виходячи з реальних розмірів даних.

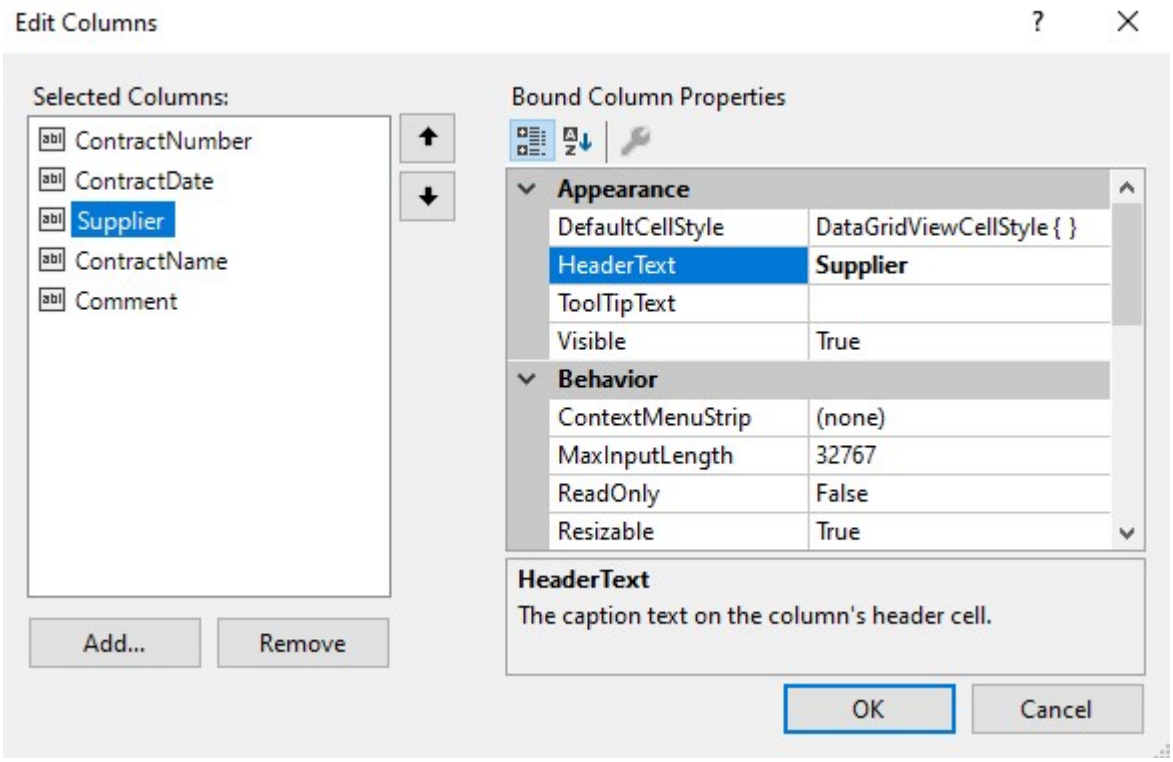


Рисунок 2.24

8. Вибрати стовпець Supplier та властивість ColumnType. Установіть для цього стовпця тип DataGridViewComboBoxColumn. Встановити для цього стовпця значення властивості DataSource. Для цього викликати список джерел даних та вибрати джерело даних, що відповідає представленню View_3 (рисунок 2.25). Для властивості DisplayMember встановити значення Supplier, а властивості ValueMember – SupplierID (рисунок 2.25). Після цього вікно Edit Columns закрити натиснувши кнопку OK. В результаті в стовпці Supplier має з'явитися кнопка поля зі списком. Перевірити працездатність зміненої форми. Форма може мати вигляд, аналогічний наведеному на рисунку 2.26. Закрити програму і повернутися в режим Design для Form5.cs.

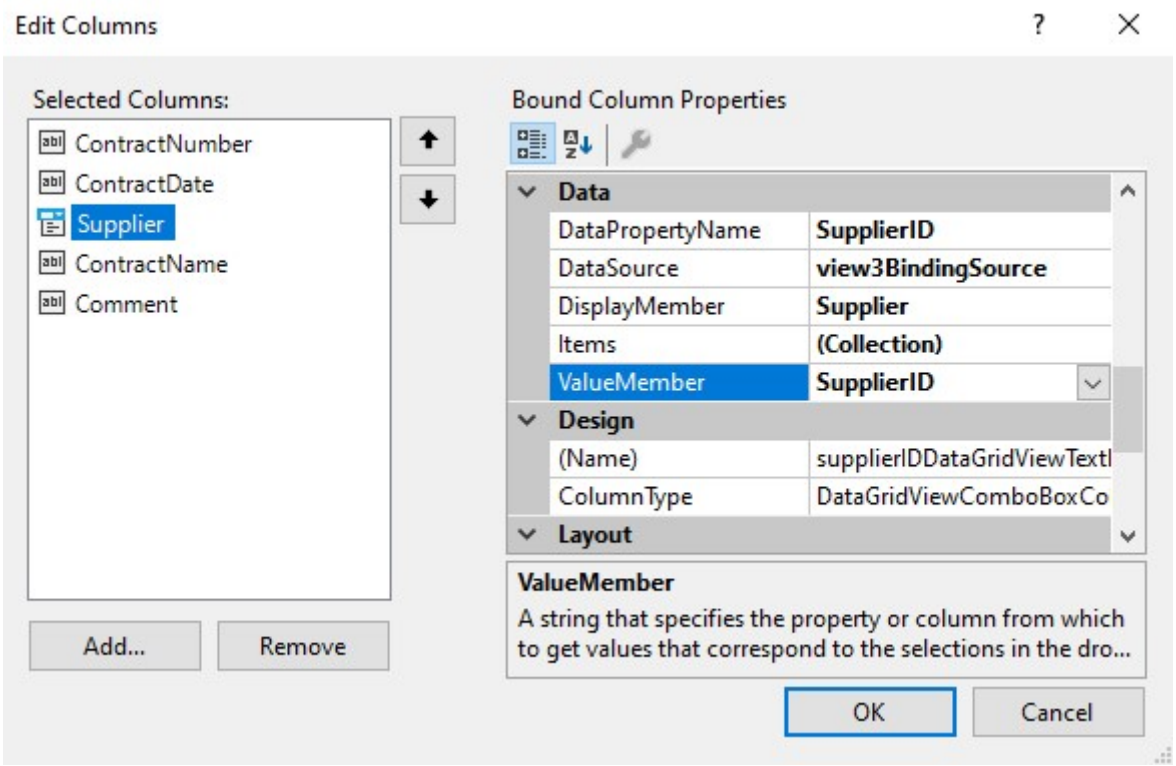


Рисунок 2.25

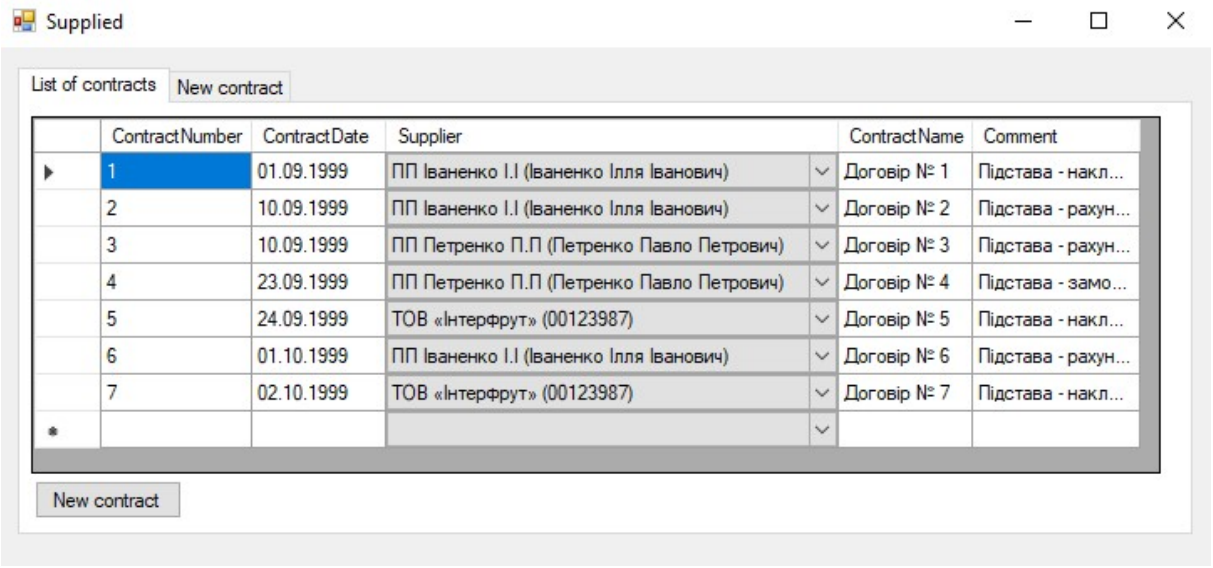


Рисунок 2.26

9. Список договорів у цій формі можна лише переглядати. Зміна списку договорів неможлива. Щоб забезпечити можливість користувачеві створювати нові договори, внесемо зміни у форму. На вкладці List of contracts створимо кнопку. Для цього в панелі Toolbox виберемо об'єкт типу Button та розмістимо його на формі (рисунок 2.26). Для властивості Text необхідно встановити значення New contract. Для події Click цієї кнопки потрібно створити функцію – обробник події. Текст цієї функції наведено рисунку 2.27. Перевірити працездатність зміненої форми. В результаті натискання кнопки має відкриватися вкладка New contract. Закрити програму і повернутися в режим Design для Form5.cs.

Примітка. З функціональної точки зору наявність кнопки New contract у формі не є обов'язковою. Вона додана з метою ілюстрації можливостей керування об'єктами форми.

```
private void button1_Click(object sender, EventArgs e)
{
    this.tabControl1.SelectedTab = tabPage2;
}
```

Рисунок 2.27

10. Відкрити вкладку New contract (tabPage2). На цій вкладці розмістити чотири об'єкти типу Label, за допомогою яких формуються коментарі для полів введення даних (рисунок 2.28). Для введення дати укладання договору можна використовувати об'єкт типу DateTimePicker. Для цього об'єкта встановлюється ім'я dateTimePicker1. Для введення назви договору та коментаря до договору необхідно використовувати об'єкт типу TextBox. Для цих об'єктів встановлюються назви textBox1 та textBox2. Для введення даних про постачальника необхідно використовувати об'єкт типу ComboBox. Для цього об'єкта встановлюється ім'я comboBox1. Також на вкладці потрібно розмістити два об'єкти типу Button. Для властивості Text цих кнопок потрібно встановити значення Save та Cancel. Для імен цих об'єктів встановити значення button2 та button3 відповідно.

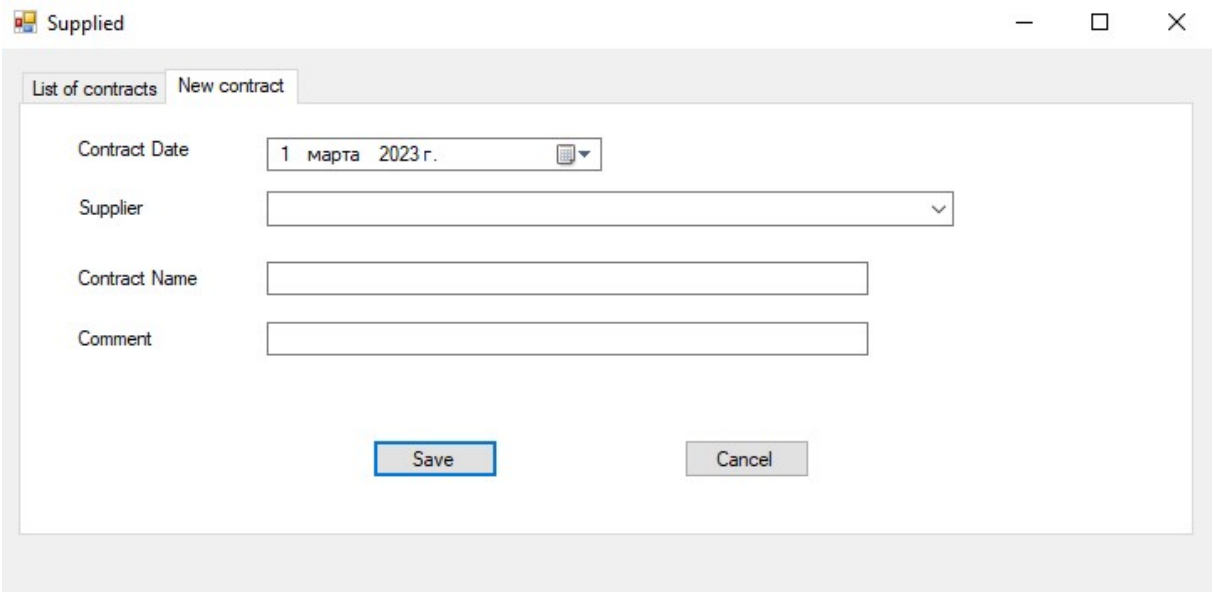


Рисунок 2.28

11. Для об'єкта comboBox1 потрібно встановити джерело даних – список постачальників. Для цього потрібно клацнути мишею по об'єкту і натиснути кнопку, що з'явиться у верхньому правому кутку об'єкта. Внаслідок цього з'явиться вікно ComboBox Tasks. У цьому вікні потрібно увімкнути прапорець для пункту Use data bound items (рисунок 2.29), а

потім встановити джерело даних, а також значення для властивостей DisplayMember та ValueMember (рисунок 2.29). Як джерело даних можна використовувати уявлення View_3.

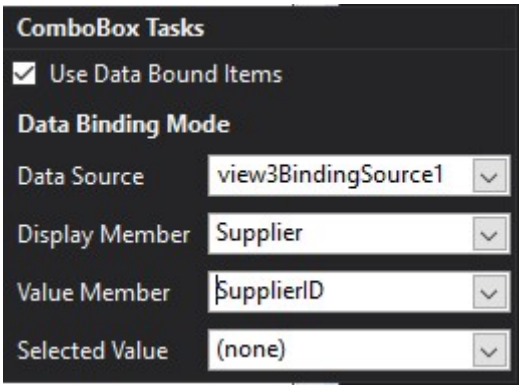


Рисунок 2.29

12. Для об'єкта button2 (кнопка Save) створити функцію – обробник події Click. Текст функції наведено рисунку 2.30.

```
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        int codeSupplier = int.Parse(this.comboBox1.SelectedValue.ToString());
        DateTime datedgvr = Convert.ToDateTime(this.dateTimePicker1.Text);
        string dgvrName = Convert.ToString(this.textBox1.Text);
        string dgvrComment = Convert.ToString(this.textBox2.Text);
        conn = new SqlConnection();
        conn.ConnectionString = "integrated security=SSPI;data source=\".\"; persist security info=False; initial catalog = delivery";
        conn.Open();
        SqlCommand myCommand = conn.CreateCommand();
        myCommand.CommandText = "INSERT INTO Contracts (SupplierID,ContractDate,ContractName,Comment) VALUES (@codeSupplier,@datedgvr,@dgvrName,@dgvrComment)";
        myCommand.Parameters.Add("@codeSupplier", SqlDbType.Int, 4);
        myCommand.Parameters["@codeSupplier"].Value = codeSupplier;
        myCommand.Parameters.Add("@datedgvr", SqlDbType.DateTime, 8);
        myCommand.Parameters["@datedgvr"].Value = datedgvr;
        myCommand.Parameters.Add("@dgvrName", SqlDbType.NVarChar, 50);
        myCommand.Parameters["@dgvrName"].Value = dgvrName;
        myCommand.Parameters.Add("@dgvrComment", SqlDbType.NVarChar, 50);
        myCommand.Parameters["@dgvrComment"].Value = dgvrComment;

        int UspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
        if (UspeshnoeIzmenenie != 0)
        {
            MessageBox.Show("Changes complited", "Record modified");
        }
        else
        {
            MessageBox.Show("Changes didn't complited", "Record modified");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
    finally
    {
        conn.Close();
    }
    this.contractsTableAdapter.Fill(this.deliveryDataSet.Contracts);
    this.tabControl1.SelectedTab = tabPage1;
}
}
```

Рисунок 2.30

Крім введення тексту функції, також потрібно змінити програмний код. Зміни наведено на рисунку 2.31 (цей код знаходиться на початку програмного коду форми).

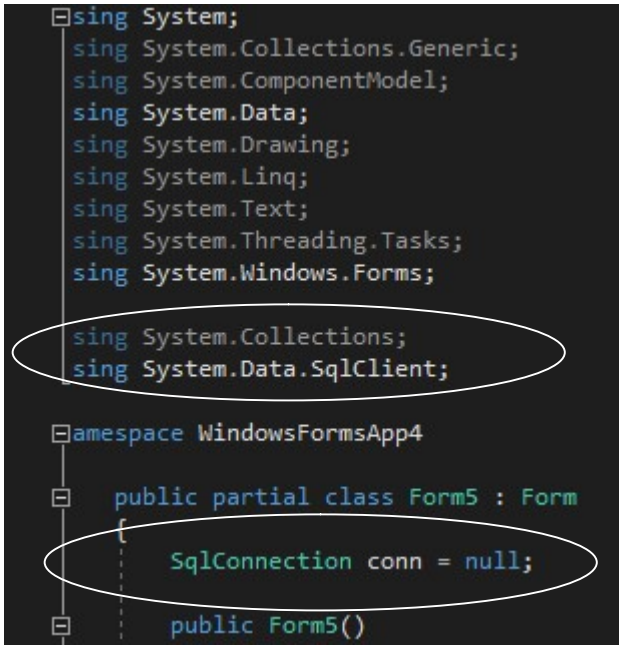


Рисунок 2.31

13. Перевірити працездатність зміненої форми. Для цього потрібно запустити застосунок та ввести дані нового договору (наприклад, як на рисунку 2.32). Натиснути кнопку Save. У разі успішного введення даних на екран буде виведено повідомлення (рисунок 2.33). В результаті вкладку New contract буде закрито та відкрито вкладку List of contracts, в якій з'явиться новий договір (рисунок 2.34).

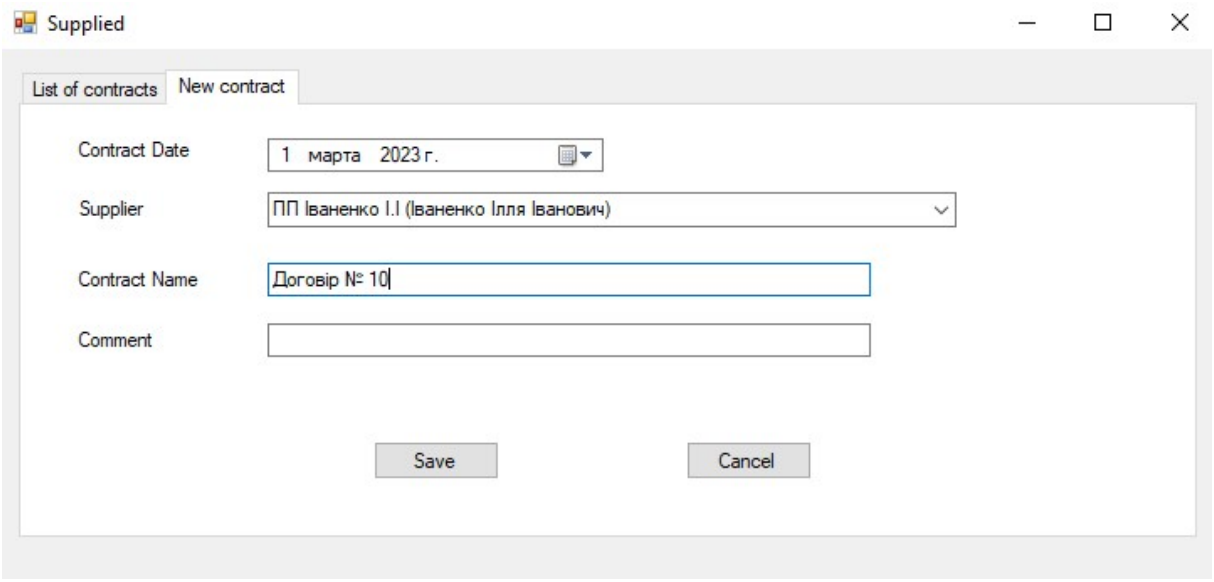


Рисунок 2.32

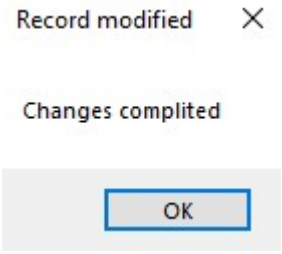


Рисунок 2.33

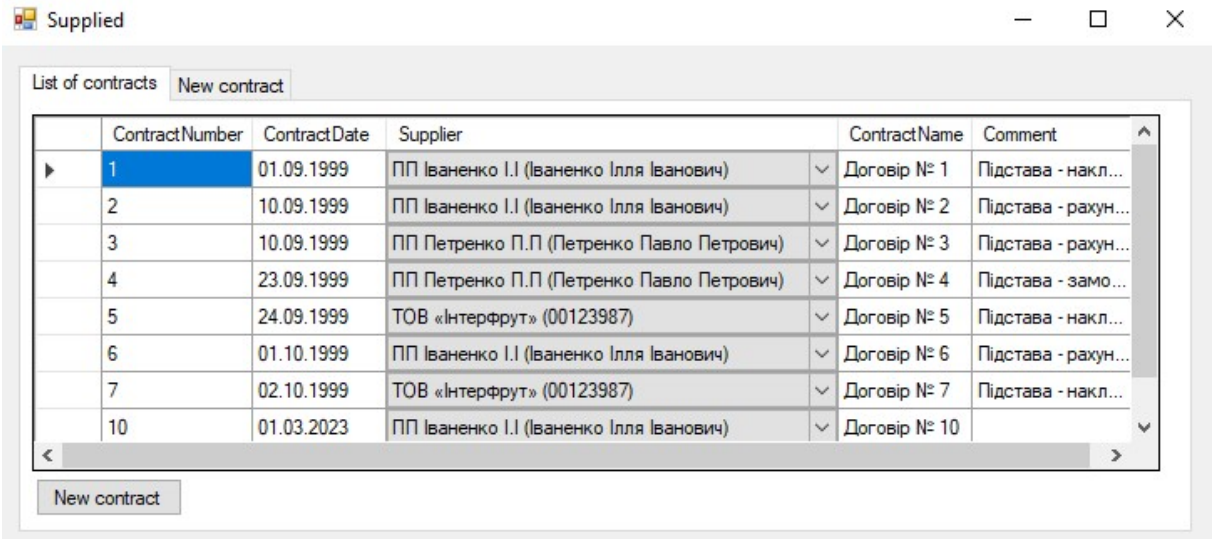


Рисунок 2.34

14. Закрити програму та повернутися в режим Design для Form5.cs. На вкладці New contract для об'єкта button3 (кнопка Cancel) створити функцію – обробник події Click. Текст функції наведено на рисунку 2.35. Перевірити роботу цієї кнопки (при її натисканні закривається вкладка New contract та відкривається вкладка List of contracts без додавання нового договору).

```
private void button3_Click(object sender, EventArgs e)
{
    this.tabControl1.SelectedTab = tabPage1;
}
```

Рисунок 2.35

15. У процесі роботи з даними про закупівлі продукції може виникнути необхідність видалення раніше укладених договорів. Для цього на вкладці List of contracts створимо відповідну кнопку. Для властивості Text цієї кнопки необхідно встановити значення Delete contract. Для цього об'єкта встановити ім'я button4. Для об'єкта button4 створити функцію – обробник події Click. Текст функції наведено на рисунку 2.36.

16. Перевірити роботу цієї кнопки. Для видалення договору потрібно вибрати договір у списку договорів (наприклад, останній створений договір) та натиснути кнопку Delete contract. В результаті на екрані з'явиться вікно (рисунок 2.37), за допомогою якого потрібно підтвердити видалення договору або відмовитись від цієї операції. При натисканні кнопки Так договір буде видалено та список договорів буде оновлено з урахуванням видалення договору. Таким чином, розроблена форма дозволяє бачити список договорів, додавати до бази даних нові договори та видаляти раніше створені договори.

```
private void button4_Click(object sender, EventArgs e)
{
    int dgvrNumber = int.Parse(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex]
        .Cells["ContractNumberDataGridViewTextBoxColumn"].Value.ToString());
    DialogResult result = MessageBox.Show("Contract Number " + Convert.ToString(dgvrNumber) +
        " will be deleted. Are you sure?", "Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Warning,
        MessageBoxDefaultButton.Button2);
    switch (result)
    {
        case DialogResult.Yes:
        {
         try
          {
           conn = new SqlConnection();
           conn.ConnectionString = "integrated security=SSPI;data source=\".\"; " +
                      "persist security info=False; initial catalog = delivery";
           conn.Open();
           SqlCommand myCommand = conn.CreateCommand();
           myCommand.CommandText = "DELETE FROM Contracts WHERE ContractNumber = @dgvrNumber ";
           myCommand.Parameters.Add("@dgvrNumber", SqlDbType.Int, 4);
           myCommand.Parameters["@dgvrNumber"].Value = dgvrNumber;

           int UspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
           if (UspeshnoeIzmenenie != 0)
              {
               MessageBox.Show("Changes complited", "Record modified");
              }
              else
              {
               MessageBox.Show("Changes didn't complited", "Record modified");
              }
          }
          catch (Exception ex)
          {
           MessageBox.Show(ex.ToString());
          }
          finally
          {
           conn.Close();
          }
          this.contractsTableAdapter.Fill(this.deliveryDataSet.Contracts);
          break;
        }
    }
}
```

Рисунок 2.36

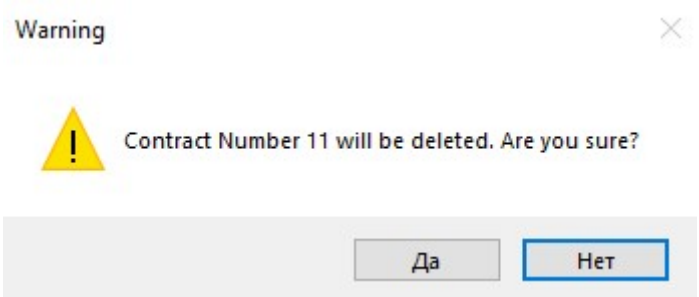


Рисунок 2.37

17. Робота з договорами щодо закупівлі продукції не буде зручною, якщо користувач не матиме можливості працювати з даними про товари, закуплені на підставі договорів. У зв'язку з цим функціональність

розробленої форми потрібно розширити, забезпечивши користувачеві можливість працювати з даними про закуплені товари. Насамперед необхідно збільшити вертикальний розмір форми. Після цього під об'єктом tabControl1 потрібно розмістити приблизно такий самий (за розміром) об'єкт TabControl. Для цього об'єкта встановити ім'я tabControl2. Об'єкт повинен містити дві вкладки з іменами tabPage3 та tabPage4. Для властивості Text цих вкладок встановити значення List of products і Add product відповідно. Перевірити працездатність зміненої форми. Зовнішній вигляд форми наведено на рисунку 2.38.

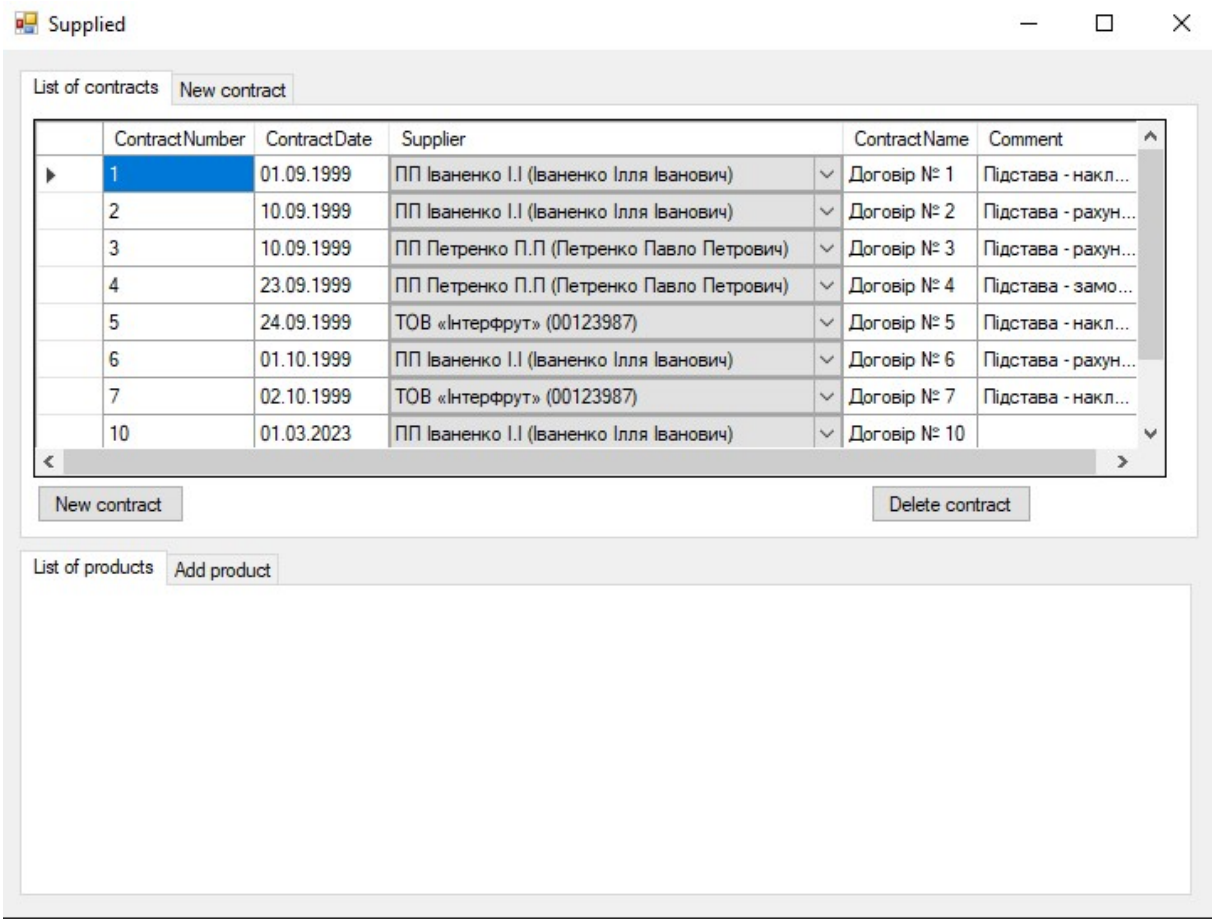


Рисунок 2.38

18. Розмістити на вкладці tabPage3 об'єкт типу DataGridView. Для об'єкта встановити ім'я dataGridView2. Клацнути по кнопці у верхньому правому кутку об'єкта dataGridView2 для того, щоб відкрити вікно

DataGridView Tasks (рисунок 2.39). За допомогою цього вікна визначити джерело даних dataGridView1 (рисунок 2.39), вибравши його зі списку джерел даних. При виборі джерела даних необхідно вибрати джерело, яке дозволить зв'язати дані про договори та поставлені на підставі цих договорів товари (рисунок 2.39). Встановити для властивості об'єкта ReadOnly dataGridView2 значення True.

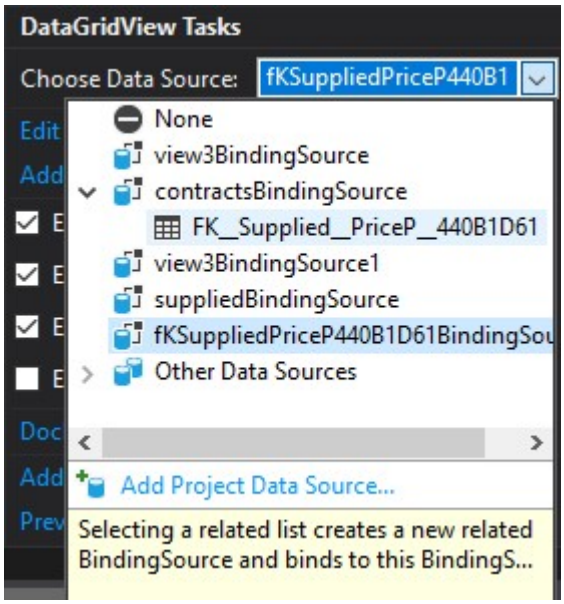


Рисунок 2.39

19. Перевірити працездатність зміненої форми. Зовнішній вигляд форми наведено на рисунку 2.40. Слід звернути увагу, що при виборі договору у списку договорів, у списку товарів має виводитись список товарів, закуплених саме на підставі обраного договору.

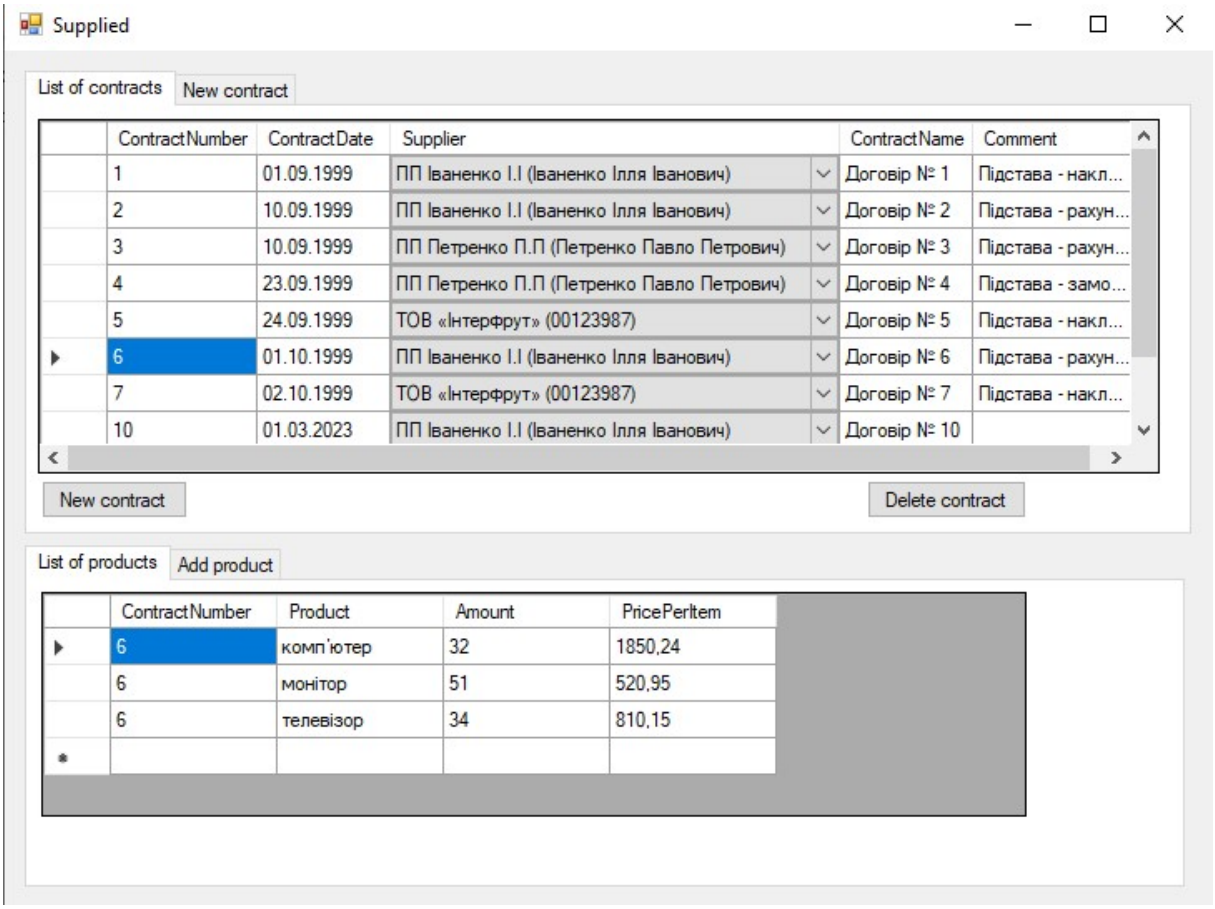


Рисунок 2.40

20. Список товарів у цій формі можна лише переглядати. Зміна списку товарів неможлива. Для того щоб забезпечити можливість користувачеві вводити дані про закуплені товари, внесемо зміни до форми. Для цього необхідно відкрити вкладку Add product (tabPage4). На цій вкладці розміщено три об'єкти типу Label, за допомогою яких формуються коментарі для полів введення даних (рисунок 2.41). Для введення назви товару, кількості одиниць та ціни за одиницю необхідно використовувати об'єкти типу TextBox. Для цих об'єктів встановлюються імена TextBox3, TextBox4, TextBox5 відповідно. Також на вкладці потрібно розмістити два об'єкти типу Button. Для властивості Text цих кнопок потрібно встановити значення Save та Cancel. Для імен цих об'єктів встановити значення button5 та button6 відповідно.

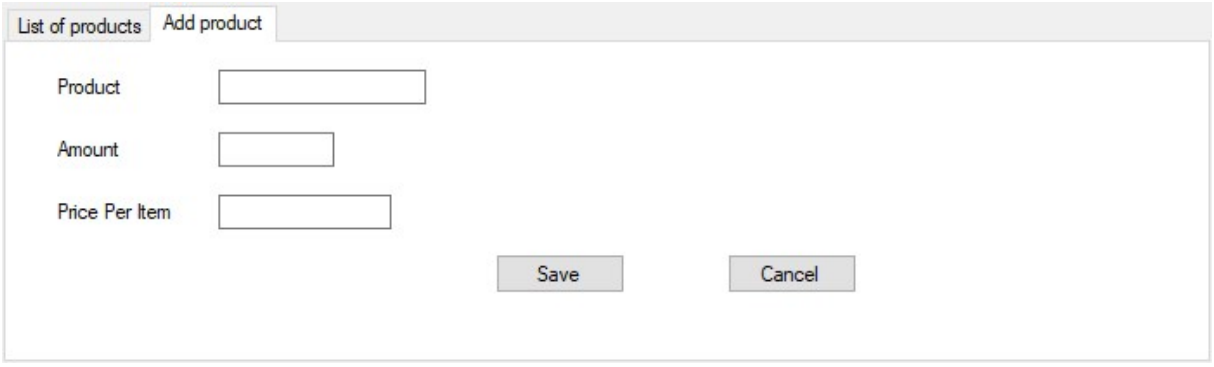


Рисунок 2.41

21. Для об'єкта button5 створити функцію – обробник події Click. Текст функції наведено на рисунку 2.42.

```
private void button5_Click(object sender, EventArgs e)
{
    try
    {
        int dgvrNumber = int.Parse(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex]
           .Cells["ContractNumberDataGridViewTextBoxColumn"].Value.ToString());
        string tovar = Convert.ToString(this.textBox3.Text);
        int tovar_kol = int.Parse(this.textBox4.Text.ToString());
        Decimal tovar_cena = Convert.ToDecimal(this.textBox5.Text);
        conn = new SqlConnection();
        conn.ConnectionString = "integrated security=SSPI;data source=\".\"; " +
                                "persist security info=False; initial catalog = delivery";
        conn.Open();
        SqlCommand myCommand = conn.CreateCommand();
        myCommand.CommandText = "INSERT INTO Supplied (ContractNumber,Product,Amount,PricePerItem) " +
                                        "VALUES (@dgvrNumber,@tovar,@tovar_kol,@tovar_cena)";
        myCommand.Parameters.Add("@dgvrNumber", SqlDbType.Int, 4);
        myCommand.Parameters["@dgvrNumber"].Value = dgvrNumber;
        myCommand.Parameters.Add("@tovar", SqlDbType.NVarChar, 20);
        myCommand.Parameters["@tovar"].Value = tovar;
        myCommand.Parameters.Add("@tovar_kol", SqlDbType.Int, 8);
        myCommand.Parameters["@tovar_kol"].Value = tovar_kol;
        myCommand.Parameters.Add("@tovar_cena", SqlDbType.Decimal, 8);
        myCommand.Parameters["@tovar_cena"].Value = tovar_cena;

        int UspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
        if (UspeshnoeIzmenenie != 0)
        {
            MessageBox.Show("Changes complited", "Record modified");
        }
        else
        {
            MessageBox.Show("Changes didn't complited", "Record modified");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
    finally
    {
        conn.Close();
    }
    this.suppliedTableAdapter.Fill(this.deliveryDataSet.Supplied);
    this.tabControl2.SelectedTab = tabPage3;
}
```

Рисунок 2.42

22. Перевірити працездатність форми. Для цього додати новий товар до будь-якого договору. Припустимо, що товар додається до договору 7. Дані нового товару наведено на рисунку 2.43. Після натискання кнопки Save на екран має бути виведене повідомлення, що підтверджує успішне введення даних, потім вкладку Add product буде закрито та відкрито вкладку List of products. Новий товар має бути присутнім у списку (рисунок 2.44).

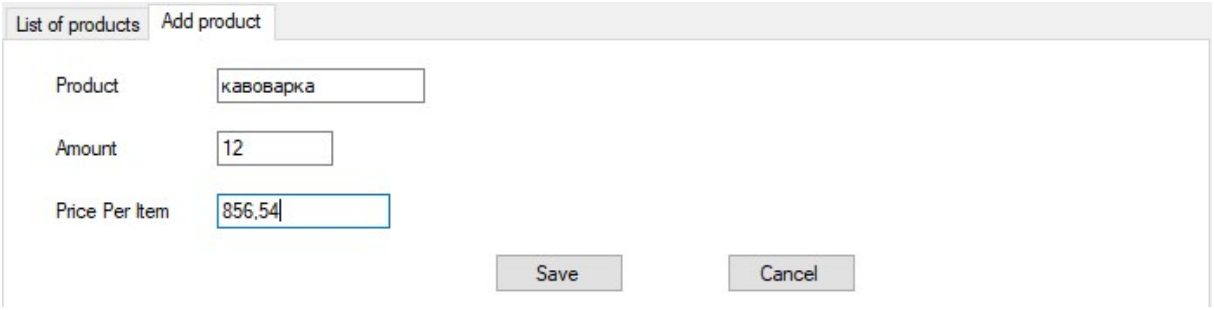


Рисунок 2.43

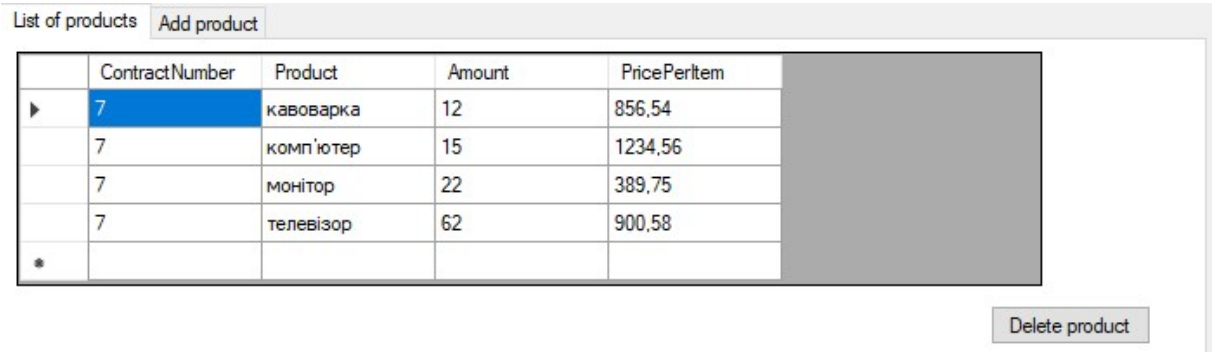


Рисунок 2.44

23. Закрити застосунок і повернутися в режим Design для Form5.cs. На вкладці Add product для об'єкта button6 (кнопка Cancel) створити функцію – обробник події Click. Текст функції наведено на рисунку 2.45. Перевірити роботу цієї кнопки (при її натисканні закривається вкладка Add product та відкривається вкладка List of products без додавання нового товару).

```
private void button6_Click(object sender, EventArgs e)
{
    this.tabControl2.SelectedTab = tabPage3;
}
```

Рисунок 2.45

24. У процесі роботи з даними про закупівлі продукції може виникнути необхідність видалення раніше введених товарів. Для цього на вкладці List of products створимо відповідну кнопку (рисунок 2.44). Для властивості Text цієї кнопки необхідно встановити значення Delete product. Для імені цього об'єкта встановити значення button7. Для об'єкта button7 створити функцію – обробник події Click. Текст функції наведено на рисунку 2.46.

```
private void button7_Click(object sender, EventArgs e)
{
    int dgvrNumber = int.Parse(dataGridView2.Rows[dataGridView2.CurrentCell.RowIndex]
        .Cells["ContractNumberDataGridViewTextBoxColumn1"].Value.ToString());
    String dgvrTovar = Convert.ToString(dataGridView2.Rows[dataGridView2.CurrentCell.RowIndex]
        .Cells["ProductDataGridViewTextBoxColumn"].Value.ToString());

    DialogResult result = MessageBox.Show("Product "+ dgvrTovar + " from Contract Number " + Convert.ToString(dgvrNumber) +
        " will be deleted. Are you sure?", "Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Warning,
        MessageBoxDefaultButton.Button2);
    switch (result)
    {
        case DialogResult.Yes:
            {
                try
                {
                    conn = new SqlConnection();
                    conn.ConnectionString = "integrated security=SSPI;data source=\".\"; " +
                              "persist security info=False; initial catalog = delivery";
                    conn.Open();
                    SqlCommand myCommand = conn.CreateCommand();
                    myCommand.CommandText = "DELETE FROM Supplied WHERE " +
                                            "ContractNumber = @dgvrNumber AND Product=@dgvrTovar";
                    myCommand.Parameters.Add("@dgvrNumber", SqlDbType.Int, 4);
                    myCommand.Parameters["@dgvrNumber"].Value = dgvrNumber;
                    (local variable) SqlCommand myCommand
                    SqlDbType.NVarChar, 20);
                    = dgvrTovar;
                    int UspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
                    if (UspeshnoeIzmenenie != 0)
                    {
                        MessageBox.Show("Changes complited", "Record modified");
                    }
                    else
                    {
                        MessageBox.Show("Changes didn't complited", "Record modified");
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
                finally
                {
                    conn.Close();
                }
                this.suppliedTableAdapter.Fill(this.deliveryDataSet.Supplied);
                break;
            }
    }
    }
}
```

Рисунок 2.46

25. Перевірити роботу кнопки Delete product. Для видалення товару потрібно вибрати товар у списку товарів, поставлених за договором (наприклад, за договором 7) та натиснути кнопку Delete product. В

результаті на екрані з'явиться вікно (рисунок 2.47), за допомогою якого потрібно підтвердити видалення товару або відмовитись від цієї операції.

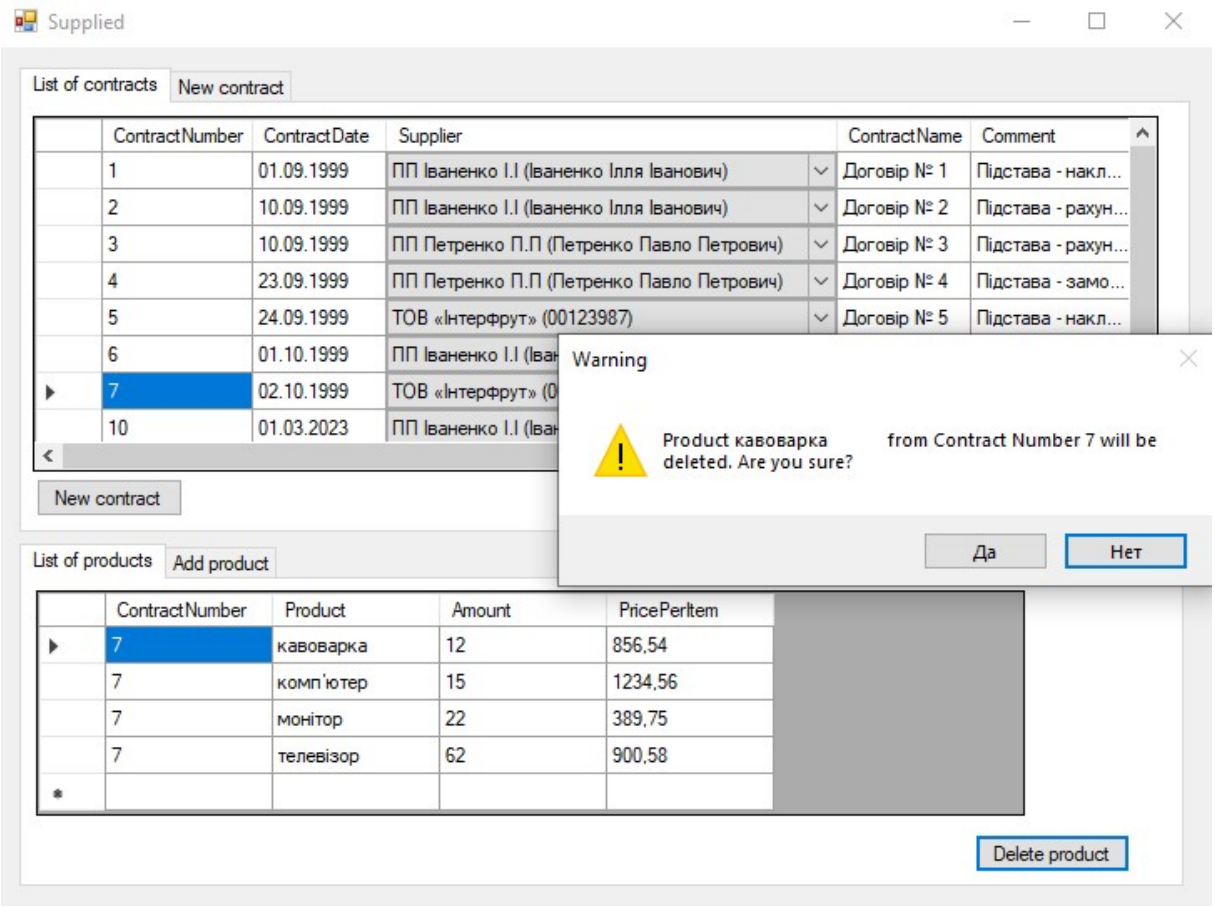


Рисунок 2.47

26. Для підвищення зручності роботи користувача з даними про закупівлі продукції необхідно, щоб користувач, вибираючи конкретний договір, бачив не лише список товарів, а й міг би бачити підсумкові дані за договором – загальну кількість одиниць закуплених товарів та загальну суму закупки. Для формування цих даних можна використовувати збережену процедуру. Таку збережену процедуру вже було створено при виконанні попередніх лабораторних робіт, але, про всяк випадок, текст запиту, за допомогою якого буде створена така збережена процедура, наведено на рисунку 2.48.

```
CREATE PROCEDURE [dbo].[sp_sum_dgvr]
    @dgvrNumber int,
    @sum_kol_vo int output,
    @sum_summa decimal(8,2) out
AS
BEGIN
    SET NOCOUNT ON
    SELECT @sum_kol_vo=SUM(Amount),
           @sum_summa= SUM(Amount*PricePerItem)
        FROM Supplied
        WHERE ContractNumber=@dgvrNumber
END
GO
```

Рисунок 2.48

27. На вкладці List of products (tabPage3) розмістити два об'єкти типу Label та два об'єкти типу TextBox. Для об'єкта типу Label для властивості Text встановити значення Total Amount та Total Cost (рисунок 2.49). Для об'єктів типу TextBox встановити імена textBox6 та textBox7 відповідно.

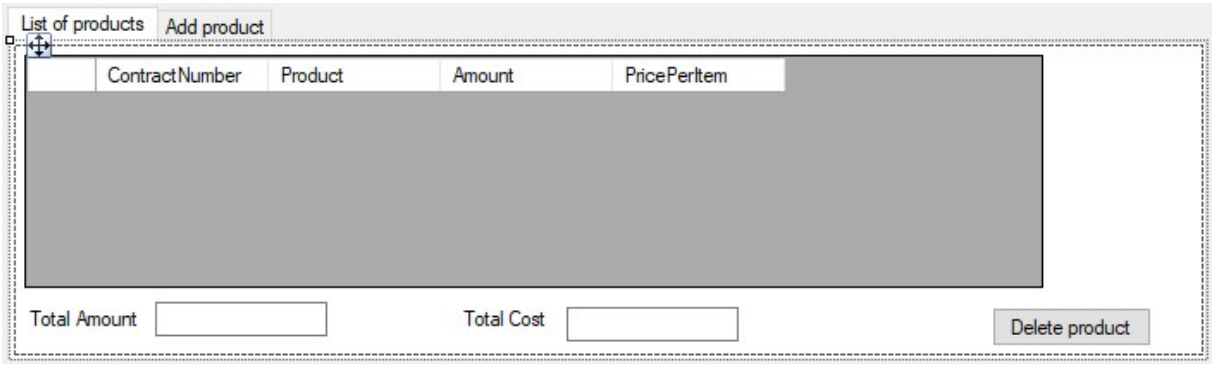


Рисунок 2.49

28. Для об'єкта dataGridView1 створити функцію – обробник події CellClick. Текст функції наведено на рисунку 2.50. За допомогою вікна Properties перевірити, що подія та функція-обробник справді пов'язані (рисунок 2.51).

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        int dgvrNumber = int.Parse(dataGridView1.Rows[dataGridView1.CurrentCell.RowIndex]
           .Cells["ContractNumberDataGridViewTextBoxColumn"].Value.ToString());
        conn = new SqlConnection();
        conn.ConnectionString = "integrated security=SSPI;data source=\".\"; " +
                                "persist security info=False; initial catalog = delivery";
        SqlCommand myCommand = conn.CreateCommand();
        myCommand.CommandType = CommandType.StoredProcedure;
        myCommand.CommandText = "[sp_sum_dgvr]";
        myCommand.Parameters.Add("@dgvrNumber", SqlDbType.Int, 4);
        myCommand.Parameters["@dgvrNumber"].Value = dgvrNumber;
        myCommand.Parameters["@dgvrNumber"].Direction = ParameterDirection.Input;
        myCommand.Parameters.Add("@sum_kol_vo", SqlDbType.Int, 4);
        myCommand.Parameters["@sum_kol_vo"].Direction = ParameterDirection.Output;
        myCommand.Parameters.Add("@sum_summa", SqlDbType.Money);
        myCommand.Parameters["@sum_summa"].Direction = ParameterDirection.Output;
        conn.Open();
        myCommand.ExecuteScalar();
        textBox6.Text = Convert.ToString(myCommand.Parameters["@sum_kol_vo"].Value);
        textBox7.Text = Convert.ToString(myCommand.Parameters["@sum_summa"].Value);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
    finally
    {
        conn.Close();
    }
}
```

Рисунок 2.50

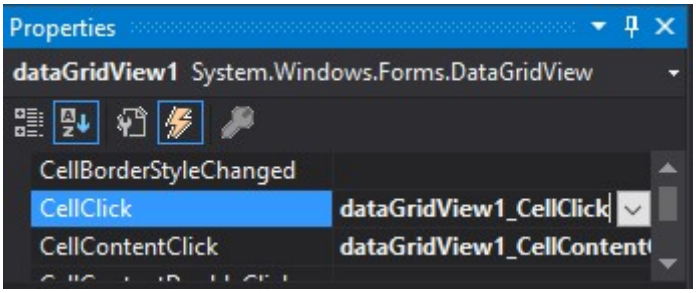


Рисунок 2.51

29. Перевірити працездатність форми. В результаті при виборі договору у списку договорів повинні розраховуватися та виводитися на екран підсумкові дані щодо договору (рисунок 2.52). Закрити застосунок та повернутися в режим Design для Form5.cs. Закрити форму Form5.cs.

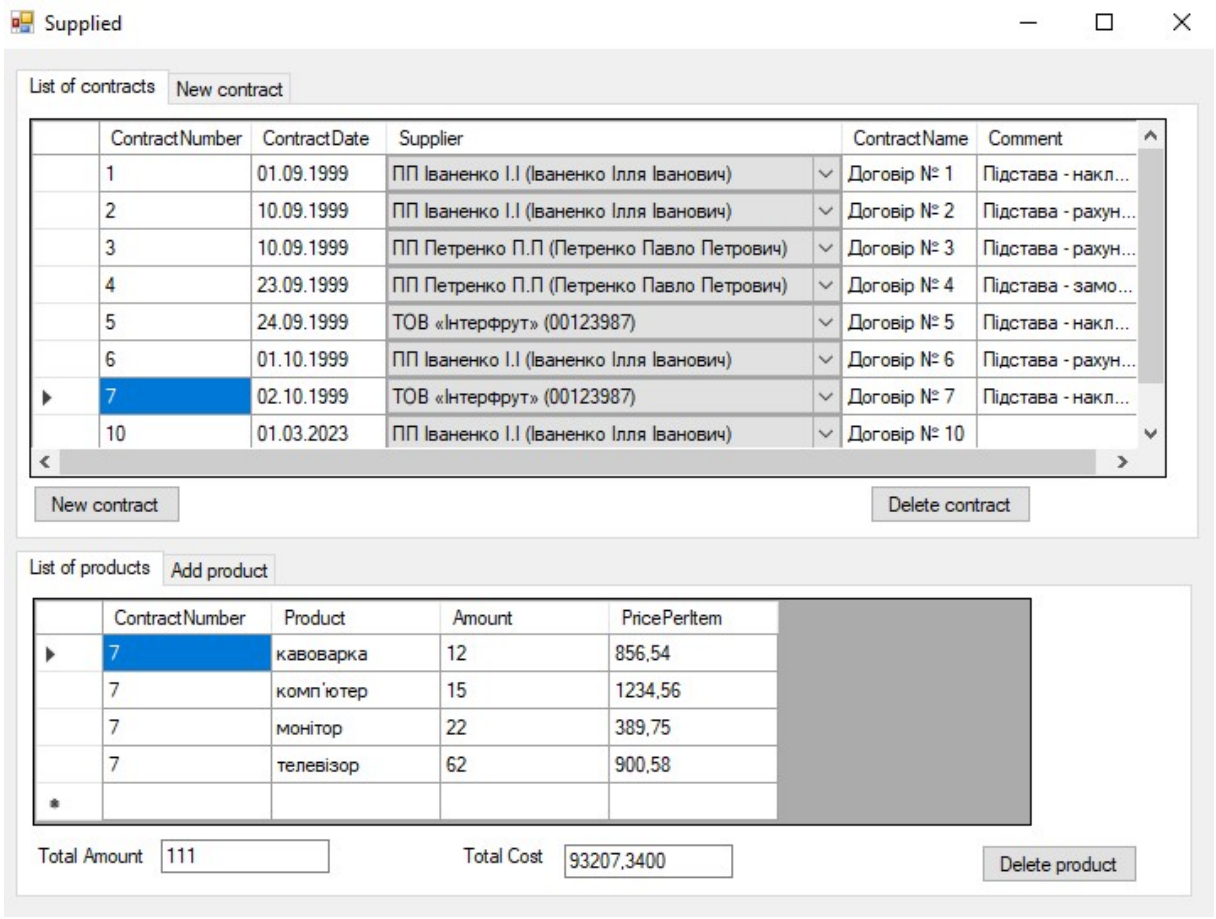


Рисунок 2.52

2.4 Створення звітної форми з узагальненими даними

Розглянемо процес створення форми, за допомогою якої кінцевий користувач буде мати можливість переглядати узагальнені дані про договори закупівлі.

Припустимо, що кінцевим користувачам треба бачити список договорів (із зазначенням номера, дати поставки та даних про постачальника), загальну кількість поставлених товарів та загальну суму за кожним договором. При формуванні даних постачальника для усіх постачальників треба вивести контактні дані, для постачальників-фізичних осіб вивести прізвище та ініціали, для постачальників-юридичних осіб – податковий номер. При цьому прізвище та ініціали або податковий номер повинні бути виведені в одному полі, тобто, або прізвище та ініціали, або податковий номер. У результат такого звіту мають бути включені лише ті

договори, на підставі яких товари дійсно поставлялися (тобто результат запиту не повинен потрапити так звані «порожні» договори).

Щоб отримати такі дані, створимо у базі даних представлення. Текст визначального запиту такого представлення може мати вигляд, наведений на рисунку 2.53. Нескладно помітити, що подібний запит вже було розроблено при виконанні попередніх лабораторних робіт. Отже, можна використати текст вже існуючого запиту. Таке представлення необхідно зробити за допомогою SQL Server Management Studio та зберегти з ім'ям View_4. Потім у вікні Data Sources потрібно клацнути правою кнопкою миші по джерелу даних deliveryDataSet і в меню вибрати пункт Configure DataSet with Wizard ... У вікні потрібно повторно відзначити пункт Views. В результаті новостворене представлення має з'явитися у списку об'єктів джерела даних. Змінене джерело даних треба зберегти.

```
SELECT Contracts.ContractNumber, ContractDate, SUM(Amount) AS кількість, SUM(Amount*PricePerItem) AS сума,
    CAST(Note AS CHAR(40)) AS КонтактиПостачальника,
    ISNULL(LegalEntities.TaxNumber, RTRIM(LastName)+' '+SUBSTRING(FirstName,1,1)+'.'++SUBSTRING(SecondName,1,1)+'.')
        AS Постачальник
FROM ((Suppliers LEFT JOIN IndividualEntrepreneurs ON Suppliers.SupplierID=IndividualEntrepreneurs.SupplierID)
    LEFT JOIN LegalEntities ON Suppliers.SupplierID=LegalEntities.SupplierID)
        INNER JOIN Contracts ON Suppliers.SupplierID=Contracts.SupplierID
    INNER JOIN Supplied ON Supplied.ContractNumber=Contracts.ContractNumber
GROUP BY Contracts.ContractNumber, ContractDate, CAST(Note AS CHAR(40)),
ISNULL(LegalEntities.TaxNumber, RTRIM(LastName)+' '+SUBSTRING(FirstName,1,1)+'.'++SUBSTRING(SecondName,1,1)+'.')
```

Рисунок 2.53

Для підключення такого представлення до застосунку треба виконати наступну послідовність дій.

1. Створити нову форму, наприклад, з ім'ям Form6.cs та додати її до проекту. Для властивості Text форми встановити значення Aggregate data of contracts.

2. В результаті до проекту буде додано нову форму. Вибравши у списку об'єктів джерела даних представлення View_4, потрібно за допомогою миші перетягнути її на форму. В результаті на формі з'являться об'єкти типу DataGridView та BindingNavigator (рисунок 2.54), за

допомогою яких можна переглянути дані які будуть сформовані за допомогою представлення. Форму Form6.cs можна зберегти та закрити.

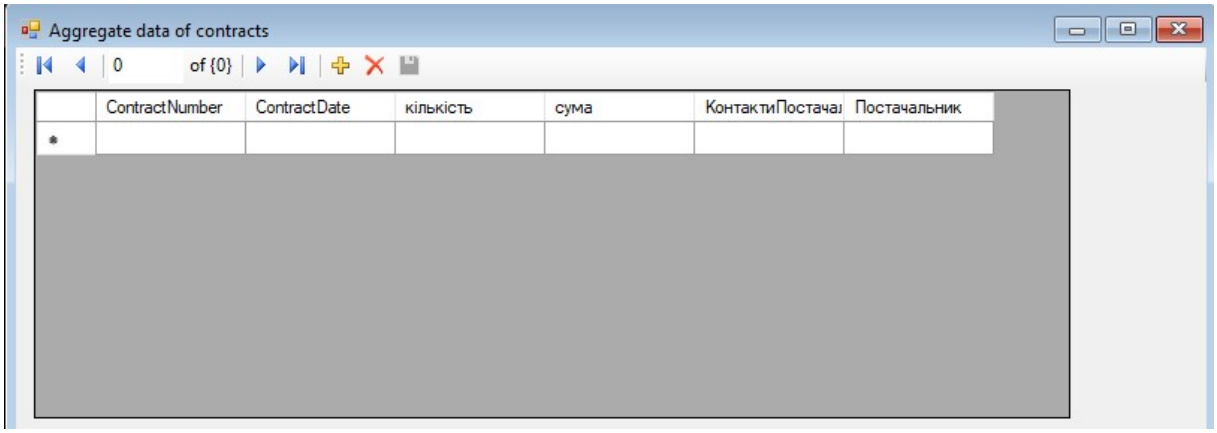


Рисунок 2.54

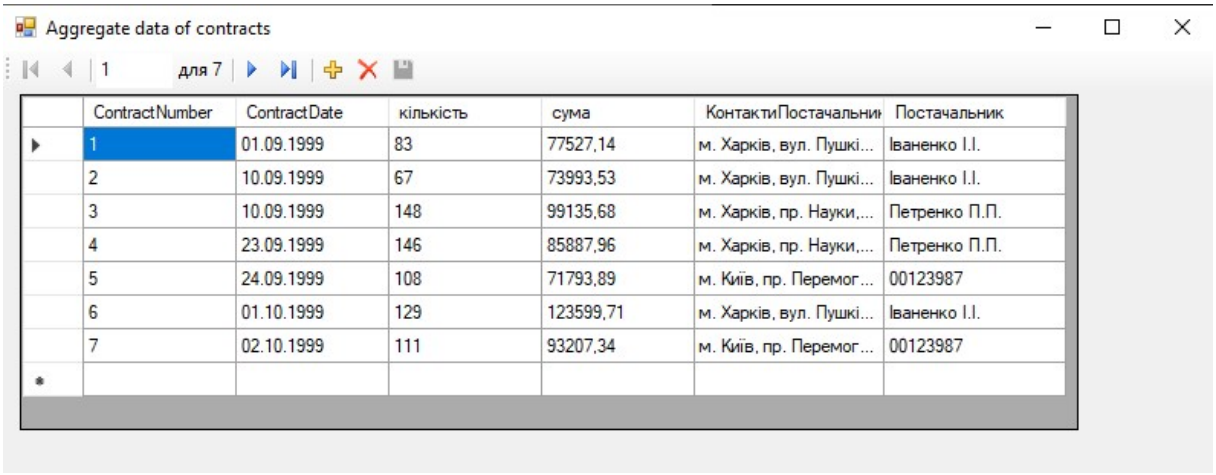
3. Тепер створену форму потрібно підключити до спільного меню програми. Для цього потрібно перейти на роботу з головною формою (Form1) (в режимі Design) і для пункту головного меню Reports створити підменю. Перший пункт цього підменю слід назвати Aggregate data of contracts (рисунок 2.55). Подвійне натискання миші на цьому пункті меню відкриє вікно коду, в якому можна ввести текст функції, що забезпечує відкриття форми при виборі цього пункту меню (рисунок 2.56).

Рисунок 2.55

```
private void toolStripMenuItem2_Click(object sender, EventArgs e)
{
    Form6 frm = new Form6();
    frm.Show();
}
```

Рисунок 2.56

4. Перевірити працездатність форми. В результаті при виборі відповідного пункту меню кінцевий користувач буде мати можливість переглядати узагальнені дані про договори закупівлі (рисунок 2.57).



	ContractNumber	ContractDate	кількість	сума	КонтактиПостачальни	Постачальник
▶	1	01.09.1999	83	77527,14	м. Харків, вул. Пушкі...	Іваненко І.І.
	2	10.09.1999	67	73993,53	м. Харків, вул. Пушкі...	Іваненко І.І.
	3	10.09.1999	148	99135,68	м. Харків, пр. Науки,...	Петренко П.П.
	4	23.09.1999	146	85887,96	м. Харків, пр. Науки,...	Петренко П.П.
	5	24.09.1999	108	71793,89	м. Київ, пр. Перемог...	00123987
	6	01.10.1999	129	123599,71	м. Харків, вул. Пушкі...	Іваненко І.І.
	7	02.10.1999	111	93207,34	м. Київ, пр. Перемог...	00123987
*						

Рисунок 2.57

2.5 Збереження результатів роботи

Зберегти папку проєкту і всі файли, що знаходяться в ній. Назву папки (наприклад, \WindowsFormsApplication1) слід подивитися на своєму робочому комп’ютері.

3 ВИМОГИ ДО ЗВІТУ

Відповіднім чином оформлений та роздрукований звіт з лабораторної роботи є документом, що підтверджує виконання студентом лабораторної роботи.

У звіті з лабораторної роботи:

1) стисло описати основні етапи виконання завдання;
2) описати створений клієнтський застосунок та особливості роботи з ним;
3) зробити висновки за результатами виконання лабораторної роботи.

Звіт з лабораторної роботи роздруковується на аркушах формату А4, він повинен мати відповідній титульний аркуш. Роздрукованій звіт здається студентом викладачу у файлі.

Звіт має буті оформлень за такими вимогами:

– параметри сторінки: лівий відступ – 3 см; правий – 1,5 см; верхній та нижній відступи по 2 см;

– шрифт Times New Roman, 14;

– налаштування абзацу: вирівнювання – за шириною, відступи зліва та справа – 0 см, відступ першого рядка - 1,25 см, інтервал перед та після абзацу – 0 пт, міжрядковій інтервал – одинарний; на вкладці «Положення на сторінці» відключити функцію «Заборона висячих рядків».

Усі скріншоти розміщені у звіті, є рисунками, отже повинні мати підписи та відповідну нумерацію.

Ознакою того, що студент не тільки виконав лабораторну роботу, але й здав її (тобто підтвердив наявність відповідних знань та навичок), є наявність на титульному аркуші підпису викладача та дати здачі.

4 ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

1. Типи користувачів автоматизованої обчислювальної системи.
2. Хто такі кінцеві користувачі?
3. Що таке інтерфейс користувача?
4. Складові інтерфейсу користувача
5. Що таке графічний інтерфейс користувача?
6. Характеристики графічного інтерфейсу користувача
7. Принципи побудови «дружнього» інтерфейсу користувача
8. Правила проектування інтерфейсу користувача. Загальна характеристика.
9. Що значить "дати контроль користувачу"?
10. Що є основними положеннями надання контролю користувачу над інтерфейсом?
11. Що значить " безрежимність "?
12. Що значить " гнучкість "?
13. Що значить " можливість переривання "?
14. Що значить " корисність "?
15. Що значить " поблажливість "?
16. Що значить " можливість орієнтування "?
17. Що значить " доступність "?
18. Що значить " спрощення користування "?
19. Що значить " пристосовуваність "?
20. Що значить " інтерактивність "?
21. Що значить "зменшити навантаження на пам’ять користувача"?
22. Що є основними положеннями зменшення навантаження на пам’ять користувача?
23. Що значить " запам'ятовування "?
24. Що значить " розпізнавання "?
25. Що значить " інформування "?
26. Що значить " терпимість "?

27. Що значить " швидкість "?

28. Що значить " інтуїтивність "?

29. Що значить " перенос "?

30. Що значить " контекст "?

31. Що значить " організація "?

32. Що значить "зробити інтерфейс сумісним"?

33. Що є основними положеннями розроблення сумісного інтерфейсу?

34. Що значить " послідовність інтерфейсу "?

35. Що значить " досвід "?

36. Що значить " прогнозування "?

37. Що значить " відношення "?

38. Що значить " передбачуваність "?

39. Як у середовищі Microsoft Visual Studio створити проєкт Windows Forms Application?

40. Як додати меню у форму в проєкті Windows Forms Application?

41. Як додати форму в проєкт Windows Forms Application?

42. Як створити нове джерело даних в проєкті Windows Forms Application?

43. Об’єкт DataGridView, його призначення та особливості застосування як складової інтерфейсу в проєкті Windows Forms Application

44. Як за допомогою об’єктів DataGridView відобразити звязані дані в формі проєкту Windows Forms Application?

СПИСОК ЛІТЕРАТУРИ

1. Програмування баз даних. Практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка» / М. В. Добролюбова, М. В. Філіппова, О. М. Маркіна. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 164 с.

2. Anthony DeBarros. Practical SQL. 2nd Edition. – No Starch Press, Inc., 2022. – 440 c.

3. Jeffrey A. Hoffer, V. Ramesh, Heikki Topi. Modern Database Management. Thirteenth Edition. – Pearson Education Limited, 2020. – 591 p.

4. Dušan Petković. Microsoft SQL Server 2019. A Beginner’s Guide. Seventh Edition. – McGraw-Hill Education, 2020. – 865 p.

5. Mukesh Negi. Fundamentals of Database Management System: Learn essential concepts of database systems. – BPB Publications, 2019. – 175 p.

6. Edward Sciore. Database Design and Implementation: Second Edition. – Springer Nature, 2020. – 468 p.

7. Gavin Powell. Database Modeling Step by Step. – CRC Press, 2020. – 268 p.

8. Sanjiv Purba. Handbook of Data Management: 1999 Edition. – CRC Press, 2019. – 1101 p.

9. C. J. Date. Database Design and Relational Theory: Normal Forms and All That Jazz. – Apress, 2019. – 451 p.

10. Jonathan Eckstein, Bonnie R. Schultz. Introductory Relational Database Design for Business, with Microsoft Access. – John Wiley & Sons, 2018. – 328 p.

11. Alan Beaulieu. Learning SQL: Generate, Manipulate, and Retrieve Data. – O'Reilly Media, Inc., 2020. – 384 p.

12. M. Tamer Özsu, Patrick Valduriez. Principles of Distributed Database Systems. – Springer Nature, 2019. – 674 p.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи за темою «Розробка засобами Microsoft Visual Studio прикладного програмного забезпечення для роботи з базою даних Microsoft SQL Server»
для студентів спеціальностей
121 «Інженерія програмного забезпечення»
122 «Комп'ютерні науки»
126 «Інформаційні системи та технології»

Укладачі:
ОРЛОВСЬКИЙ Дмитро Леонідович
КОПП Андрій Михайлович

Відповідальний за випуск Годлевський М.Д.
Роботу до видання рекомендував Гамаюн І.П.

План 2023 р., поз. 499

Підп. до друку __.__.__. Гарнітура Times New Roman.
Ум. друк. арк. 2,9.

Видавничий центр НТУ «ХПІ»,
вул. Кирпичова, 2, м. Харків, 61002
Свідоцтво про державну реєстрацію ДК № 3478 від 21.08.2017 р.

Самостійне електронне видання