МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна
Бахмутський навчально-науковий професійно-педагогічний інститут
Кафедра електромеханічних та комп'ютерних систем

До захисту допущено
Завідувач кафедри
_____ Інна НЕФЬОДОВА
(підпис) (ім'я, прізвище)
«07» грудня 2024 року

КВАЛІФІКАЦІЙНА РОБОТА (ПРОЄКТ)

рівень вищої освіти другий (магістерський)

спеціальність 015.39 Професійна освіта (Цифрові технології)

освітньо-професійна програма Професійна освіта. Комп'ютерні технології в управлінні та навчанні

тема «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти»

Виконав(ла)

здобувач(ка) групи БД-К23мг
(шифр групи)

Валерій ПАВЛОВСЬКИЙ
(ім'я, прізвище) _____ (підпис)

Керівник роботи
к.т.н., доц. Павло ЧИКУНОВ
(науковий ступінь, вчене звання, ім'я, прізвище) _____ (підпис)

Рецензент роботи
к.пед.н., доц. Наталія ЛОГІНОВА
(науковий ступінь, вчене звання, ім'я, прізвище) _____ (підпис)

Консультант
к.пед.н., доц. Юлія БОБРИКОВА
(науковий ступінь, вчене звання, ім'я, прізвище) _____ (підпис)

Засвідчую, що у цій роботі немає цитат та вилучень з праць інших авторів без відповідних посилань
здобувач (ка) _____ (підпис)

Харків – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет/ННІ Бахмутський навчально-науковий професійно-педагогічний інститут
Кафедра Електромеханічних та комп'ютерних систем
Рівень вищої освіти другий (магістерський)
Спеціальність 015.39 Професійна освіта (Цифрові технології)
Освітньо-професійна програма Професійна освіта. Комп'ютерні технології в управлінні та навчанні

ЗАТВЕРДЖУЮ

Завідувач кафедри
(підпис) Інна НЕФЬОДОВА (ім'я, прізвище)

«08» жовтня 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Павловський Валерій Валерійович
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи Професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти
керівник роботи Чикунов Павло Олександрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «08» жовтня 2024 року № 5101-5/3236

2. Строк подання здобувачем роботи «02» грудня 2024 р.

3. Перелік питань, які потрібно розробити: Актуальність професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти. Характеристика об'єктів галузі: стан і стратегії розвитку. Вимоги до кадрового забезпечення об'єкту галузі. Методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

4. План роботи

№ з/п	Назви етапів роботи
1	Огляд літературних джерел, нових розробок, опублікованих даних та іншої інформації, пов'язаної з темою роботи.
2	Дослідження теоретичних підходів до актуальності професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти».
3	Характеристика об'єктів галузі: стан і стратегії розвитку.
4	Розробка методики професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.
5	Розробка вимог до кадрового забезпечення об'єкту галузі
6	Оформлення першого варіанту тексту, подання його на ознайомлення науковому керівнику
7	Усунення недоліків, написання остаточного варіанту тексту, оформлення дипломної роботи
8	Подання роботи на кафедру, перевірка на плагіат та зовнішнє рецензування роботи
9	Захист дипломної роботи у ЕК

5. Дата видачі завдання «08» жовтня 2024 р.

Здобувач(ка) _____ Валерій ПАВЛОВСЬКИЙ
(підпис) (ім'я, прізвище)

Керівник роботи _____ Павло ЧИКУНОВ
(підпис) (ім'я, прізвище)

РЕФЕРАТ

Мета дослідження – теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

Об'єктом дослідження роботи є процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET».

Предметом дослідження роботи є методика професійної підготовки фахівців з цифрових технологій до розробки комплексу освітніх ресурсів.

Охарактеризовано систему професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

Виконана постановка 4 нових лабораторних робіт, присвячених реалізації графічного інтерфейсу користувача.

Виконано аналіз вимог до кадрового забезпечення об'єкту ІТ-галузі.

Розроблено дидактичний проект консультативного заняття.

За основними результатами дослідження виконана публікація тези доповіді на VII Всеукраїнської науково-практичної інтернет-конференції «Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні» (м. Харків, 05-06 грудня 2024 р.).

Обсяг дипломної роботи становить: пояснювальна записка, презентація доповіді. Пояснювальна записка складається з вступу, чотирьох розділів, висновків, списку використаних джерел, додатків. Загальний обсяг роботи 81 сторінок, з яких 55 сторінок основного тексту. Список використаних джерел становить 52 найменувань, 6 таблиць, 33 рисунків.

ПРОФЕСІЙНА ПІДГОТОВКА, ПРАКТИКУМ, ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА, ПЛАТФОРМА .NET, КАДРОВЕ ЗАБЕЗПЕЧЕННЯ, МЕТОДИЧНА РОЗРОБКА

ABSTRACT

Research Objective – to theoretically substantiate and partially test the methodology of professional training for digital technology specialists in teaching the educational module "Implementation of Graphical User Interface on the .NET Platform" at higher education institutions.

Object of the Research: the process of professional training of digital technology specialists for teaching the educational module "Implementation of Graphical User Interface on the .NET Platform."

Subject of the Research: the methodology of professional training for digital technology specialists to develop a set of educational resources.

The system of professional training of digital technology specialists for teaching the educational module "Implementation of Graphical User Interface on the .NET Platform" at higher education institutions has been characterized.

Four new laboratory works dedicated to the implementation of graphical user interfaces have been developed.

An analysis of the requirements for staffing in the IT sector has been conducted.

A didactic project for a consultative session has been developed.

The main results of the study have been published as a thesis report at the VII All-Ukrainian Scientific and Practical Online Conference "Modern Technologies in Energy, Electromechanics, Control Systems, and Mechanical Engineering" (Kharkiv, December 5–6, 2024).

The diploma work includes an explanatory note and a presentation of the report. The explanatory note consists of an introduction, four chapters, conclusions, a list of references, and appendices. The total volume of the work is 81 pages, including 55 pages of the main text. The list of references includes 52 sources, 6 tables, and 33 figures.

PROFESSIONAL TRAINING, PRACTICAL, GRAPHICAL USER INTERFACE, .NET, STAFFING, METHODOLOGICAL DEVELOPMENT.

ЗМІСТ

ВСТУП

У контексті розбудови української державності постає нагальна потреба у створенні нової системи освіти, що передбачає визначення ключових теоретичних і методичних основ її впровадження, зокрема у сфері вищої освіти. Особливого значення набуває питання реформування мети, конкретизації завдань, оновлення змісту та вдосконалення технологій професійної підготовки фахівців у галузі цифрових технологій. Зокрема, це стосується викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET», який є важливим напрямом інтеграції у закладах вищої освіти.

У зв’язку з цим виникає необхідність модернізації державної політики у сфері вищої освіти. Це має сприяти підвищенню якості професійної підготовки фахівців з цифрових технологій, їх конкурентоспроможності, а також розробці сучасного змісту освіти, ефективних форм, методів і засобів навчання. Важливим аспектом є вдосконалення освітніх технологій на основі компетентнісного підходу.

Як виявлено на основі аналізу наукової літератури, ученими досліджено окремі аспекти порушеної проблеми. Науково-методологічним аспектам освітньої сфери присвячені роботи вчених О. Антонюка, В. Бакуменка, О. Батанова , В. Гриценко, Ю. Журавльової, А. Кобця, В. Коврегіна, В. Кременя, В. Лугового, В. Мороза , В. Огаренка та ін.; різні аспекти професійної підготовки майбутніх інженерів-педагогів досліджують Е. Абільтарова, І. Васильєва, Н. Волкова, Н. Брюханова, Р. Горбатюк, О. Коваленко, М. Лазарєв, В. Кулешова, В. Мальована, С. Хоменко, Л. Штефан та ін.

Необхідність дослідження й вирішення проблеми та вивчення педагогічного досвіду показали зумовлені наявні суперечності між:

– зростаючими потребами сфери послуг у професійних кадрах, які можуть швидко адаптуватись та змінюватись в умовах організації і технології надання послуг, ефективно здійснювати професійну діяльність,

вдосконалювати професійні навички шляхом постійного саморозвитку, і недостатнім рівнем сформованості професійної компетентності майбутніх фахівців;

– потребою осучаснення та підвищення рівня якості професійної підготовки майбутніх фахівців відповідно до оновлених стандартів з урахуванням необхідності формування навичок адаптивності та недостатньою розробленістю теоретичних основ й педагогічних умов їхньої професійної підготовки.

Необхідність подолання виявлених суперечностей й обумовила тему дослідження «Професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти».

Мета дослідження – теоретично обґрунтувати та частково перевірити методику професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

У зв'язку з поставленою метою, в роботі вирішуються такі завдання.

1. Проаналізувати та визначити ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

2. Обґрунтувати теоретичну і методичну базу концепції методичної системи професійної підготовки фахівців з цифрових технологій.

Об'єкт дослідження: процес професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

Предмет дослідження: методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

В ході написання роботи використані такі методи дослідження:

– теоретичні: аналіз наукової літератури для порівняння, з метою визначення психолого-педагогічних підходів, зіставлення різних поглядів на досліджувану проблему;

– емпіричні методи (анкетування, бесіди з учасниками експерименту, педагогічне спостереження, самооцінювання, тестування) потрібні для визначення рівнів сформованості компетентності майбутніх фахівців з цифрових технологій.

Наукова новизна одержаних результатів дослідження полягає в теоретичному обґрунтуванні структури професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти та в уточненні сутності поняття «професійна підготовка» фахівців з цифрових технологій.

Подальшого розвитку набули зміст навчання майбутніх фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти для формування самоосвітньої компетентності.

Теоретичне та практичне значення одержаних результатів полягає в розробці методики професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти, обґрунтуванні змісту, форм та методів організації процесу підготовки фахівців.

Матеріали дослідження використовувались при підготовці здобувачів вищої освіти Навчально-наукового професійно-педагогічного інституту УІПА (м. Бахмут) зі спеціальності 015 Професійна освіта (Цифрові технології).

За основними результатами дослідження виконана доповідь на VII Всеукраїнській науково-практичній інтернет-конференції «Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні» (м. Харків, 05-06 грудня 2024 р.).

РОЗДІЛ 1 АКТУАЛЬНІСТЬ ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «РЕАЛІЗАЦІЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА НА ПЛАТФОРМІ .NET» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ

Необхідність цілеспрямованого формування та розвитку ключових компетентностей для забезпечення навчання впродовж життя, особливо в умовах реформування системи освіти в Україні, включаючи вищу освіту, викликає необхідність перегляду підходів до професійної підготовки фахівців у сфері цифрових технологій.

Стрімка модернізація й оновлення цифрових та інформаційних технологій, а також зростання обсягу інформаційних потоків вимагають від сучасного фахівця високого рівня цифрової компетентності. Це, своєю чергою, передбачає впевнене володіння цифровими інструментами для навчання та професійної діяльності. Сучасний ринок праці вимагає дедалі більшої кількості фахівців, здатних ефективно використовувати інформаційно-комунікаційні та цифрові технології у своїй роботі, і ця потреба буде лише зростати.

У цьому контексті цифровізація освіти набуває особливого значення, оскільки вона спрямована на підготовку молоді до життя у цифровому суспільстві та до професій майбутнього. Пріоритетними завданнями є формування цифрових навичок і компетентностей, що стане основою для прискорення розвитку цифрової економіки.

Основними напрямами цифровізації освіти є:

– створення сучасних освітніх ресурсів і цифрових платформ, які містять інтерактивний та мультимедійний контент і є доступними для всіх закладів вищої освіти та здобувачів освіти;

– розроблення та впровадження інноваційних засобів навчання, таких як комп’ютерні, мультимедійні й комп’ютерно орієнтовані технології, а також

створення цифрового навчального середовища;

– забезпечення широкосмугового доступу до Інтернету для здобувачів освіти в закладах вищої освіти.

Особливу актуальність має підготовка фахівців із цифрових технологій для викладання сучасних дисциплін, таких як «Реалізація графічного інтерфейсу користувача на платформі .NET». Такі спеціалісти повинні не лише професійно виконувати свою роботу та володіти трудовими, творчими та управлінськими навичками, а й демонструвати здатність до швидкої адаптації до змінюваних вимог роботодавців. Вони мають бути готовими оперативно вирішувати проблеми в умовах мінливої ринкової економіки, проявляти гнучкість та інноваційність у професійній діяльності, що є важливою умовою успішної інтеграції у цифрове середовище сучасного світу [47].

Сучасна освітня концепція України орієнтована на забезпечення безперервного навчання протягом життя. У зв’язку з цим ключовими компетентностями вважаються здатність до саморозвитку, самоосвіти та мотивація до постійного вдосконалення, що є основою для формування успішної особистості.

Підготовка фахівців із цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти спрямована на надання молоді необхідних навичок і якостей. Головною метою є формування сучасного, конкурентоспроможного професіонала, який володіє мотивацією до саморозвитку, прагненням до самореалізації та готовністю відповідати викликам сучасного ринку праці [41].

Сучасний фахівець у галузі цифрових технологій виконує широкий спектр завдань, які відображають багатогранність його професійної діяльності. Ефективне виконання цих завдань значною мірою залежить від рівня професійної підготовки у закладах вищої освіти, що базується на принципах компетентнісного підходу.

Організація освітнього процесу на основі компетентнісного підходу

орієнтована на досягнення конкретних результатів навчання. Вона сприяє формуванню висококваліфікованих фахівців, здатних до творчої трансформаційної діяльності, розроблення різнопланового навчального й наукового контенту, а також до самоаналізу та постійного саморозвитку.

Однією з базових компетентностей, необхідних для навчання впродовж життя, є цифрова компетентність. Вона передбачає розвиток логічного мислення, здатності ефективно управляти інформацією в цифровому середовищі та використовувати цифрові інструменти як у професійній діяльності, так і в повсякденному житті.

Питання формування та розвитку компетентностей різного рівня розглядали у своїх дослідженнях такі науковці, як С. Алєксєєва, Н. Арістова, В. Гриньова, М. Гриньова, Я. Кодлюк, О. Лаврентьєва, О. Малихін, І. Осадченко, С. Сисоєва, О. Топузов та В. Чайка.

Грунтовною основою вивчення та розв’язання проблеми дослідження теоретичних та методичних засад підготовки фахівців з цифрових та інформаційних технологій стали результати напрацювань відомих науковців з різних напрямів освіти: педагогіки професійної освіти (В. Безрукова, Р. Гуревич, О. Дубасенюк, О. Дубинчук, Н. Кузьміна, Л. Лук’янова, В. Мадзігон, Н. Ничкало, Л. Оршанський, Л. Сидорчук, В. Тименко, А. Цина); структурування знань у змісті освіти (Б. Гершунський, В. Гінецинський, В. Лєдньов, О. Щербак, та ін.); інтеграції технологій в освітній процес (О. Білик, М. Корець, Д. Корчевський, М. Піддячий та ін.); застосування цифрових технологій в освіті (О. Авраменко, В. Биков, Т. Бодненко, Т. Вакалюк, І. Войтович, А. Гедзик, Ю. Горошко, А. Гуржій, М. Жалдак, Л. Карташова, В. Лапінський, Л. Макаренко, Н. Морзе, Ю. Рамський, С. Семеріков, О. Спірін, Г. Ткачук, Ю. Триус, В. Франчук, С. Яшанов та ін.); різні аспекти професійної підготовки майбутніх інженерів-педагогів (Е. Абільтарова, І. Васильєва, Н. Волкова, Н. Брюханова, Р. Горбатюк, О. Коваленко, В. Кулешова, В. Мальована, М. Лазарєв, С. Хоменко, Л. Штефан та ін.); процес формування комунікативної компетентності та її складових у здобувачів освіти закладів

інженерної та інженерно-педагогічної освіти (Т. Калініченко, К. Ковальова, В. Кручек та ін.).

Аналіз наукових досліджень, присвячених означеній проблематиці, виявив ключове протиріччя в системі інженерно-педагогічної освіти. Воно полягає у необхідності комплексної підготовки майбутніх фахівців із цифрових технологій для закладів вищої освіти, що включає якісну інженерну, психолого-педагогічну та робітничу підготовку [43-44]. Зважаючи на те, що основна діяльність таких фахівців безпосередньо пов'язана з педагогічним процесом підготовки кваліфікованих кадрів у галузі цифрових технологій, інженерно-педагогічну освіту варто розглядати як педагогічну за своєю сутністю, водночас враховуючи важливість інженерних знань та навичок для її успішної реалізації.

Викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти має бути спрямоване на розвиток у здобувачів умінь самоосвіти, що забезпечить їх здатність адаптуватися до динамічних змін у сфері цифрових технологій та організовувати свою професійну діяльність на високому рівні.

Особливого значення в підготовці майбутніх фахівців із цифрових технологій набуває не лише формування професійних компетентностей, а й розвиток особистісних якостей, таких як адаптивність, креативність, відповідальність і вміння працювати в команді [36, 40].

Завдання вдосконалення професійної підготовки фахівців із цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти полягає у двох аспектах. З одного боку, це коригування та розширення професійної освіти фахівців, а з іншого – розвиток економічно стабільної системи вищої освіти, оснащеної сучасною методологією і здатної реалізовувати гнучкі форми навчання. Усе це підкреслює актуальність проблеми професійної підготовки фахівців у сфері цифрових технологій.

Система професійної підготовки має орієнтуватися, по-перше, на

потреби особистості, зацікавленої у виявленні та розвитку власного потенціалу. По-друге, нові види професійної діяльності у сфері цифрових технологій гарантують затребуваність фахівців на ринку праці, враховуючи розширення професійного поля діяльності та підвищення їх мобільності. Це сприятиме не лише конкурентоспроможності випускників, але й забезпеченню ефективної інтеграції цифрових технологій у сучасний освітній і професійний простір.

Висновки до розділу

У кваліфікаційній роботі відповідно до мети і завдань розкрито стан наукової проблеми. У ході дослідження уточнено та систематизовано основні поняття, що характеризують сутність та структуру процесу професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти та виділено пріоритетні напрямки удосконалення професійних компетентностей .

З’ясовано, що професійна підготовка фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

Охарактеризовано систему професійної підготовки фахівців з цифрових технологій для викладання освітнього модуля «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти.

РОЗДІЛ 2 ХАРАКТЕРИСТИКА ОБ'ЄКТІВ ГАЛУЗІ: СТАН І СТРАТЕГІЇ РОЗВИТКУ

2.1 Огляд засобів розробки застосунків з графічним інтерфейсом

Платформа .NET призначена для розробки різноманітних кросплатформних застосунків та підтримує створення програм (застосунків) для Windows, macOS, Linux, Android, iOS та інших систем [1]. Платформа .NET підтримує різні типи застосунків:

– десктопні застосунки з графічним інтерфейсом користувача,

– вебдодатки та мобільні застосунки,

– хмарні застосунки,

– інтернет речей та штучний інтелект.

Таким чином, .NET є універсальною платформою, що забезпечує інструменти для розробки сучасних застосунків на різних пристроях та операційних системах.

З роками розвиток технологій і вимог призвело до появи різних способів написання додатків для Windows. Мета залишається незмінною: допомогти розробникам у створенні інтерфейсу користувача і базового стандартного коду, які потім можна доповнювати унікальними функціями. Саме для цього корпорація Майкрософт створила багато засобів і бібліотек.

GUI (Graphical User Interface) Windows – це графічний інтерфейс користувача у Windows, який дозволяє користувачу взаємодіяти з ОС Windows або додатками на основі графічних елементів, таких як вікна, кнопки, іконки та меню, замість текстових команд [2]. У GUI Windows користувач може використовувати мишу для навігації, вибору елементів і виконання дій, що робить взаємодію інтуїтивнішою та зручнішою.

Головними компонентам GUI в Windows є робочий стіл, вікна, панель завдань, меню, іконки. GUI у Windows суттєво спрощує виконання повсякденних завдань, дозволяючи користувачам відкривати файли, запускати

програми, налаштовувати параметри системи та багато іншого через зручні для сприйняття візуальні елементи.

Розробка програмного забезпечення для Windows має давню історію, яка зазнала значних змін та покращень протягом багатьох років. Три основні платформи, які наразі використовуються для створення програм Windows – це Windows Forms, Windows Presentation Foundation та Universal Windows Platform. Кожна з них має свої переваги, недоліки та області застосування, і розробники часто вибирають платформу залежно від специфіки проекту.

Windows Forms – це графічний інтерфейс користувача з відкритим кодом для .NET, стандартний набір бібліотек базових класів і API, що спрощують поширені завдання програми [3]. За допомогою такого середовища розробки, як Visual Studio, можна створювати інтелектуальні клієнтські програми Windows Forms, які відображають інформацію, вимагають введення від користувачів та обмінюються даними з віддаленими комп'ютерами.

Windows Forms була однією з перших графічних бібліотек для розробки застосунків на Windows з .NET Framework. Ця технологія дозволяє розробникам створювати додатки з віконним інтерфейсом, використовуючи простий та інтуїтивно зрозумілий підхід. WinForms характеризується легкістю освоєння та використання. Інтерфейси в ній створюються через вибір і розташування контролів на формі, що значно полегшує процес розробки.

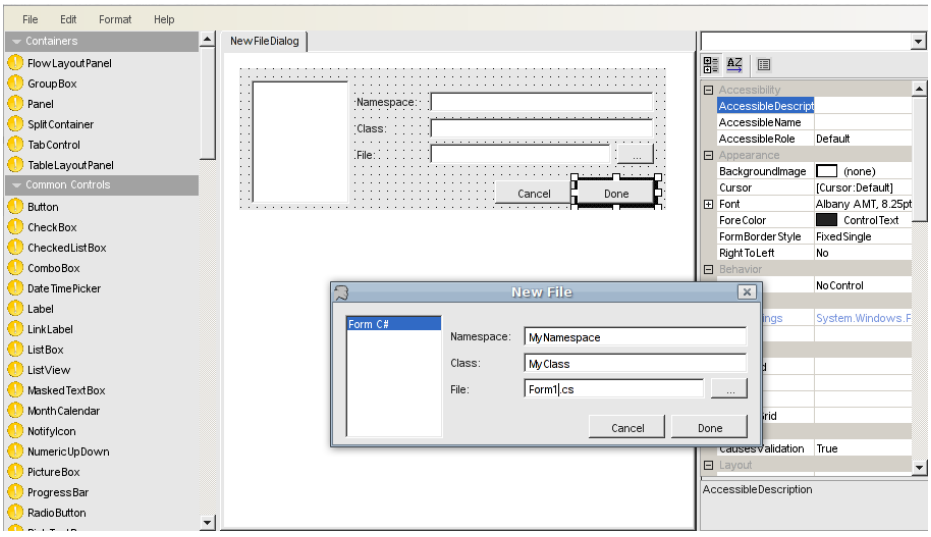

Рисунок 2.1 – Приклад візуальної розробки GUI у Windows Forms

Однак WinForms має свої недоліки. Вона застаріла порівняно з більш сучасними технологіями, її функціонал обмежений при створенні складних, насичених графічними елементами інтерфейсів. WinForms працює лише в межах Windows-середовища, що зменшує її привабливість для сучасних крос-платформних проектів. Тим не менш, вона залишається популярною в рішеннях, де потрібна швидка розробка і підтримка старих систем.

Компанія Microsoft у 2015 році разом з ОС Windows 10 представила Universal Windows Platform (UWP), яка надає загальну платформу програм на кожному пристрої під керуванням Windows [4]. Основні API UWP однакові на всіх пристроях з Windows. Якщо програма використовує лише основні API, вона виконується на будь-якому пристрої з Windows 10, будь то настільний комп'ютер, Xbox, гарнітура змішаної реальності або щось інше. Вона дозволяє використовувати декларативний підхід для опису інтерфейсу (XAML), що спрощує розробку і покращує читабельність коду.

UWP підтримує сучасний дизайн та інтеграцію з Windows Store, що дозволяє розробникам розповсюджувати свої програми більш широко. Також UWP пропонує високий рівень безпеки завдяки обмеженому доступу до системних ресурсів. Однак ця платформа також має свої недоліки: обмежений набір API та залежність від Windows 10 і новіших версій.

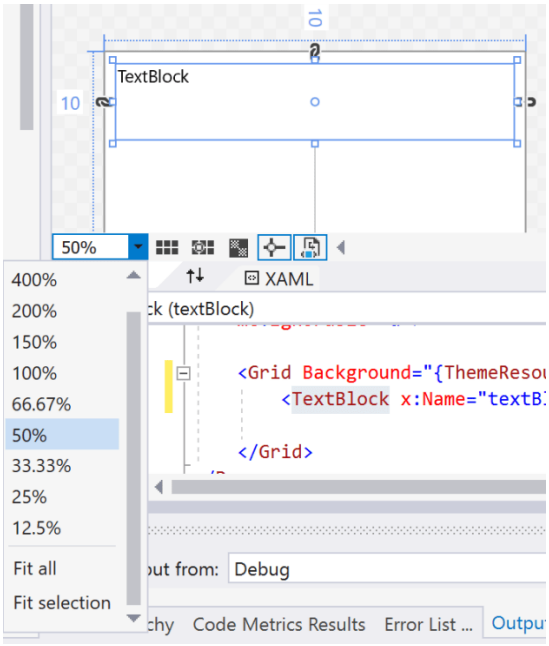

Рисунок 2.2 – Приклад візуальної розробки GUI у UWP

До переваг належать такі:

– модель інтерфейсу користувача на основі мови XAML з вбудованою підтримкою адаптації до змін розміру DPI і екрану;

– модель безпеки, в якій користувачі явно надають програмам доступ до можливостей пристроїв;

– модель упаковки, що підтримує установки та можливість публікації безпосередньо у Microsoft Store.

Windows Presentation Foundation (WPF) – це платформа інтерфейсу користувача для створення клієнтських програм для настільних систем [5]. Платформа розробки WPF підтримує широкий набір функцій розробки програм, у тому числі: Модель програми, Ресурси, Елементи керування, Графіка, Макет, Прив'язка даних, Документи, Безпека. Це частина бібліотек, що належать до ASP.NET Core або Windows Forms. WPF використовує XAML для надання декларативної моделі програмування програм.

WPF має значні переваги перед WinForms завдяки підтримці тривимірної графіки, анімацій, стилів та шаблонів. Це дозволяє розробникам створювати красиві, сучасні інтерфейси. Крім того, WPF забезпечує кращу продуктивність при відображенні складних візуальних компонентів. Але WPF також має свої обмеження: хоча вона значно просунута, але все одно залишається технологією, яка працює тільки у Windows. Для складних крос-платформних рішень вона не завжди підходить.

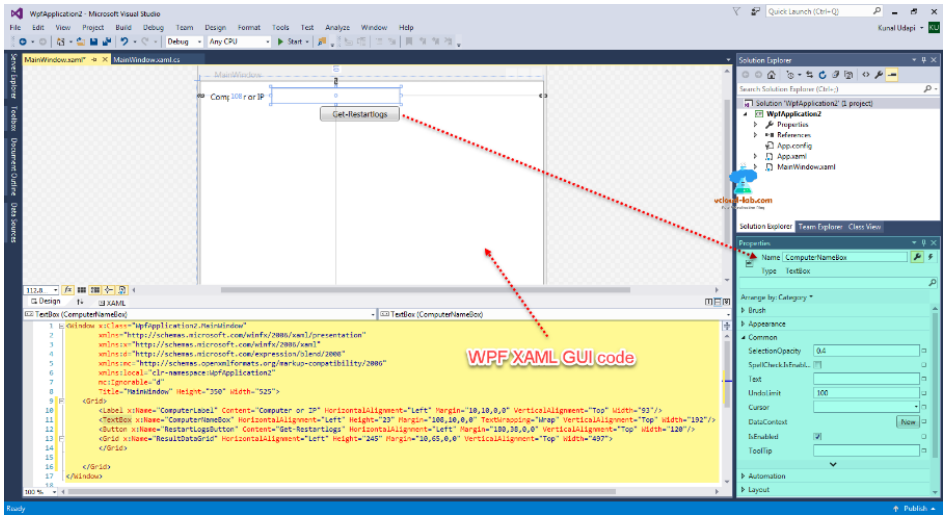

Рисунок 2.3 – Приклад візуальної розробки GUI у WPF

Кожна з трьох технологій має свої переваги й недоліки, і вибір між ними залежить від цілей проекту. Якщо необхідно розробити простий корпоративний застосунок, WinForms може стати кращим вибором завдяки своїй простоті. WPF підходить для більш складних проектів з вимогами до графіки і дизайну. UWP ж рекомендується для тих проектів, які орієнтовані на Windows 10 і мають потребу в крос-пристроєвій підтримці та інтеграції з магазином додатків Windows.

WPF та Windows Forms найкраще використовувати, якщо потрібно більш високий рівень продуктивності кодування, ніж C++ та Win32. Оскільки частини, відмінні від користувальницького інтерфейсу сучасної платформи.

У результаті аналізу сучасних GUI можна зробити висновок, що кожна з платформ залишається актуальною у своїй ніші. Правильний вибір залежить від конкретних вимог проекту, цільової аудиторії та бажаних можливостей розширення в майбутньому.

2.2 Постановка лабораторної роботи «Реалізація графічного інтерфейсу користувача засобами Windows Forms для роботи з колекціями»

Постановка завдання лабораторної роботи.

1. Згідно індивідуальному варіанту обрати предметну область. Визначити базовий клас у відповідності з індивідуальним варіантом опису предметної області. Додати до класу хоча б п’ять публічних різнотипних автореалізованих властивостей (обов’язково використати типи string, int, bool).

2. Розробити застосунок з графічним інтерфейсом користувача Windows Forms, за допомогою якого можна додавати об’єкти базового класу до колекції, видалення елементів із колекції та відображення елементів колекції. У вікні застосунку створити 5 текстових полів TextBox, у які користувач буде вводити данні з клавіатури. Створити кнопки для виконання операцій додавання об’єктів базового класу до колекції, видалення елементів із колекції

та відображення елементів колекції за допомогою елемента dataGridView. Інтерфейс не повинен бути переповнений елементами керування, але водночас повинен надавати найбільшу функціональність.

3. Підготувати звіт до захисту лабораторної роботи, що містить:

– опис предметної області;

– програмний код мовою C#;

– скриншот з результатами роботи програми.

Етап виконання лабораторної роботи.

Далі представлена послідовність дій для створення програми для зберігання списку студентів у колекції List.

1. Відкрийте MS Visual Studio.
2. Створіть нову програму Windows Forms.

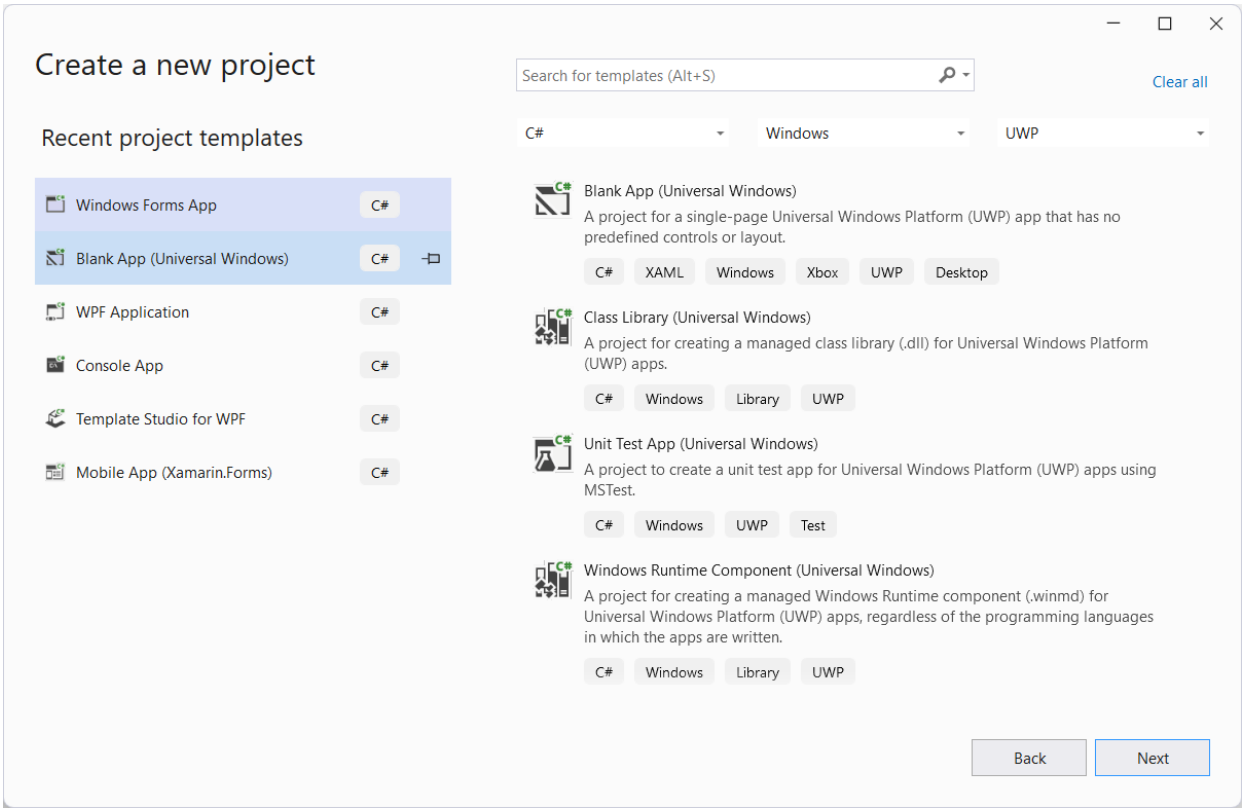

Рисунок 2.4 – Вікно створення проекту

3. З'явиться конструктор форми. Виділіть форму та у властивостях спробуйте змінити її назву та колір (заголовок форми – властивість Text, колір форми – BackColor).

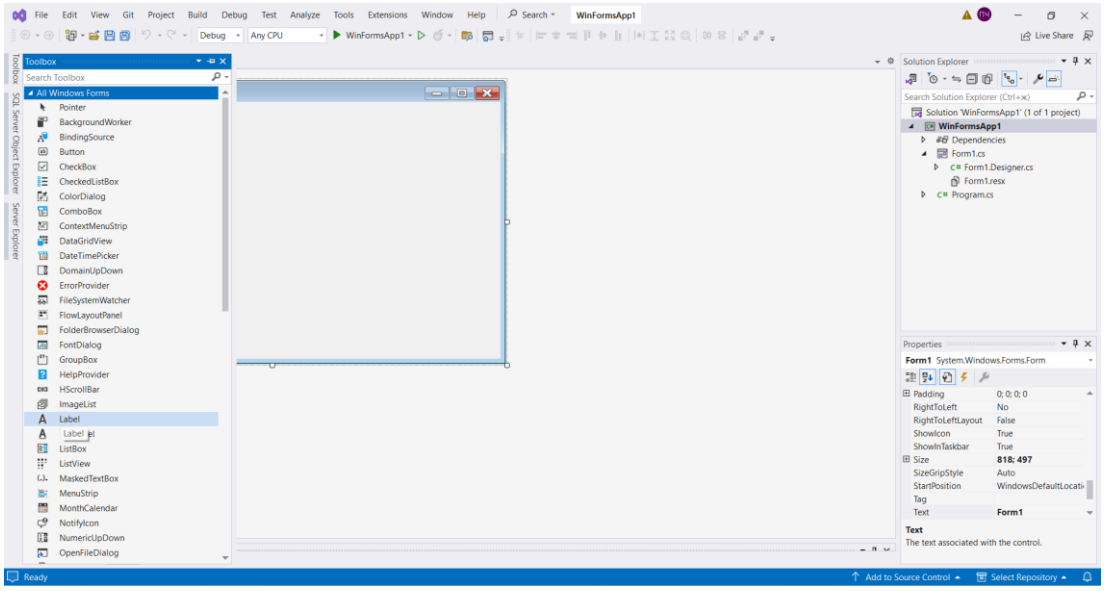

Рисунок 25 – Конструктор форми

3. Встановіть на форму панель SplitContainer з вікна "Toolbox". Це забезпечити поділ форми на 2 частини. Одна з частин буде використовуватися для введення даних, а інша для їх відображення.

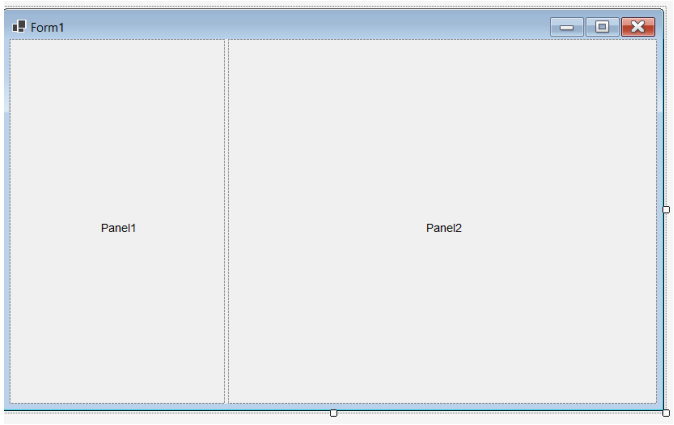

Рисунок 2.6 – Зовнішній вигляд форми після розміщення компонента SplitContainer

4. Розмістіть в областях Panel1 і Panel2 два компоненти Panel с розділу «Контейнери», і встановіть у панелей властивість Dock значення Fill. Ліва частина форми використовуватиметься для відображення даних, а права – для введення даних. Зробіть обідві частини контейнера та панелі однаковими за розміром.

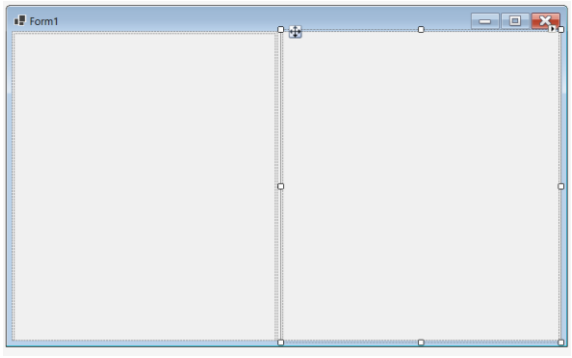

Рисунок 2.7 – Налаштування розмірів панелей

5. Для відображення списку студентів буде використано компонент DataGridView. Розмістить у лівій області форми компонент DataGridView і встановіть у нього властивість Dock значення Fill. З правого боку розмістить кілька елементів управління: три Label, три TextBox та кнопку Button. Дайте назву елементам у властивості "Text". Елементи введення можна озаглавлювати компонентом Label.

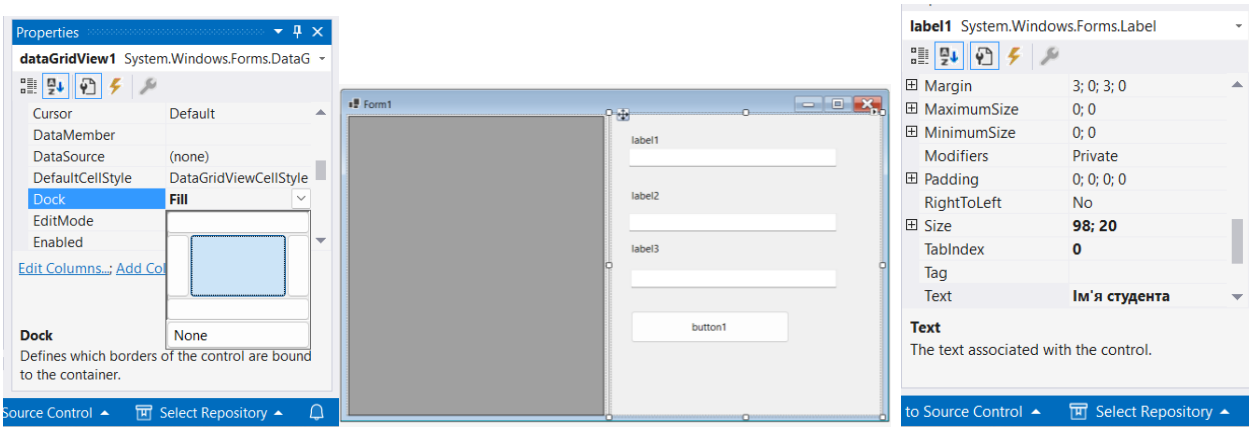

Рисунок 2.8 – Налаштування елементів управління

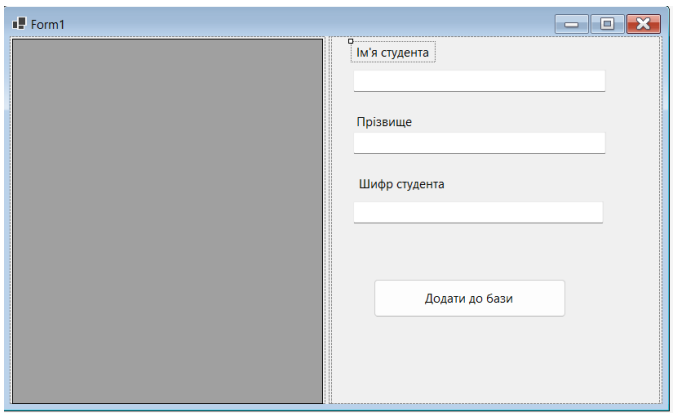

Рисунок 2.9 – Остаточний вигляд інтерфейсу програми

6. Оскільки програма буде працювати зі списком студентів, необхідно розробити клас студента. Створіть клас Student (виберіть у меню Project пункт «Add Class»).

Зразковий текст класу студента (файл Student.cs) наведено нижче:

```
internal class Student
{
  private string name;
  private string surname;
  private string recordBookNumber;

  public Student(string name, string surname, string recordBookNumber)
  {
    this.name = name;
    this.surname = surname;
    this.recordBookNumber = recordBookNumber;
  }

  public string getName()
  {
    return this.name;
  }
  public string getSurname()
  {
    return this.surname;
  }
  public string getRecordBookNumber()
  {
    return this.recordBookNumber;
  }
  public void setName(string name)
  {
    this.name = name;
  }
  public void setSurname(string surname)
  {
    this.surname = surname;
  }
  public void setRecordBookNumber(string recordBookNumber)
  {
    this.recordBookNumber = recordBookNumber;
  }
}
```

7. Встановіть на форму компонент ContextMenuStip (контекстне меню), додайте в нього одне поле «Видалити» та у властивості ContextMenuStip компонента DataGridView виберіть contextMenuStip1. Таким чином, у компонента DataGridView з'явиться власне контекстне меню.

8. Налаштуйте властивості DataGridView таким чином: ReadOnly – True, MultiSelect – False, RowHeadersVisible – False, SelectionMode – FullRowSelect.

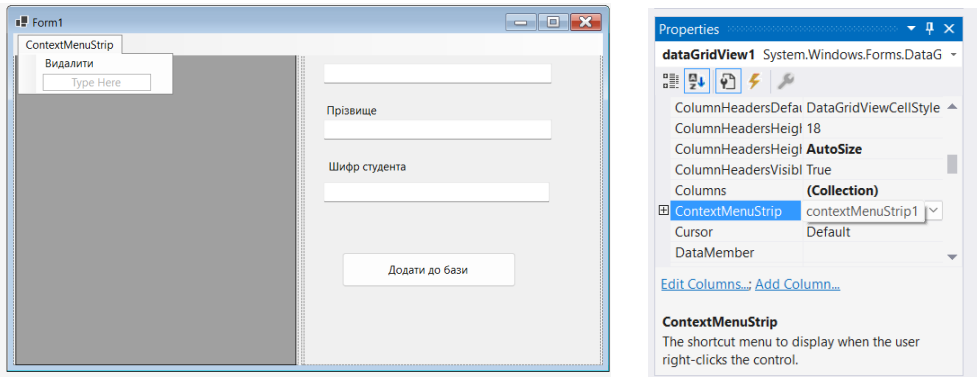

Рисунок 2.10 – Створення контекстного меню та його зв’язування з DataGridView

9. Компонент DataGridView складається з колонок DataGridViewColumn та рядків DataGridViewRow. Для того, щоб додавати в DataGridView рядки необхідно, щоб у ньому були колонки. Тому при ініціалізації форми необхідно додати DataGridView необхідну кількість колонок. Оскільки студент містить три поля, то й колонок у таблиці теж має бути три.

Відкрийте форму в режимі коду (натисніть F7). Додайте у клас Form1 перед конструктором “public Form1()” три поля-посилання на колонки таблиці:

```
private DataGridViewColumn dataGridViewColumn1=null;
private DataGridViewColumn dataGridViewColumn2=null;
private DataGridViewColumn dataGridViewColumn3=null;
```

Додайте у клас Form1 чотири методи – обробники даних:

```
public partial class Form1 : Form
{
  private DataGridViewColumn dataGridViewColumn1=null;
  private DataGridViewColumn dataGridViewColumn2=null;
  private DataGridViewColumn dataGridViewColumn3=null;

  //Ініціалізація таблиці
  private void initDataGridView()
  {
    dataGridView1.DataSource = null;
    dataGridView1.Columns.Add(getDataGridViewColumn1());
    dataGridView1.Columns.Add(getDataGridViewColumn2());
    dataGridView1.Columns.Add(getDataGridViewColumn3());
    dataGridView1.AutoResizeColumns();
  }

  //Динамічне створення першої колонки у таблиці
  private DataGridViewColumn getDataGridViewColumn1()
  {
    if (dataGridViewColumn1 == null)
```

```
    {
      dataGridViewColumn1 = new DataGridViewTextBoxColumn();
      dataGridViewColumn1.Name = "";
      dataGridViewColumn1.HeaderText = "Ім'я";
      dataGridViewColumn1.ValueType = typeof(string);
      dataGridViewColumn1.Width = dataGridView1.Width / 3;
    }
    return dataGridViewColumn1;
  }

  //Динамічне створення другої колонки у таблиці
  private DataGridViewColumn getDataGridViewColumn2()
  {
    if (dataGridViewColumn2 == null)
    {
      dataGridViewColumn2 = new DataGridViewTextBoxColumn();
      dataGridViewColumn2.Name = "";
      dataGridViewColumn2.HeaderText = "Прізвище";
      dataGridViewColumn2.ValueType = typeof(string);
      dataGridViewColumn2.Width = dataGridView1.Width / 3;
    }
    return dataGridViewColumn2;
  }

  //Динамічне створення третьої колонки у таблиці
  private DataGridViewColumn getDataGridViewColumn3()
  {
    if (dataGridViewColumn3 == null)
    {
      dataGridViewColumn3 = new DataGridViewTextBoxColumn();
      dataGridViewColumn3.Name = "";
      dataGridViewColumn3.HeaderText = "Заліковка";
      dataGridViewColumn3.ValueType = typeof(string);
      dataGridViewColumn3.Width = dataGridView1.Width / 3;
    }
    return dataGridViewColumn3;
  }

  public Form1()
  {
    InitializeComponent();
  }
}
```

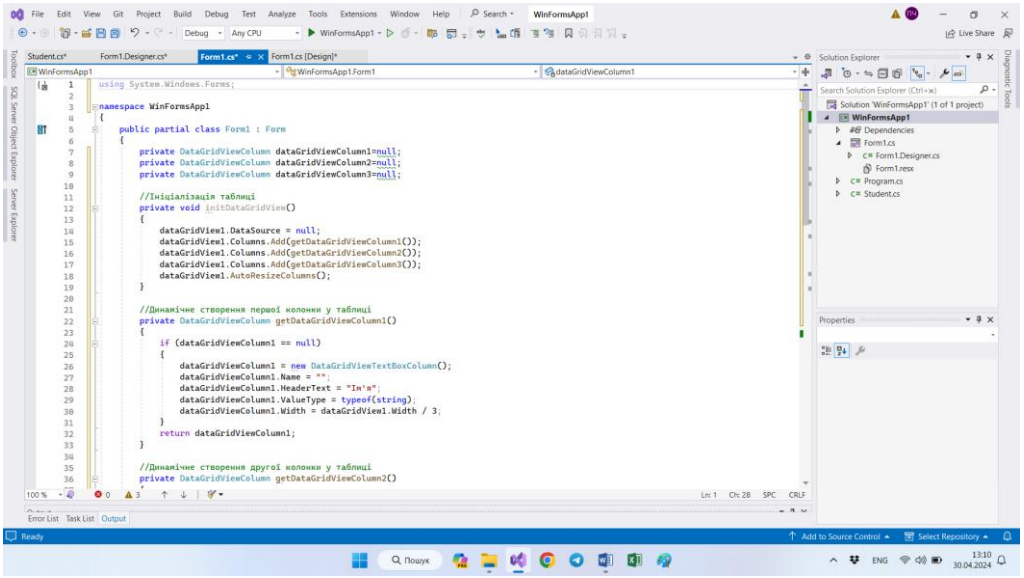

Рисунок 2.11 – Приклад процесу редагування коду

Перший метод додає в dataGridView1 три колонки, а останні три створюють ці колонки.

Додайте в конструктор виклик методу initDataGridView:

```
public Form1()
{
  InitializeComponent();
  initDataGridView();
}
```

10. Додайте перед конструктором поле-посилання на колекцію, в якій зберігатимуться записи про студентів:

```
//Колекція List
private IList<Student> studentList = new List<Student>();
```

11. Додайте у клас Form1 методи для додавання та видалення студентів до колекції та з колекції та для відображення колекції у DataGridView:

```
//Додавання студента до колекції
 private void addStudent(string name, string surname, string
recordBookNumber)
 {
   Student st = new Student(name, surname, recordBookNumber);
   studentList.Add(st);
   textBox1.Text = "";
   textBox2.Text = "";
   textBox2.Text = "";
   showListInGrid();
 }

 //Видалення студента з колекції
 private void deleteStudent(int elementIndex)
 {
   studentList.RemoveAt(elementIndex);
   showListInGrid();
 }

 //Відображення колекції таблиці
 private void showListInGrid()
 {
   dataGridView1.Rows.Clear();
   foreach (Student st in studentList)
   {
     DataGridViewRow row = new DataGridViewRow();
     DataGridViewTextBoxCell cell1 = new DataGridViewTextBoxCell();
     DataGridViewTextBoxCell cell2 = new DataGridViewTextBoxCell();
     DataGridViewTextBoxCell cell3 = new DataGridViewTextBoxCell();
     cell1.ValueType = typeof(string);
     cell1.Value = st.getName();
     cell2.ValueType = typeof(string);
     cell2.Value = st.getSurname();
     cell3.ValueType = typeof(string);
     cell3.Value = st.getRecordBookNumber();
     row.Cells.Add(cell1);
     row.Cells.Add(cell2);
     row.Cells.Add(cell3);
```

```
            dataGridView1.Rows.Add(row);
        }
    }
```

12. Додайте у клас Form1 обробники подій на кнопку додавання студента та на кнопку меню видалення студента:

```
//Обробник натискання на кнопку додавання
private void button1_Click(object sender, EventArgs e)
{
  addStudent(textBox1.Text, textBox2.Text, textBox3.Text);
}

//Обробник натискання на меню "Видалити"
private void ВидалитиToolStripMenuItem_Click(object sender, EventArgs e)
{
  int selectedRow = dataGridView1.SelectedCells[0].RowIndex;
  DialogResult dr = MessageBox.Show("Видалити студента?", "",
MessageBoxButtons.YesNo);
  if (dr == DialogResult.Yes)
  {
    try
    {
      deleteStudent(selectedRow);
    }
    catch (Exception)
    {
    }
  }
}
```

Далі необхідно зв’язати ці обробники з елементами управління. Кнопку button1 зв’язуємо з обробником *button1_Click*, пункт меню «Видалити» зв’язуємо з обробником *ВидалитиToolStripMenuItem_Click.* Для цього відмітимо кнопку, у вікні «Properties» обираєму вкладку «Events», знаходимо пункт «Click», у списку обираємо «button1_Click».

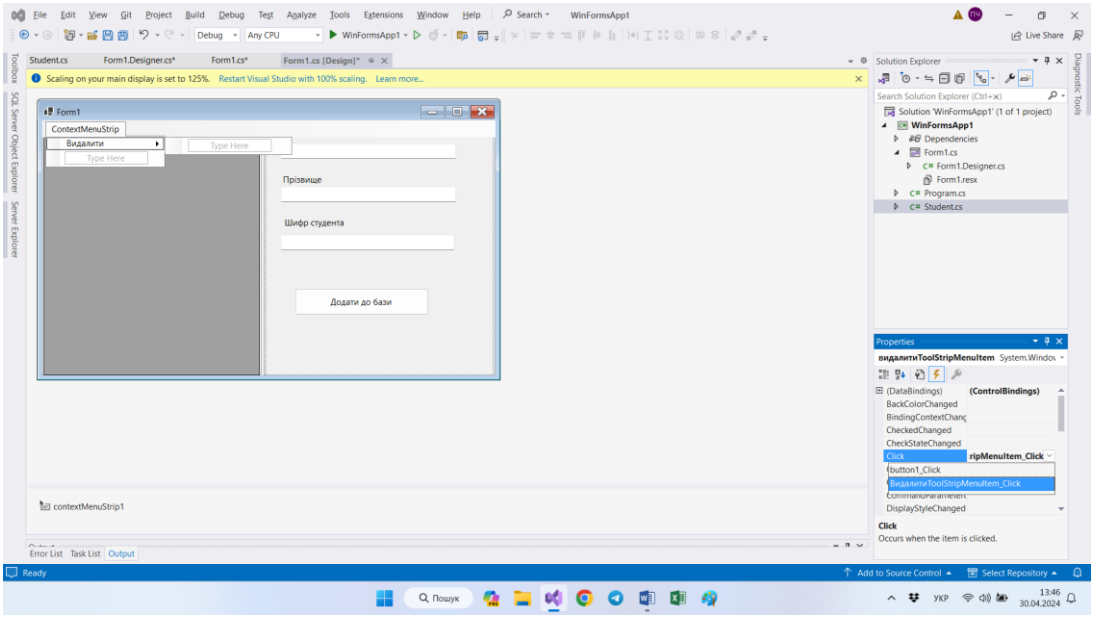

Рисунок 2.12 – Приклад зв’язування пункту меню «Видалити»

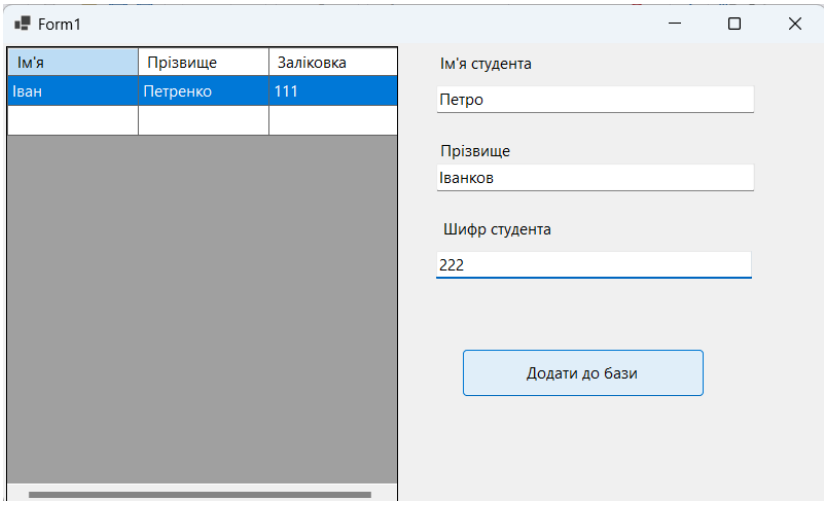

Рисунок 2.13 – Приклад додавання нового студента

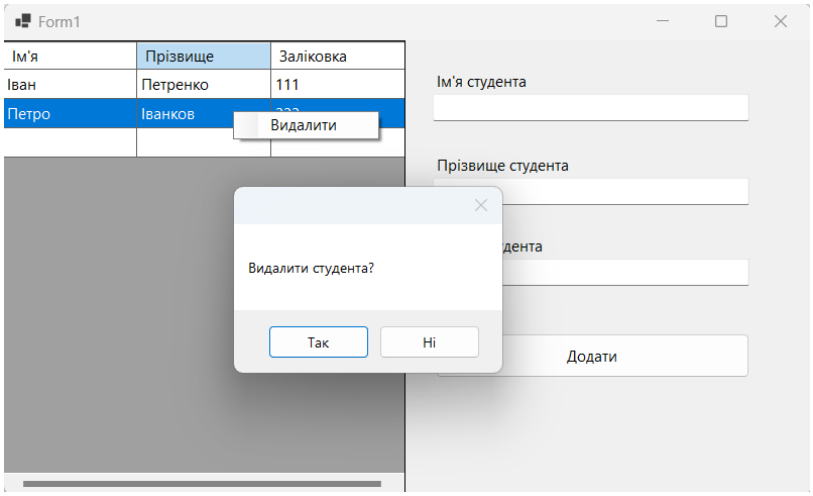

Рисунок 2.14 – Приклад видалення студента за допомогою контекстного меню

2.3 Постановка лабораторної роботи «Реалізація графічного інтерфейсу користувача засобами Windows Forms для перегляду вмісту збірки»

Постановка завдання лабораторної роботи.

1. Розробити застосунок з графічним інтерфейсом користувача Windows Forms, за допомогою якого можна переглядати вміст збірки. Застосунок буде виводити на екран інтерфейси та класи збірки, конструктори, методи та поля класів, передані та повертаються параметри методів.

2. Підготувати звіт до захисту лабораторної роботи, що містить:

– опис предметної області;

– програмний код мовою C#;

– скриншот з результатами роботи програми.

Далі представлена послідовність дій для створення програми для перегляду змісту збірки.

Етапи виконання лабораторної роботи.

1. Створіть нову програму Windows Forms.

2. Спроектуйте інтерфейс користувача: розмістить на формі меню (компонент MenuStrip), додайте пункт меню «Файл», до пункту меню файл додайте пункт меню «Відкрити збірку».

3. Розмістить на формі компонент TreeView і розгорніть його за розмірами форми, додайте компонент ImageList, зв’яжіть властивість «ImageList» компоненту TreeView з компонентом ImageList, додайте кілька картинок для вузлів дерева, додайте компонент OpenFileDialog.

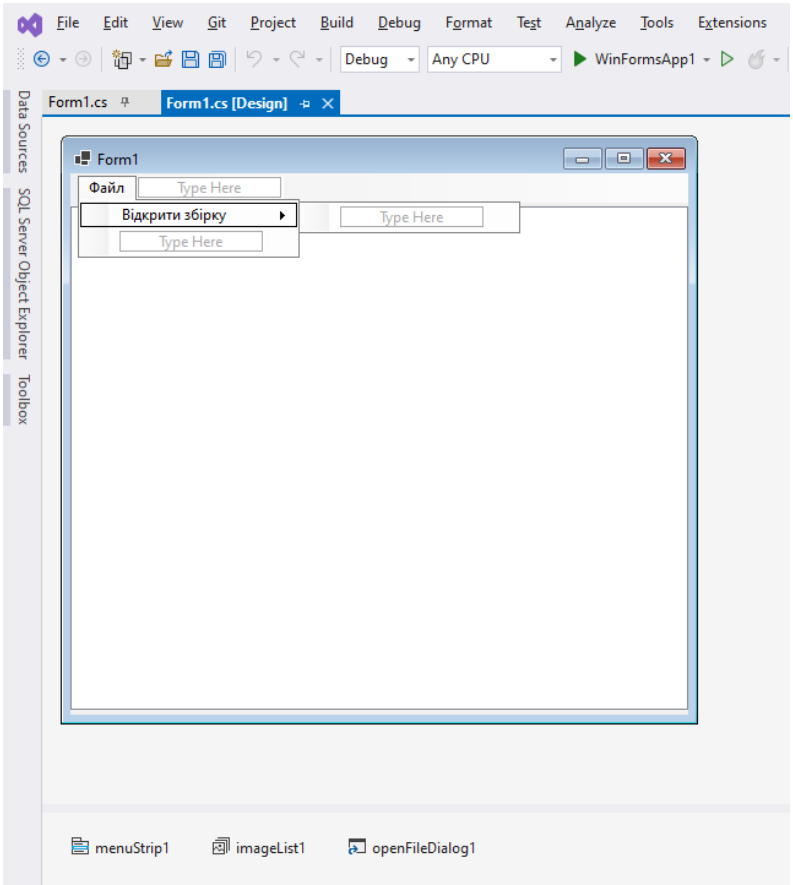

Рисунок 2.15 – Розробка інтерфейсу додатку

4. Відкрийте форму в режимі редактора коду, додайте до класу Form1 методи для відкриття збірки за допомогою OpenFileDialog.

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private string selectAssemblyFile()
    {
        openFileDialog1.Filter = "Dll files (*.dll)|*.dll|Exe files (*.exe) | *.exe | All files(*.*) | *.* ";
        openFileDialog1.Title = "Select assembly file";
        return (openFileDialog1.ShowDialog() == DialogResult.OK) ? openFileDialog1.FileName : null;
    }
```

Рисунок 2.16 – Метод selectAssemblyFile()

```
// Завантаження збірки
private Assembly openAssembly(string path)
{
    try
    {
        Assembly a = Assembly.LoadFrom(path);
        return a;
    }
    catch (Exception)
    {
        MessageBox.Show("Не вдалося завантажити збірку!", "Помилка!",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
}
```

Рисунок 2.17 – Метод openAssembly(string)

5. Додайте посилання на збирання:

```
public partial class Form1 : Form
{
    private Assembly assembly; // посилання на збірку

    public Form1()
    {
        InitializeComponent();
    }
```

Рисунок 2.18 – Посилання на збирання

6. Додайте методи додавання до дерева всіх класів та інтерфейсів, а також полів, конструкторів та методів класу.

```
// Додати всі класи та інтерфейси збірки до вузлів дерева
void addRoot(TreeNode root, Type[] types)
{
    TreeNode node = null;
    foreach (Type type in types)
    {
        node = new TreeNode();
        node.Text = type.ToString();
        // Якщо вузол – клас
        if (type.IsClass)
        {
            node.ImageIndex = 1;
            node.SelectedImageIndex = 1;
            addFirstLevel(node, type);
            root.Nodes.Add(node);
        }
        // Якщо вузол – інтерфейс
        else if (type.IsInterface)
        {
            node.ImageIndex = 2;
            node.SelectedImageIndex = 2;
            addFirstLevel(node, type);
            root.Nodes.Add(node);
        }
    }
}
```

Рисунок 2.19 – Код методу додавання класів та інтерфейсів

```
// Завіантажити всі поля, конструткори, методи
private void addFirstLevel(TreeNode node, Type type)
{
    TreeNode node1 = null;
    FieldInfo[] fields = type.GetFields(); MethodInfo[] methods = type.GetMethods();
    ConstructorInfo[] constructors = type.GetConstructors();

    // Завантажити поля
    foreach (FieldInfo field in fields)
    {
        node1 = new TreeNode();
        node1.Text = field.FieldType.Name + " " + field.Name;
        node1.ImageIndex = 5; node1.SelectedImageIndex = 5;
        node.Nodes.Add(node1);
    }

    // Завантажити конструктори
    foreach (ConstructorInfo constructor in constructors)
    {
        String s = "";
        ParameterInfo[] parametrs = constructor.GetParameters();
        foreach (ParameterInfo parametr in parametrs)
        { s = s + parametr.ParameterType.Name + ", "; }
        s = s.Trim(); s = s.TrimEnd(',');
        node1 = new TreeNode();
        node1.Text = node.Text + "(" + s + ")";
        node1.ImageIndex = 6; node1.SelectedImageIndex = 6;
        node.Nodes.Add(node1);
    }

    // Завіантажити методи
    foreach (MethodInfo method in methods)
    {
        String s = "";
        ParameterInfo[] parametrs = method.GetParameters();
        foreach (ParameterInfo parametr in parametrs)
        { s = s + parametr.ParameterType.Name + ", "; }
        s = s.Trim(); s = s.TrimEnd(',');
        node1 = new TreeNode();
        node1.Text = method.ReturnType.Name + " " + method.Name + "(" + s + ")";
        node1.ImageIndex = 4; node1.SelectedImageIndex = 4;
        node.Nodes.Add(node1);
    }
}
```

Рисунок 2.20 – Код методу додавання полів, методів та конструкторів

7. Додайте обробник натискання на пункт меню «Відкрити збірку».

```
private void відкритиЗбіркуToolStripMenuItem_Click(object sender, EventArgs e)
{
    treeView1.Nodes.Clear();
    string path = selectAssemblyFile();
    if (path != null)
    {
        assembly = openAssembly(path);
    }
    if (assembly != null)
    {
        TreeNode root = new TreeNode();
        root.Text = assembly.GetName().Name;
        root.ImageIndex = 0;
        root.SelectedImageIndex = 0;
        treeView1.Nodes.Add(root);
        Type[] types = assembly.GetTypes();
        addRoot(root, types);
    }
}
```

Рисунок 2.21 – Код методу відкриття збірки

8. Додайте до коду додатку опис базового класу індивідуального варіанту.

9. Запустіть проект і спробуйте відкрити будь яку збірку (*.dll). Можна, наприклад, відкрити поточну збірку проекту.

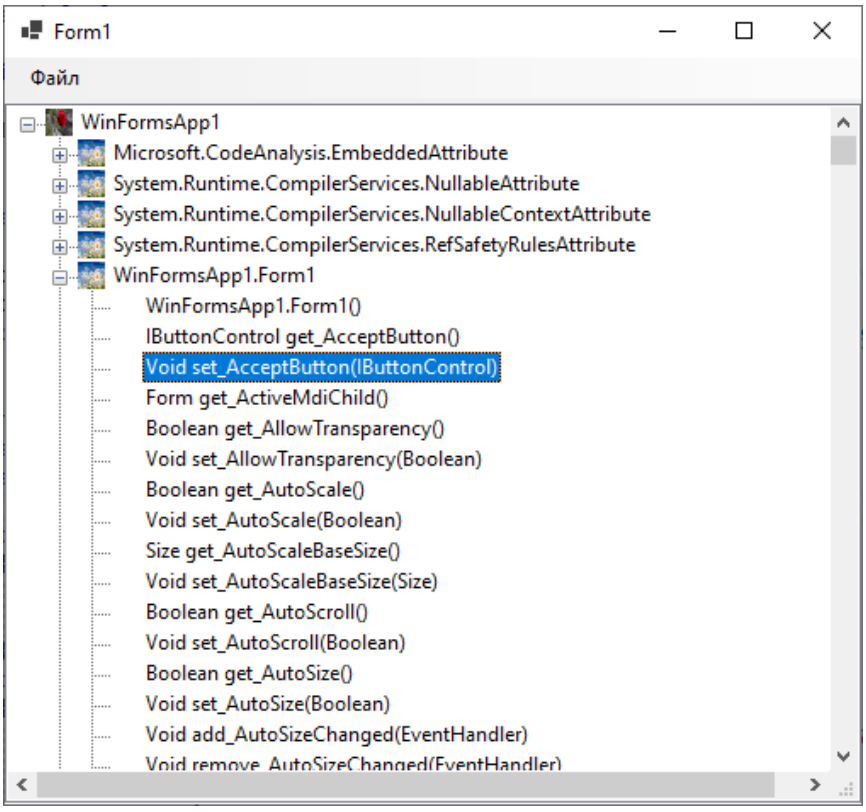

Рисунок 2.22 – Перегляд інформації про поточну збірку

2.4 Постановка лабораторної роботи «Реалізація графічного інтерфейсу користувача засобами Windows Presentation Foundation»

1. Згідно індивідуальному варіанту обрати предметну область.

2. Розробити застосунок з графічним інтерфейсом Windows Presentation Foundation, за допомогою якого можна додавати записи до колекції, видалення елементів із колекції та відображення елементів колекції. У вікні застосунку створити 5 текстових полів TextBox, у які користувач буде вводити данні з клавіатури. Створити кнопки для виконання операцій додавання об’єктів базового класу до колекції, видалення елементів із колекції та відображення елементів колекції за допомогою елемента dataGrid. Інтерфейс не повинен бути переповнений елементами керування, але водночас повинен надавати найбільшу функціональність.

3. Підготувати звіт до захисту лабораторної роботи, що містить:

– опис предметної області;

– програмний код мовою C#;

– скриншот з результатами роботи програми.

Етап виконання лабораторної роботи.

Далі представлена послідовність дій для створення програми для зберігання списку рядків у колекції.

Опис предметної області: у банку вакансій кожна позиція характеризується назвою, компанією, що її пропонує, категорією вакансії та регіоном. Забезпечити виведення на екран інформації про вакансії будь-якого типу групування.

1. Відкрийте MS Visual Studio.

2. Створіть нову програму Windows Presentation Foundation.

3. Після створення програми Visual Studio повинна відкритися панель конструктора XAML для форми за замовчуванням MainWindow. Підтримка WPF у Visual Studio складається з п'яти важливих компонентів, з якими ви будете взаємодіяти під час створення програми (рис. 2.23).

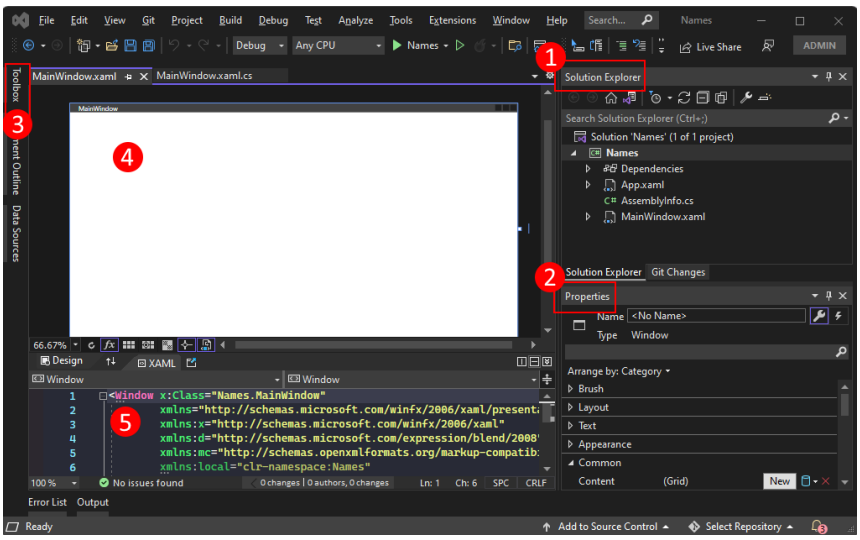

Рисунок 2.23 – Компоненти середовища Visual Studio для WPF

– оглядач рішень: усі файли проекту, код, вікна та ресурси відображаються у цій області;

– властивості: на цій панелі відображаються параметри властивостей, які можна настроїти залежно від вибраного елемента;

– панель елементів містить усі елементи керування, які можна додати до форми; щоб додати елемент керування на поточну форму, двічі клацніть або перетягніть елемент керування;

– конструктор XAML – це архітектор для документа; він є інтерактивним, на нього можна перетягувати об'єкти з панелі елементів; вибираючи та переміщуючи елементи в конструкторі, можна візуально створювати інтерфейс користувача для програми;

– редактор коду XAML, це спосіб створення інтерфейсу користувача вручну без конструктора; Конструктор може обчислювати значення властивостей елемента управління при додаванні в конструктор.

4. Після створення проекту в редакторі коду XAML відображається мінімальний обсяг коду XAML для відображення вікна. Він описує інтерфейс WPF і взаємодіє з кодом .NET. Вводимо код XAML, показаний далі:

```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApp1"
```

```
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">

    <!-- Контекстне меню Drid'y -->

    <Window.Resources>
        <ContextMenu x:Key="ContextMenu">
            <MenuItem Header="Видалити" Click="DeleteMenuItem_Click"/>
        </ContextMenu>
    </Window.Resources>

    <Grid Margin="0,0,0,5">
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <!-- Ліва область -->
        <DataGrid Name="dataGridView" AutoGenerateColumns="False" Margin="10,10,177,10"
                  IsReadOnly="True" SelectionMode="Single" CanUserAddRows="False"
                  CanUserDeleteRows="False" CanUserResizeRows="False"
SelectionUnit="FullRow"
                  CanUserResizeColumns="False" ItemsSource="{Binding vacancies}"
                  MouseRightButtonDown="DataGridView_MouseRightButtonDown">

            <DataGrid.Columns>
                <DataGridTextColumn Header="Назва" Binding="{Binding Title}" Width="*"
IsReadOnly="True"/>
                <DataGridTextColumn Header="Компанія" Binding="{Binding Company}"
Width="*" IsReadOnly="True"/>
                <DataGridTextColumn Header="Категорія" Binding="{Binding Category}"
Width="*" IsReadOnly="True"/>
                <DataGridTextColumn Header="Регіон" Binding="{Binding Region}" Width="*"
IsReadOnly="True"/>
            </DataGrid.Columns>
        </DataGrid>

        <!-- Права область -->
        <Grid Grid.Row="0" Background="White" Margin="628,0,0,0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>

            <!-- Елементи управління -->
            <Grid Grid.Column="0">
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto" MinHeight="69.298"/>
                    <RowDefinition Height="Auto" MinHeight="25.96"/>
                </Grid.RowDefinitions>
                <Label Grid.Row="0" Content="Назва" Margin="0,0,0,18" Grid.RowSpan="2"/>
                <Label Grid.Row="1" Content="Компанія" Margin="0,56,0,-30"
Grid.RowSpan="2"/>
                <Label Grid.Row="2" Content="Категорія" Margin="0,30,0,-30"/>
                <Label Grid.Row="2" Content="Регіон:" Margin="0,82,0,-82"/>
            </Grid>

            <!-- Поля вводу -->
            <Grid Grid.Column="1">
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="19*"/>
                    <RowDefinition Height="107*"/>
                </Grid.RowDefinitions>
```

```
                <TextBox Grid.Row="1" Name="VacancyTitle" Margin="-82,32,10,341"/>
                <TextBox Grid.Row="1" Name="VacancyCompany" Margin="-82,81,10,291"/>
                <TextBox Grid.Row="1" Name="VacancyCategory" Margin="-82,135,10,238"/>
                <TextBox Grid.Row="1" Name="VacancyRegion" Margin="-82,173,10,196"/>
                <Button Grid.Row="1" Content="Додати" Click="Button_Click" Margin="-
82,217,10,147"/>
            </Grid>
        </Grid>
    </Grid>
    </Window>
```

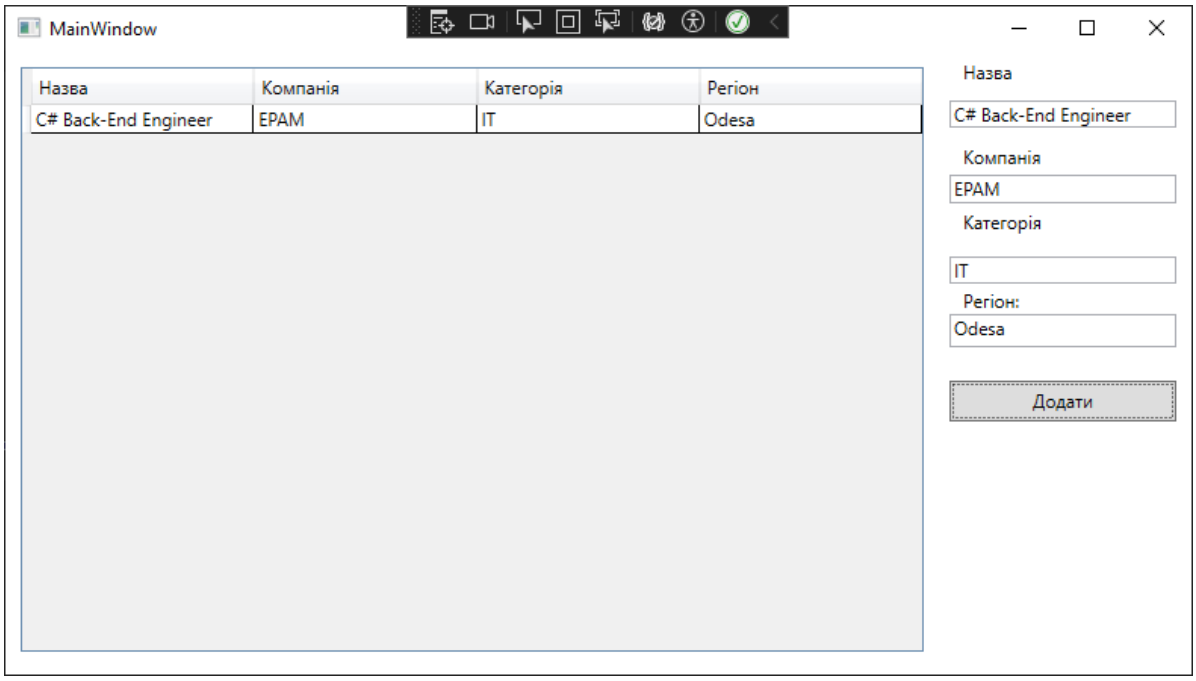

Рисунок 2.24 – Приклад додавання записів у WPF застосунку

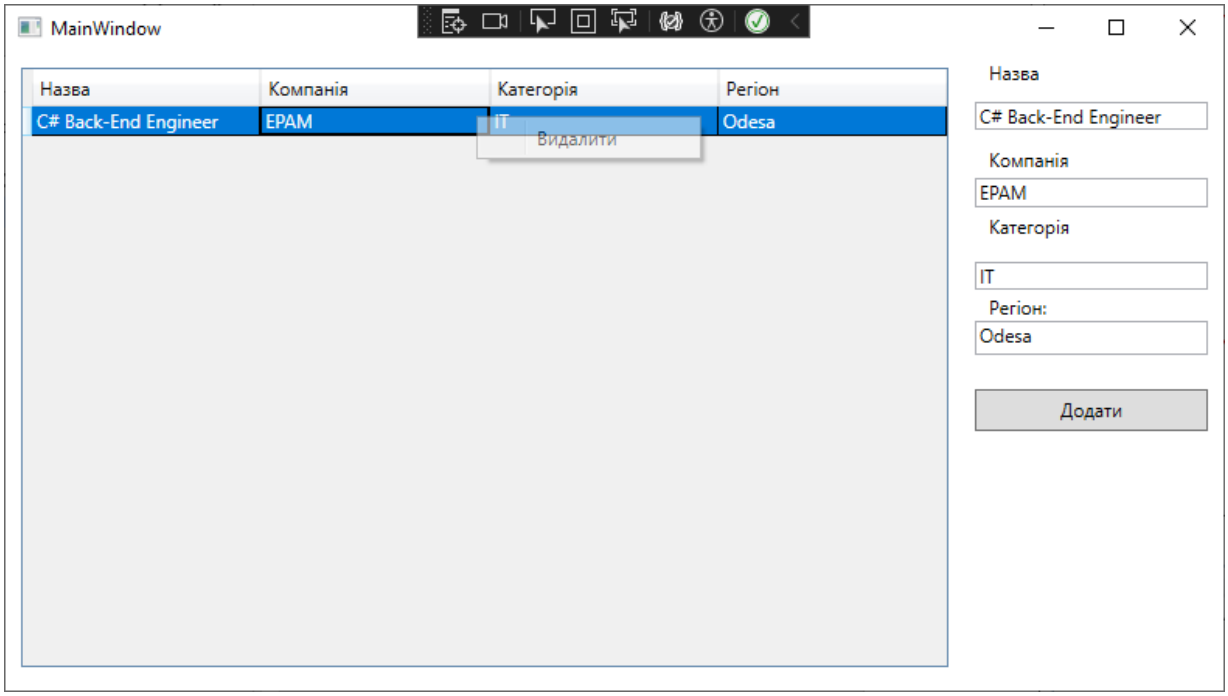

Рисунок 2.25 – Приклад видалення записів у WPF застосунку

2.5 Постановка лабораторної роботи «Реалізація графічного інтерфейсу користувача засобами Universal Windows Platform»

1. Згідно індивідуальному варіанту обрати предметну область.

2. Розробити застосунок з графічним інтерфейсом Universal Windows Platform, за допомогою якого можна додавати записи до колекції, видалення елементів із колекції та відображення елементів колекції. У вікні застосунку створити 5 текстових полів TextBox, у які користувач буде вводити данні з клавіатури. Створити кнопки для виконання операцій додавання об’єктів базового класу до колекції, видалення елементів із колекції та відображення елементів колекції за допомогою елемента dataGrid. Інтерфейс не повинен бути переповнений елементами керування, але водночас повинен надавати найбільшу функціональність.

3. Підготувати звіт до захисту лабораторної роботи, що містить:

– опис предметної області;

– програмний код мовою C#;

– скриншот з результатами роботи програми.

Етап виконання лабораторної роботи.

Далі представлена послідовність дій для створення програми для зберігання списку рядків у колекції.

1. Відкрийте Visual Studio та створіть новий проєкт. Виберіть шаблон Blank App (Universal Windows). Вкажіть назву проєкту (MyApp), місце збереження та натисніть Create (рис. 2.26). Виберіть цільову та мінімальну версію платформи, з якою буде сумісний застосунок.

2. Під час проектування інтерфейсу необхідно у головному файлі MainPage.xaml створити базовий інтерфейс, що містить 5 текстових полів TextBox для введення даних з мітками, а також додати кнопки для виконання основних дій (рис. 2.27):

– кнопку «Додати» для збереження даних з текстових полів до колекції.

– кнопку «Видалити» для видалення обраного елемента з колекції.

– кнопку «Показати» для оновлення списку у DataGrid, який відображатиме всі елементи колекції.

Також використаємо елемент DataGrid для зручного відображення записів.

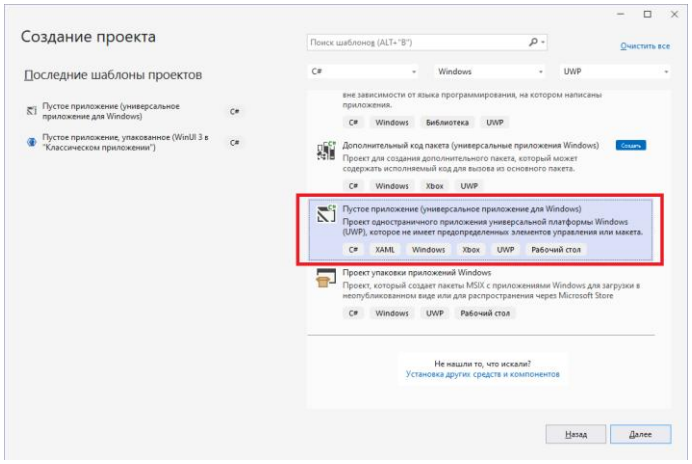

Рисунок 2.26 – Вікно створення UWP застосунку

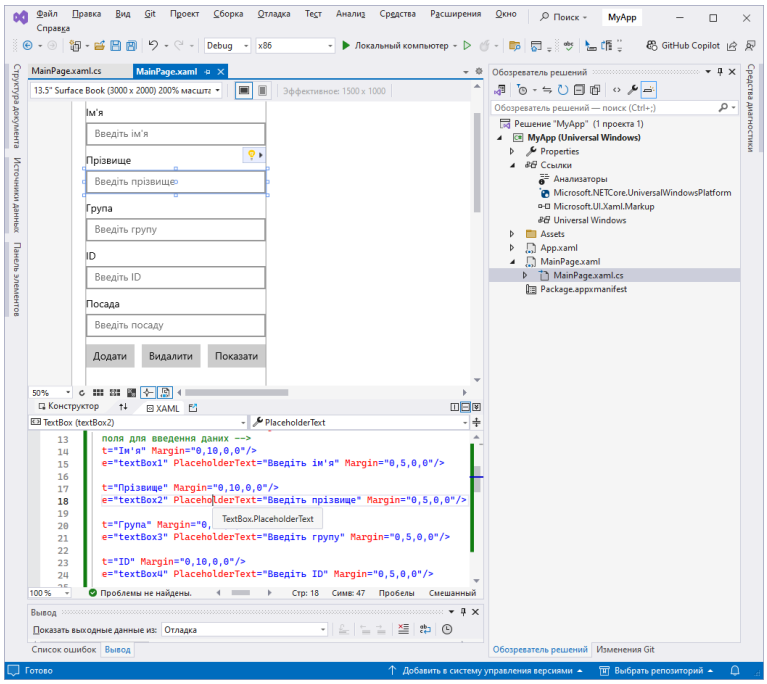

Рисунок 2.27 – Приклад розробки графічного інтерфейсу

3. У UWP застосунках інтерфейси користувача створюються за допомогою мови розмітки XAML, що дозволяє створювати адаптивні та інтерактивні елементи для додатків Windows, таких як текст, кнопка та поле вводу. XAML-код інтерфейсу показано далі.

```
<Page
    x:Class="MyApp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:MyApp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Grid Padding="20">
        <StackPanel Orientation="Vertical" HorizontalAlignment="Left">
            <!-- Мітки та поля для введення даних -->
            <TextBlock Text="Ім'я" Margin="0,10,0,0"/>
            <TextBox x:Name="textBox1" PlaceholderText="Введіть ім'я"
Margin="0,5,0,0"/>

            <TextBlock Text="Прізвище" Margin="0,10,0,0"/>
            <TextBox x:Name="textBox2" PlaceholderText="Введіть прізвище"
Margin="0,5,0,0"/>

            <TextBlock Text="Група" Margin="0,10,0,0"/>
            <TextBox x:Name="textBox3" PlaceholderText="Введіть групу"
Margin="0,5,0,0"/>

            <TextBlock Text="ID" Margin="0,10,0,0"/>
            <TextBox x:Name="textBox4" PlaceholderText="Введіть ID"
Margin="0,5,0,0"/>

            <TextBlock Text="Посада" Margin="0,10,0,0"/>
            <TextBox x:Name="textBox5" PlaceholderText="Введіть посаду"
Margin="0,5,0,0"/>

            <!-- Кнопки для операцій -->
            <StackPanel Orientation="Horizontal" Margin="0,10,0,0">
                <Button Content="Додати" Click="AddButton_Click" Margin="0,0,10,0"/>
                <Button Content="Видалити" Click="RemoveButton_Click"
Margin="0,0,10,0"/>
                <Button Content="Показати" Click="ShowButton_Click"/>
            </StackPanel>
        </StackPanel>

        <!-- GridView для відображення елементів -->
        <GridView x:Name="gridView" HorizontalAlignment="Right" Width="400"
Margin="20,0,0,0" SelectionMode="Single">
            <GridView.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal" Margin="5">
                        <TextBlock Text="{Binding Field1}" Margin="5,0"/>
                        <TextBlock Text="{Binding Field2}" Margin="5,0"/>
                        <TextBlock Text="{Binding Field3}" Margin="5,0"/>
                        <TextBlock Text="{Binding Field4}" Margin="5,0"/>
                        <TextBlock Text="{Binding Field5}" Margin="5,0"/>
                    </StackPanel>
                </DataTemplate>
            </GridView.ItemTemplate>
        </GridView>
    </Grid>
</Page>
```

4. Створення моделі даних. У файлі MainPage.xaml.cs додаємо клас, який буде представляти запис з полями, що відповідають текстовим полям. Створюємо колекцію ObservableCollection<Record> для зберігання записів.

Реалізуємо логіку для додавання нового запису до колекції, видалення вибраного запису та оновлення відображення. Код на мові C# показано далі.

```
namespace MyApp
{
    public class Record
    {
        public string Field1 { get; set; }
        public string Field2 { get; set; }
        public string Field3 { get; set; }
        public string Field4 { get; set; }
        public string Field5 { get; set; }
    }

    public sealed partial class MainPage : Page
    {
        private ObservableCollection<Record> records = new
                ObservableCollection<Record>();

        public MainPage()
        {
            this.InitializeComponent();
            gridView.ItemsSource = records;
        }

        private void AddButton_Click(object sender, RoutedEventArgs e)
        {
            // Додавання нового запису до колекції
            records.Add(new Record
            {
                Field1 = textBox1.Text, Field2 = textBox2.Text,
                Field3 = textBox3.Text, Field4 = textBox4.Text,
                Field5 = textBox5.Text
            });

            // Очищення полів після додавання
            textBox1.Text = string.Empty; textBox2.Text = string.Empty;
            textBox3.Text = string.Empty; textBox4.Text = string.Empty;
            textBox5.Text = string.Empty;
        }

        private void RemoveButton_Click(object sender, RoutedEventArgs e)
        {
            // Видалення вибраного запису
            if (gridView.SelectedItem is Record selectedRecord)
            {
                records.Remove(selectedRecord);
            }
        }

        private void ShowButton_Click(object sender, RoutedEventArgs e)
        {
            // Оновлення відображення
            gridView.ItemsSource = null;
            gridView.ItemsSource = records;
        }
    }
}
```

5. Запуск і тестування. Запустіть застосунок, натиснувши F5 або кнопку Start у Visual Studio. У вікні введіть дані в текстові поля, натисніть кнопку «Додати», запис з’явиться в GridView (рис. 2.28). Виділіть запис у GridView

(рис. 2.29) та натисніть кнопку «Видалити», запис буде видалений. Натисніть кнопку «Показати» для оновлення відображення списку записів (рис. 2.30).

Застосунок має простий і функціональний інтерфейс, де користувач може додавати, переглядати та видаляти записи, використовуючи GridView для відображення колекції записів.

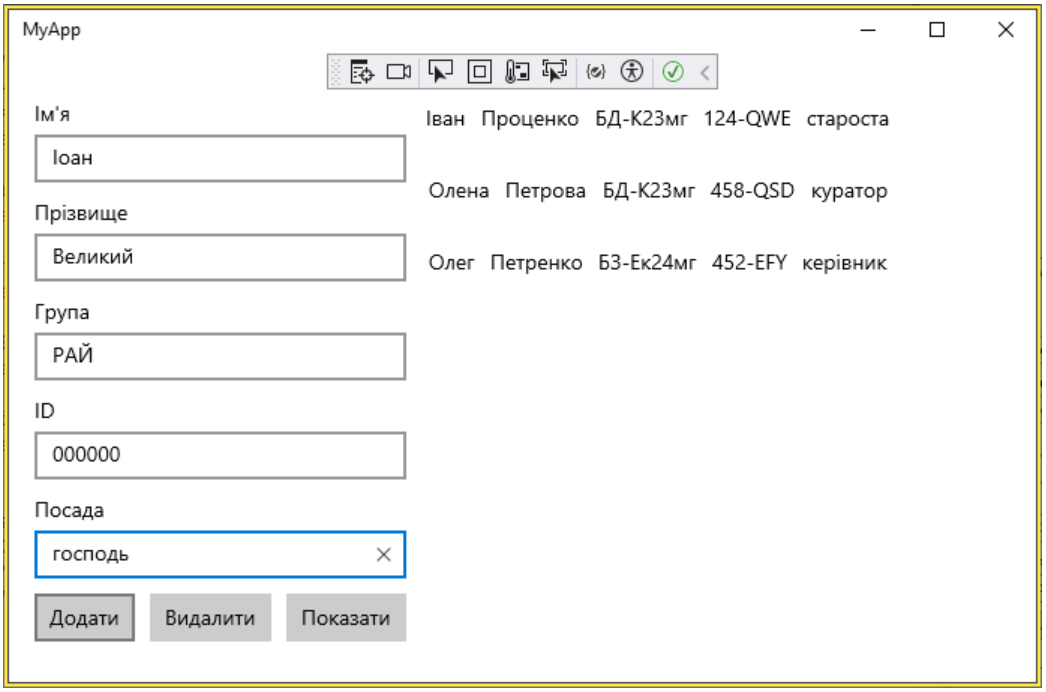

Рисунок 2.28 – Приклад додавання запису до колекції

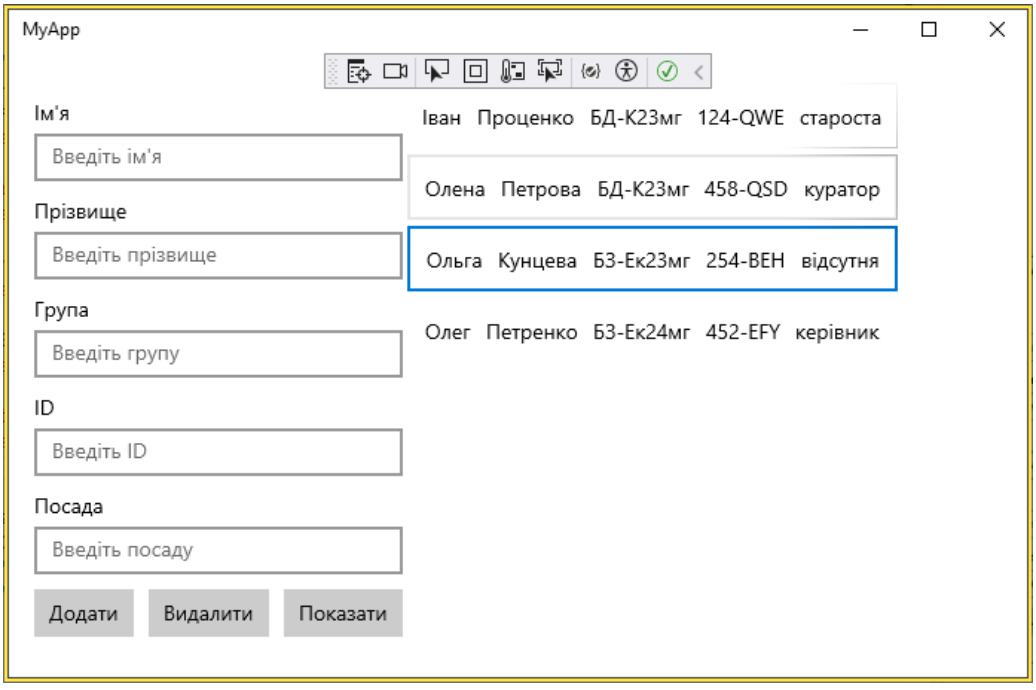

Рисунок 2.29 – Приклад вибору запису у колекції

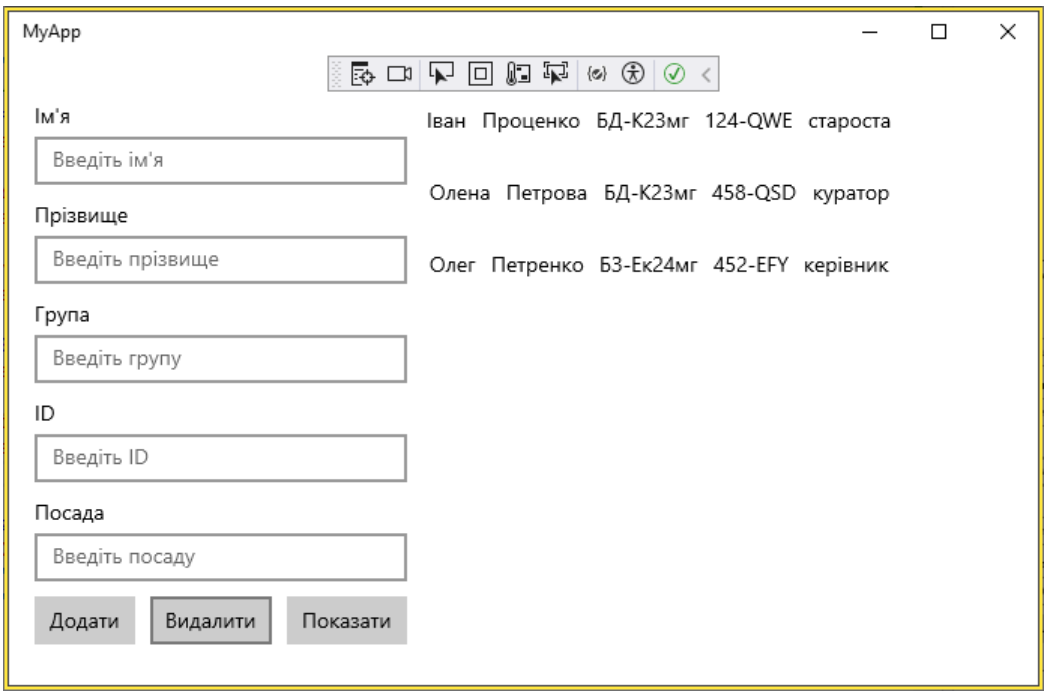

Рисунок 2.30 – Приклад оновленої колекції після видалення

Висновки до розділу

У розділі виконано огляд засобів розробки застосунків з графічним інтерфейсом для платформи .NET. Виконана постановка нового лабораторного практикуму для вивчення теми «Реалізація графічного інтерфейсу користувача на платформі .NET», зокрема поставлено такі роботи:

1. реалізація графічного інтерфейсу користувача засобами Windows Forms для роботи з колекціями;

2 реалізація графічного інтерфейсу користувача засобами Windows Forms для перегляду вмісту збірки;

3. реалізація графічного інтерфейсу користувача засобами Windows Presentation Foundation;

4. реалізація графічного інтерфейсу користувача засобами Universal Windows Platform.

Розроблено завдання лабораторних робіт, наведено приклади виконання робіт та оформлення звітів, що містять описи етапів побудови застосунків, програмний код на мові C#, приклад виконання програми у Visual Studio.

РОЗДІЛ 3 ВИМОГИ ДО КАДРОВОГО ЗАБЕЗПЕЧЕННЯ ОБ'ЄКТУ ГАЛУЗІ

У сучасному світі інформаційних технологій роль розробників програмного забезпечення, особливо тих, хто спеціалізується на створенні Windows-застосунків з графічним інтерфейсом користувача (GUI), стає дедалі важливішою. Кадрове забезпечення цієї категорії фахівців є ключовим чинником для успішної реалізації проектів, які мають на меті задовольнити потреби кінцевого користувача, забезпечити високу продуктивність та безпеку. Далі наведено результати аналізу, як відбувається кадрове забезпечення розробників, які навички та знання вони повинні мати, а також вплив цього процесу на якість кінцевого продукту.

Розробники Windows-застосунків з графічним інтерфейсом виконують важливу функцію в розробці програмного забезпечення. Вони відповідають за створення інтуїтивно зрозумілого та привабливого інтерфейсу, що полегшує взаємодію користувачів з програмою. Інтерфейс користувача – це перше, що бачить користувач, тому його дизайн має бути не лише естетичним, а й функціональним. Успішний розробник повинен мати глибоке розуміння принципів дизайну, знання сучасних технологій, таких як XAML і UWP, а також вміння адаптувати додатки під різні пристрої та екрани [6].

Кадрове забезпечення розробників Windows-застосунків передбачає не лише наявність технічних навичок, а й особистісних якостей. Серед основних технічних навичок можна виділити таке.

1. Знання мов програмування: основною мовою для розробки Windows-застосунків є C#. Розробники повинні володіти цією мовою, а також знати основи програмування на C++ або JavaScript, оскільки це розширює можливості роботи з різними фреймворками та бібліотеками.

2. Досвід роботи з платформою Windows: розуміння особливостей Windows API, UWP, WinUI, WPF, а також здатність використовувати

різноманітні інструменти розробки, такі як Visual Studio, є критично важливими для успіху.

3. Дизайн інтерфейсу користувача: розробники повинні бути знайомі з основами UX/UI дизайну, включаючи принципи створення доступних і зручних інтерфейсів, щоб задовольнити потреби користувачів з різними можливостями [7].

4. Знання архітектурних патернів: вміння працювати з патернами проектування, такими як MVC (Model-View-Controller), MVVM (Model-View-ViewModel) та інші, є важливим для організації коду і спрощення його підтримки та розширення.

5. Кросплатформена розробка: у сучасному світі важливо також знати технології, які дозволяють створювати кросплатформені застосунки, що працюють не лише на Windows, але й на інших платформах, таких як Android та iOS.

Рисунок 3.1 – Етапи кадрового відбору розробників ПЗ

Багато розробників отримують освіту в галузі комп'ютерних наук, дизайну або суміжних дисциплін. Ступінь бакалавра або магістра у цих областях є звичним початком кар'єри. Університети та коледжі пропонують програми, що спеціалізуються на взаємодії людини з комп'ютером (HCI) та графічному дизайні.

Важливим елементом у підготовці розробника є створення портфоліо, яке демонструє його навички та проекти. Це може бути вирішальним фактором під час пошуку роботи, оскільки роботодавці хочуть бачити реальні приклади роботи кандидатів.

Важливим аспектом підготовки є практичний досвід. Студенти часто залучаються до стажувань або проектів, що дозволяють їм працювати над реальними задачами та розвивати портфоліо. Це може включати участь у командних проектах, хакатонах або навіть фріланс.

У США існує великий акцент на новітніх технологіях і трендах у дизайні. Розробники графічного інтерфейсу повинні бути в курсі останніх досягнень у сфері UX/UI-дизайну, мобільної розробки, доповненої реальності (AR) та віртуальної реальності (VR) [8].

Розробники часто активно беруть участь у професійних спільнотах, конференціях та форумах, що дозволяє їм обмінюватися досвідом, отримувати поради від колег та бути в курсі новинок галузі.

Процес кадрового забезпечення починається з навчання та підготовки спеціалістів. Університети та технічні навчальні заклади пропонують програми, що орієнтуються на комп'ютерні науки, програмування та розробку програмного забезпечення. Важливо, щоб навчальні програми включали практичні заняття, де студенти можуть отримати реальний досвід роботи з проектами.

Крім формальної освіти, важливими є також курси підвищення кваліфікації, семінари та онлайн-курси, які допомагають розробникам тримати руку на пульсі нових технологій і трендів. Платформи, такі як Coursera, Udemy, Pluralsight [9-11], пропонують безліч курсів, присвячених Windows-

розробці, що дозволяє фахівцям безперервно вдосконалювати свої навички. Багато розробників проходять додаткові курси або сертифікаційні програми, які фокусуються на специфічних технологіях та інструментах для створення графічних інтерфейсів, таких як Adobe XD, Figma, Sketch, а також фреймворки для веб-розробки, такі як React або Angular.

Рисунок 3.2 – Онлайн-курс Coursera для кадрового відбору розробників ПЗ

Якість кадрового забезпечення безпосередньо впливає на якість кінцевого продукту. Непідготовлені або недостатньо досвідчені розробники можуть призвести до створення продукту з низькою продуктивністю, помилками в інтерфейсі, або навіть до безпеки програмного забезпечення. Наявність кваліфікованих фахівців у команді може суттєво покращити не лише якість коду, але й задоволеність користувачів.

Компанії, які інвестують у навчання своїх співробітників, здатні створювати більш конкурентоспроможні продукти. Коли розробники постійно навчаються, вони стають більш інноваційними, здатними впроваджувати нові технології та методи, що, в свою чергу, підвищує загальну продуктивність команди.

Одним із основних викликів у кадровому забезпеченні розробників Windows-застосунків є швидка еволюція технологій. З новими фреймворками та інструментами, які з'являються на ринку, важко підтримувати актуальність навичок фахівців. Крім того, конкуренція на ринку праці за кваліфікованих

розробників зростає, тому компанії повинні пропонувати привабливі умови праці, такі як гнучкий графік, можливості для професійного розвитку та конкурентоспроможну зарплату.

Ще одним викликом є брак спеціалізованих кадрів. У зв'язку з тим, що розробка Windows-застосунків потребує специфічних знань і навичок, підприємства стикаються з труднощами у пошуку відповідних кандидатів. Це змушує компанії шукати альтернативні шляхи, такі як стажування, програми менторства та партнерства з навчальними закладами для підготовки нових кадрів.

Висновки до розділу

У розділі виявлено, що кадрове забезпечення розробників Windows-застосунків з графічним інтерфейсом користувача є складовою частиною успіху в розробці програмного забезпечення. Якісні кадри, що володіють необхідними навичками та знаннями, можуть суттєво вплинути на кінцевий продукт, підвищуючи його якість та задоволеність користувачів. Інвестиції в навчання, розвиток та підтримку розробників є важливими для компаній, які прагнуть залишатися конкурентоспроможними в умовах швидко змінюваного ринку технологій. Тому кадрове забезпечення має бути пріоритетом для кожної організації, що займається розробкою програмного забезпечення.

РОЗДІЛ 4 МЕТОДИКА ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ ДЛЯ ВИКЛАДАННЯ ОСВІТНЬОГО МОДУЛЯ «РЕАЛІЗАЦІЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА НА ПЛАТФОРМІ .NET» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ. ДИДАКТИЧНИЙ ПРОЕКТ КОНСУЛЬТАТИВНОГО ЗАНЯТТЯ З ТЕМИ «ЗБІРКИ. РЕФЛЕКСІЯ. ДИНАМІЧНЕ ЗАВАНТАЖЕННЯ ЗБІРОК. ДОСЛІДЖЕННЯ ТИПУ» ДИСЦИПЛІНИ «ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ» ДЛЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ СПЕЦІАЛЬНОСТІ 015 ПРОФЕСІЙНА ОСВІТА (ЦИФРОВІ ТЕХНОЛОГІЇ)

4.1 Вихідні дані до проекту

Навчальний заклад: Бахмутський навчально-науковий професійно-педагогічний інститут Харківського національного університету імені В.Н. Каразіна;

галузь знань: 01 Освіта / Педагогіка;

спеціальність: 015 Професійна освіта (Цифрові технології);

рівень вищої освіти: перший (бакалаврський);

освітній ступінь: бакалавр;

дисципліна: «Об'єктно-орієнтоване програмування»;

тема: «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу».

Дисципліна містить такі характеристики, як:

кількість кредитів – 8.

Загальна кількість годин для вивчення дисципліни – 240 навчальних години з яких 150 годин самостійної роботи (30 годин курсова робота та 120 годин підготовка до аудиторних занять, контрольних заходів) та 90 годин аудиторної (30 години лекційних занять, 60 годин лабораторної роботи) для денної форми навчання;

Загальна кількість годин для вивчення дисципліни – 240 навчальних години з яких 216 годин самостійної роботи (30 годин курсова робота та 186 годин підготовка до аудиторних занять, контрольних заходів) та 24 годин аудиторної (12 годин лекційних занять та 12 годин лабораторної роботи) для заочної форми навчання;

Співвідношення кількості годин аудиторних занять до самостійної і індивідуальної роботи становить:

- для денної та заочної форми навчання – 90/150;
- для заочної форми навчання – 24/216.

Дисципліна «Об’єктно-орієнтоване програмування» викладається у 3-му та 4-му семестрі 2-го року професійної підготовки бакалаврів для денної та заочної форм навчання.

Форми контролю: залік, екзамен.

Великий обсяг навчального матеріалу, обширні, складні цілі навчання та великий відсоток часу, що відведено на самостійну роботу, обумовлюють необхідність у проведенні консультативних занять для уточнення та пояснення навчального матеріалу з дисципліни «Об’єктно-орієнтоване програмування».

4.2 Проектування цілей консультативного заняття

Визначення навчальних цілей заняття є одним з головних питань, від вирішення якого залежить методична організація заняття, вибір його форм, методів, засобів навчання та контролю. В цьому сенсі навчальні цілі є системоутворюючим фактором, бо, відображуючи кінцеві необхідні результати досягнень майбутніх фахівців, чітко детермінують принципи побудови системи навчання по всіх основних її параметрах.

Тому методична підготовка майбутніх фахівців передбачає перш за все оволодіння вміннями диференційовано визначати навчальні цілі, відповідно до певних рівнів професійної підготовки або рівнів засвоєння.

Проектування цілей консультативного заняття представлені у табл. 4.1 [51].

Таблиця 4.1

Цілі консультативного заняття

Цілі консультативного заняття	Цілі формування різних рівнів засвоєння навчального матеріалу	Умови досягнення	Результат у вигляді дій здобувачів освіти
1	2	3	4
1	З переліку визначень впізнавати основні поняття теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» такі, як метод Object.GetType, метод GetType, Module.GetTypes, GetElementType, GetInterfaces, GetTypeArray, GetTypeFromProgID, GetTypeFromHandle які містяться в бібліотеці C# які можна використовувати у своїх програмах.	Знати визначення понять «Рефлексія» «Стандартна бібліотека мови C#», C# колекція та «Середовище .NET Framework», знання сутності поняття «Конструктор».	Правильно названі з переліку основні поняття теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» такі, як метод Object.GetType, GetType, Module.GetTypes, GetElementType, GetInterfaces, GetTypeArray, GetTypeFromProgID, GetTypeFromHandle які містяться в бібліотеці C# які можна використовувати у своїх програмах.
2	Уміти характеризувати основні елементи, які включає збірка, рефлексі та динамічне завантаження збірок, формувати модулі або файли ресурсів, таких як растрові зображення і рядки, класифікувати товари, уміти складати загальний список товарів.	Виконання дій першого рівня: правильно названі з переліку основні поняття теми «Збірки. Рефлексія. Динамічне завантаження збірок» такі, як методи Object.GetType, GetType, Module.GetTypes, GetElementType, GetInterfaces, GetTypeArray, GetTypeFromProgID, GetTypeFromHandle які містяться в бібліотеці C# які можна використовувати у своїх програмах.	Правильно охарактеризований логічний та зрозумілий підхід використання збірки, застосування інтерфейсу в зовнішній та внутрішній цикл, правильне оголошення інтерфейсів.

Продовження табл. 4.1

1	2	3	4
3	Уміти аналізувати нескінченний цикл та робити висновки про доцільність використання інтерфейсу як посильний тип даних.	Виконання дій першого і другого рівнів.	Правильно проаналізований нескінченний цикл та зроблені висновки, щодо доцільність використання інтерфейсу як посильний тип даних.
4	Проектувати, розробляти та аналізувати алгоритми розв'язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та обчислюваних функцій.	Виконання дій першого, другого і третього рівнів.	Правильно розроблена програма з використанням інтерфейсу.

Таким чином, були розроблені цілі консультативного заняття з теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» дисципліни «Об'єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

4.3 Перелік джерел інформації

Майбутній фахівець має вміти самостійно користуватися джерелами інформації.

Наведемо перелік джерел інформації для підготовки здобувачів вищої освіти до консультації згідно з робочою програмою дисципліни «Об'єктно-орієнтоване програмування».

Нижче представлено перелік основної та допоміжної літератури, а також інформаційні ресурси для вивчення дисципліни та підготовки до консультативного заняття.

Рекомендована література:

1. Основна (базова) література

1. Об’єктно-орієнтоване програмування: електрон. метод. посіб. до лек. та лаб. занять для студентів ф-ту математики, фізики та інформ. технол. першого (бакалавр.) рівня освіти / уклад.: А. Л. Рачинська, О. А. Недєва, К. С. Палій. – Одеса : Одес. нац. ун-т ім. І. І. Мечникова, 2024. – 338 с.

2. Коноваленко І. В., Марущак П. О. Платформа .NET та мова програмування C# 8.0 : навч. посібник. Тернопіль : ФОП Паляниця В. А., 2020. 320 с.

3. Куліков В.М., Рябцев В.В., Паршуков С.С. К85 Об’єктно-орієнтоване програмування для фахівців з кібербезпеки: навч. посіб. / ІСЗЗІ КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2023. 365 с.

4. Муляр В. П. Об’єктно-орієнтоване програмування: лабораторний практикум. Луцьк : Вежа-Друк, 2022. 112 с.

5. Основи об’єктно-орієнтованого програмування: навч. посібник / Гришанович Т. О., Глинчук Л. Я.; ВНУ імені Лесі Українки. Луцьк : ВНУ імені Лесі Українки, 2022. 120 с.

Додаткова (допоміжна) література

1. Chykunov P. Services for creating UML class diagrams // P. Chykunov, I. Chykunov/ Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VI Всеукраїнської науково-практичної інтернет-конференції (м. Харків, 06-07 грудня 2023 р.). – Харків: ННППІ УІПА, 2023. С 42-43.

2. Prokop Y.V., Trofymenko O.G., Kapustin M.M. A study of software development tools that are required in the job market in Ukraine and the world. Proceedings of the O.S. Popov ONAT. 2018. Vol. 2, pp. 101-108. DOI 10.33243/2518-7139-2018-1-2-101-108.

3. Ланевич Т. Принципи SOLID у розробці програмного забезпечення. Матеріали VII Міжнародної студентської науково-технічної конференції «Природничі та гуманітарні науки. Актуальні питання», 2024, 89-90.

4. Омельчук Л.Л. Об’єктно-орієнтоване програмування. Лабораторний практикум: навчальний посібник. – Київ, 2021. 265 с.

5. Прокоп Ю. В., Трофименко О. Г., Толокнов А. А., Дубовой Я. В. Комплексний аналіз підходів до викладання курсів CS1 і CS2 в університетах світу та України. Вісник Черкаського державного технологічного університету. Технічні науки. 2021. № 2. С. 18-30. DOI: 10.24025/2306-4412.1.2021.229502.

6. Сурков К., Книшук А., Сорокун С. Інноваційні методи навчання при вивченні об’єктно орієнтованих мов програмування для майбутніх ІТ-фахівців. Перспективи та інновації науки, 2024, 4 (38).

7. Трофименко О.Г., Савєльєва О.С., Прокоп Ю.В., Логінова Н.І., Дика А.І. Soft skills для розробників програмного забезпечення. Кібербезпека: освіта, наука, техніка. 2023. № 3(19). С. 6-19. URL:

8. Чикунов П.О. Модернізація лабораторного практикуму з дисципліни «Об’єктно-орієнтоване програмування» / П.О. Чикунов, І.П. Чикунов // Актуальні тенденції розвитку освіти, науки та технологій : Матеріали VII Міжнародної науково-практичної конференції (м. Бахмут, м. Харків, 15 травня 2024 р.). : у 3-х ч. Бахмут – Харків: ННПІІ УІПА, 2024. Ч 2. С. 97-99.

9. Чикунов П.О. Розробка лабораторного практикуму «Об’єктно-орієнтоване програмування» / П.О. Чикунов, В.С. Лизунов // Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VI Всеукраїнської НПІК. – Харків: ННПІІ УІПА, 2023. С. 38-39.

Інформаційні ресурси

1. http://cyber.onua.edu.ua/ – робочі матеріали з курсу

2. C# Basics for Beginners: Learn C# Fundamentals by Coding – Онлайн-курс Udemy. URL: https://www.udemy.com/course/csharp-tutorial-for-beginners/

3. Learn Parallel Programming with C# and .NET – Онлайн-курс Udemy. URL: https://www.udemy.com/course/parallel-dotnet/

4. C# Intermediate: Classes, Interfaces and OOP – Онлайн-курс Udemy.

URL: https://www.udemy.com/course/csharp-intermediate-classes-interfaces-and-oop/

5. Overview of classes, structs, and records in C# – Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/

6. Основи програмування на C# – Онлайн-курс Prometheus. URL: https://courses.prometheus.org.ua/courses/Microsoft/CS201/2016_T1/about

7. C++ Programming Essentials – Онлайн-курс edX. URL: https://www.edx.org/professional-certificate/ibm-c-programming-essentials

8. Object Oriented Development using C# – Онлайн-курс Coursera. URL: https://www.coursera.org/learn/oo-development-using-c-sharp

9. Object-Oriented programming (C#). URL: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop

10. C# OOP. URL: https://www.w3schools.com/cs/cs_oop.php

11. Learn Object-Oriented Programming with C#. URL: https://www.tutorialsteacher.com/csharp/oop

12. A Complete Guide To Object Oriented Programming In C#. URL: https://www.c-sharpcorner.com/UploadFile/84c85b/object-oriented-programming-using-C-Sharp-net/

Основним джерелом для підготовки здобувачів вищої освіти до консультації є конспект лекцій, розроблений викладачем, оскільки він є найбільш адаптованим до робочої програми дисципліни «Об’єктно-орієнтоване програмування».

4.4 Визначення найбільш складних для розуміння та засвоєння питань

Визначимо найбільш складні для розуміння та засвоєння здобувачами вищої освіти питання теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» дисципліни «Об’єктно-орієнтоване програмування» та сформулюємо можливі відповіді на них (табл. 4.2) [51].

Таблиця 4.2

Обрання питань для консультування та формулювання відповідей на можливі питання

Теми (або тема) дисципліни	Зміст програми за кожною темою	Найбільш складні питання за темами (темою)	Відповіді на питання
1	2	3	4
«Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу».	1. Рефлексія C#, збірки. 2. Колекції об'єктів. 3. Графічний інтерфейс користувача. 4. Windows Forms 5. Лаба з мультиплатформ них рішень.	1.Охарактеризуйте екземпляра класу Type?	1. Примірник типу Type може бути представлений будь-яким із наступних типів: класи; - розмірні типи (Value types); - масиви; - інтерфейси; - покажчики; - індексатори (нумератори).
		2. Назвіть набір властивостей для запиту інформації за типом, який надає клас Type?	2. Клас Type надає великий набір властивостей для запиту інформації за типом, включаючи такі: - *FullName* - повертає ім'я типу; - *IsClass* - визначає, чи є тип класом; - *IsAbstract* - визначає, чи є тип абстрактним класом; - *IsNestedPublic* - визначає, чи є тип вкладеним та загальнодоступним; - *IsPublic* - визначає, чи є даний тип загальнодоступним; - *IsNotPublic* - визначає, чи є даний тип захищеним; - *IsSealed* - визначає, чи є тип кінцевим; - *IsArray* - визначає, чи представляє зазначений тип масив; - *GUID* - повертає значення типу GUID, асоційоване з цим типом (таке значення зберігається у реєстрі Windows); - *Module* - повертає модуль (DLL) у якому оголошено цей тип;

Продовження табл. 4.2

1	2	3	4
		3. Що таке збірка?	3. Збірка (assembly) – базовий будівельний блок програми в .NET Framework і є двійковим файлом, який містить логічну групу з одного або декількох керованих модулів або файлів ресурсів, таких як растрові зображення і рядки..

Отже, на даному етапі ми визначили найбільш складні для розуміння та засвоєння питання з теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» дисципліни «Об'єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

4.5 Вибір дидактичних методів активізації навчальної діяльності студентів на консультації

Оберемо методи активізації навчальної діяльності здобувачів вищої освіти на консультації (табл. 4.3) [52].

Таблиця 4.3

Методи активізації навчальної діяльності студентів на консультації

Дидактичні методи	Реалізація методів при проведенні консультаційного заняття
1	2
Методи підвищення наочності	Використання інтерактивної дошки та мультимедійного проектора для демонстрації слайдів з теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу», та використання плакатів «Колекції об'єктів. Графічний інтерфейс користувача Windows Forms».
Мотиваційні методи	Для реалізації мотивації використаємо: тип: внутрішня мотивація; вид: вступна мотивація; метод: мотивуючий вступ; прийом: віднесення до особистості. Повідомлення важливості вивчення даної теми: «Вивчення сьогоднішньої теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» для вас, як майбутніх фахівців з цифрових технологій, є достатньо актуальним. Відповідно до вашої майбутньої професійної діяльності знання

Продовження табл. 4.3

1	2
	цього навчального матеріалу знадобляться вам для програмування на підприємстві, а саме це дозволить правильно використовувати певні набори операторів і призначення керуючих конструкцій розгалуження. Від того, наскільки успішно ви справитеся з даним завданням, вже працюючи на підприємстві, залежить його благополуччя, а вам дозволить рости у якості висококваліфікованого програміста в області програмування та стверджуватимуть вас як висококваліфікованого фахівця».
Проблемні методи	Використання проблемного питання. Проблемне питання: «Скількома способами можна отримати екземпляр класу Type?»

Комунікативні методи	Імітація ситуацій з реального життя.
	Усі наші знання необхідні для подальшої вашої професійної діяльності, тому що використання збірки, рефлексії дає можливість об'єднати деяку послідовність окремих операцій. Ось, наприклад, як можна за допомогою цих властивостей отримати опис сімейства типу:
	`string GetTypeDescription(Type type)`
	`{`
	`  return type.IsClass ? "class"`
	`: type.IsInterface ? "interface"`
	`: type.IsArray ? "array"`
	`: type.IsEnum ? "enum"`
	`: type.IsValueType ? "struct"`
	`: "";`
	`}`

Таким чином, ми обрали методи активізації навчальної діяльності на консультації.

4.6 Вибір способів організації консультативного заняття

Вибір способів організації консультативного заняття здійснюється з урахуванням даних, наведених в таблиці 4.4 [50].

Таблиця 4.4

Варіанти організації консультативного заняття

№ варіанта	Етапи організації заняття	Характеристика варіанта
1	2	3
1	- вступне слово лектора; - відповіді на питання здобувачів освіти і обговорення їх; - заключне слово викладача.	Недоліком цього варіанту проведення лекції-консультації є відсутність послідовності, системи в питаннях, на які доводиться викладачу давати відповіді. Питання поступають хаотично, що знижує якість консультації.

Продовження табл. 4.4

1	2	3
2	- збір питань в письмовій формі до лекції, їх систематизація; - відповіді на питання, що поступили; - відповіді на додаткові питання; - обмін думками; - висновки.	Цей варіант, на відміну від попереднього, дозволяє викладачу групувати відповіді, що сприяє кращому засвоєнню навчального матеріалу здобувачами освіти.
3	- видача завдань на самостійне вивчення матеріалу теми; - підготовка питань лектору; - відповіді і їх обговорення.	В цьому випадку консультування грає функцію додаткового інформування зі складних питань і пояснення незрозумілого навчального матеріалу.
4	- повідомлення теми; - консультування декількома фахівцями в певній області науки і техніка з актуальних питань науки і нової техніки.	Цей варіант лекції-консультації проводиться, як правило, зі спеціальних дисциплін, іноді для цієї мети використовуються наукові семінари. Такі заняття дають можливість зіставити думки різних учених на одну і ту ж проблему і є чудовою школою ведення дискусії.

Згідно з представленою таблицею обираємо 1 варіант організації консультативного заняття, на якому викладач пояснює питання, які здалися незрозумілими здобувачам вищої освіти**.**

4.7 Розробка сценарію проведення консультативного заняття

Наведемо розробку сценарію проведення консультативного заняття у відповідності до обраного варіанту його організації (табл. 4.5) [52].

Таблиця 4.5

Сценарій консультативного заняття

Етапи проведення консультативного заняття	Дії викладача	Дії здобувачів освіти
1	2	3
Організаційний момент	Вітання, фіксація відсутніх, перевірка зовнішньої обстановки в аудиторії	Вітання викладача. Підтвердження присутності у момент переклички, настрій на здійснення навчальної діяльності

Продовження табл. 4.5

1	2	3
Повідомлення теми і мети уроку	Повідомлення теми заняття «Вивчення сьогоднішньої теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу», формулювання його цілей: сформувати уміння давати визначення поняттю «Збірка», «Рефлексія», правильно використовувати динамічне завантаження збірок.	Фіксація теми, сприйняття цілей, представлення результатів засвоєння матеріалу теми даного заняття
Мотивація мети	Повідомлення важливості вивчення даної теми: «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» для вас, як майбутніх програмістів, є достатньо актуальним. Відповідно до вашої майбутньої професійної діяльності знання цього навчального матеріалу знадобляться вам для програмування на підприємстві, а саме це дозволить правильно використовувати певні набори операторів і призначення керуючих конструкцій розгалуження. Все це є одним з основних завдань вашої майбутньої діяльності. Від того, наскільки успішно ви справитеся з даним завданням, вже працюючи на підприємстві, залежить його благополуччя та стверджуватимуть вас як висококваліфікованого фахівця».	Сприйняття важливості і актуальності вивчення теми, прояв інтересу до неї.
Актуалізація знань	Викладач проводить фронтальне усне опитування з метою перевірки базових знань: 1. Надайте відповідь поняттю метадані? 2. Що таке рефлексія?	Здобувачі беруть участь у опитуванні та відповідають на питання: 1. Метадані – це інформація про модуль, що отримується під час виконання програми. До такої інформації належать дані про тип. У разі неправильної вказівки імені типу виникає виняток. Тому вказівку типу слід укладати в блок try-catch. 2. Рефлексія – це отримання інформації про типи під час виконання програми. Наприклад, можна отримати список всіх класів та інтерфейсів збірки, список елементів кожного класу, список параметрів кожного методу.

Продовження табл. 4.5

1	2	3
Формування ООД	Викладач проводить консультацію згідно плану, за допомогою методу - пояснення: 1. Рефлексія C#, збірки. 2. Колекції об'єктів. 3. Графічний інтерфейс користувача. 4. Windows Forms 5. Лаба з мультиплатформних рішень.	Слухають пояснення, конспектують.
Визначення проблемних моментів під час вивчення питань теми та формування ВД	Викладач запитує здобувачів освіти про недоречності, які виникли у них під час самостійного вивчення теми. Викладач відповідає на поставлені запитання: 1. Збірки. 2. Рефлексія. 3. Динамічне завантаження збірок. 4. Дослідження типу.	Здобувачі освіти запитують: 1. Поясніть, будь-ласка, головні преваги збірки? 2.Які недоліки та переваги рефлексії?
Підведення підсумків	Викладач підводить підсумки проведення консультації: «Сьогодні ми розглянули незрозумілі вам питання теми для самостійного вивчення. Зараз перевіримо як ви засвоїли.	Здобувачі освіти слухають, відповідають: «Реалізація графічного інтерфейсу на платформі .NET”. Здобувачі освіти прощаються.

Отже, на цьому етапі ми розробили сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

На заключному етапі представлено контурний конспект з теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта (Цифрові технології).

За обсягом інформації, що представляється, конспекти діляться на повні і контурні (опорні), а за способом подання інформації — на плани-конспекти і конспекти-схеми. План-конспект стисло представляє зміст кожного з пунктів плану. Конспект-схема – це ієрархія понять теми, впорядкованих згідно плану і доповнених основними відомостями.

У повному конспекті міститься, переважно, вся нова основна інформація. У контурному (опорному) ж конспекті містяться тільки ключові

положення нової основної інформації, виражені за допомогою таблиць, графіків, абревіатури, різного роду позначень, акцентів.

Контурний конспект заняття з теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології) представлено у Додатку.

Висновки до розділу

У четвертому розділі розроблено дидактичний проект консультативного заняття з теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності здобувачів освіти на консультації. Здійснено вибір способів організації консультативного заняття.

Розроблено сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

Проаналізовано джерела інформації для підготовки здобувачів освіти до консультації згідно з робочою програмою дисципліни «Об’єктно-орієнтоване програмування». Подано список використаних джерел та відповідні посилання.

ВИСНОВКИ

У кваліфікаційній роботі теоретично обґрунтована та перевірена методика професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти та виділено пріоритетні напрямки удосконалення професійних компетентностей.

З'ясовано, що професійна підготовка фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET» у закладах вищої освіти є актуальною у професійній педагогіці.

Проаналізовано ступінь актуальності проблеми професійної підготовки фахівців з цифрових технологій для викладання освітнього модулю «Реалізація графічного інтерфейсу користувача на платформі .NET».

У роботі виконано огляд засобів розробки застосунків з графічним інтерфейсом для платформи .NET. Виконана постановка нового лабораторного практикуму для вивчення теми «Реалізація графічного інтерфейсу користувача на платформі .NET», зокрема поставлено такі роботи:

1. реалізація GUI засобами Windows Forms для роботи з колекціями;

2 реалізація GUI засобами Windows Forms для перегляду вмісту збірки;

3. реалізація GUI засобами Windows Presentation Foundation;

4. реалізація GUI засобами Universal Windows Platform.

Розроблено завдання лабораторних робіт, наведено приклади виконання робіт та оформлення звітів з описом етапів побудови застосунків, програмним кодом мовою C#, прикладами виконання програми у середовищі Visual Studio.

Встановлено, що кадрове забезпечення розробників Windows-застосунків з графічним інтерфейсом користувача є складовою частиною успіху в розробці програмного забезпечення. Якісні кадри, що володіють необхідними навичками та знаннями, можуть суттєво вплинути на кінцевий продукт, підвищуючи його якість та задоволеність користувачів. Тому кадрове

забезпечення має бути пріоритетом для кожної організації, що займається розробкою програмного забезпечення.

Розроблено дидактичний проект консультативного заняття з теми «Збірки. Рефлексія. Динамічне завантаження збірок. Дослідження типу» дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти спеціальності 015 Професійна освіта. (Цифрові технології).

Сформульовано цілі консультативного заняття. Обрано методи активізації навчальної діяльності здобувачів освіти на консультації. Здійснено вибір способів організації консультативного заняття.

Розроблено сценарій проведення консультативного заняття у відповідності до обраного варіанту його організації.

Проаналізовано джерела інформації для підготовки здобувачів освіти до консультації згідно з робочою програмою дисципліни «Об’єктно-орієнтоване програмування». Подано список використаних джерел та відповідні посилання.

Апробація результатів дослідження виконана під час виконання лабораторного практикуму з дисципліни «Програмна інженерія» бакалаврами спеціальності 015 Професійна освіта (Цифрові технології) БННППІ ХНУ ім. В.Н. Каразіна у жовтні 2024 р.

За основними результатами дослідження виконана публікація тез доповідей на конференції:

– Чикунов П.О. Постановка лабораторного практикуму «Реалізація графічного інтерфейсу користувача на платформі .NET» / П.О. Чикунов, В.В. Павловський // Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали VII Всеукраїнської науково-практичної інтернет-конференції (м. Харків, 05-06 грудня 2024 р.). – Харків: БННППІ УІПА ХНУ ім. Каразіна, 2024.

СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. Ласкаво просимо до .NET. URL: https://learn.microsoft.com/dotnet/welcome

2. Graphical user interface. URL: https://microsoft.fandom.com/wiki/Graphical_user_interface

3. Microsoft Forms. URL: https://forms.office.com/

4. What's a Universal Windows Platform (UWP) app? URL: https://learn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide

5. Windows Presentation Foundation. URL: https://visualstudio.microsoft.com/vs/features/wpf/

6. Остапенко В.Р. Розробка інтерфейсу та елементів програми керування роботом-гідом у крос-платформному фреймворку Xamarin. 2021. Master's Thesis. Національний університет «Запорізька політехніка».

7. Золотухіна О.А., Бажан Ю.П. Аналіз інтеграції штучного інтелекту у процес автоматизованого тестування UX/UI дизайну. Інформаційні технології та безпека, 2023, 52.

8. Chemerys H. et al. Fundamentals of UX/UI design in professional preparation of the future bachelor of computer science. In: AIP Conference Proceedings. AIP Publishing, 2022.

9. Coursera - Навчайтеся без обмежень. URL: https://www.coursera.org/

10. Pluralsight empowers you to onboard quic. URL: https://www.pluralsight.com/

11. Платформи – Udemy. URL: https://mooc4ua.online/platforms/2

12. Hunt A., Thomas D. The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley Professional, 2019.

13. Коноваленко І.В., Марущак П.О. Платформа .NET та мова програмування C# : навч. посіб. Тернопіль: ФОП Паляниця В. А., 2020. 320 с.

14. Самчинська Я.Б., Вінник М.О. Просування й розповсюдження

педагогічного програмного забезпечення на ринку України. Інформаційні технології в освіті, 2013. № 15. – С. 210-220.

15. Сурков К., Книшук А., Сорокун С. Інноваційні методи навчання при вивченні об’єктно орієнтованих мов програмування для майбутніх ІТ-фахівців. Перспективи та інновації науки. 2024. № 4 (38).

16. Трофименко О.Г., Савєльєва О.С., Прокоп Ю.В., Логінова Н.І., Дика А.І. Soft skills для розробників програмного забезпечення. Кібербезпека: освіта, наука, техніка. 2023. № 3(19). С. 6-19.

17. Чикунов П.О., Лизунов В.С. Розробка лабораторного практикуму «Об’єктно-орієнтоване програмування». Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: матер. VI Всеукр. наук.-практ. інтернет-конф. (м. Харків, 06-07 грудня 2023 р.). Харків: ННППІ УІПА, 2023. С. 38-39.

18. Чикунов П.О., Чикунов І.П. Модернізація лабораторного практикуму з дисципліни «Об’єктно-орієнтоване програмування». Актуальні тенденції розвитку освіти, науки та технологій : матер. VII Міжнар. наук.-практ. конф. (Бахмут - Харків, 15 травня 2024 р.). : у 3-х ч. Ч 2. С. 97-99.

19. Чикунов П.О. Розробка програмного середовища візуалізації та аналізу графових моделей / П.О. Чикунов, М.О. Сафонов, Д.А. Неізмайлов // Матеріали ІІІ Всеукраїнської науково-практичної інтернет-конференції «Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні» (м. Бахмут, 29-30 листопада, 2020 р.) [упоряд. П.О. Чикунов] / ННППІ УІПА(м. Бахмут) – Бахмут: ННППІ УІПА, 2020. – С. 48-49.

20. Чикунов П.О. Розробка ігрових Android-додатків мовою Kotlin / П.О. Чикунов, В.І. Гурова // Сучасні технології в енергетиці, електромеханіці, системах управління та машинобудуванні: Матеріали V Всеукраїнської науково-практичної інтернет-конференції (м. Харків, 20-21 квітня 2023 р.) / ННППІ УІПА [упоряд. П.О. Чикунов]. – Харків: ННППІ УІПА, 2023.

21. Чикунов П.О. Рефакторинг коду при конструюванні програмного

забезпечення // Актуальні тенденції розвитку освіти, науки та технологій : Матеріали VI Міжн. наук.-практ. конф. (м. Бахмут, м. Харків, 18 травня 2023 р.). : у 2-х ч. / За заг. ред. Г. Г. Михальченко. Бахмут – Харків: ННППІ УІПА, 2023. Ч 1. – С. 98-100.

22. Чикунов П.О. Застосування Xamarin.Forms для розробки мультиплатформових рішень / Інформаційне суспільство: проблеми та перспективи : матеріали VIII Всеукр. наук.-практич. конф. (м. Одеса, 2 черв. 2023 р.) / відп. ред. Н. І. Логінова. - Одеса, 2023. - 146 с. - Режим доступу : https://doi.org/10.32837/11300.25263

23. Поліщук А.М., Ніколюк П.К. Технологія програмування та основні етапи її розвитку. Комп'ютерні технології обробки даних, 2022, 89-91.

24. Бородкіна І. Інженерія програмного забезпечення: навчальний посібник. Центр учбової літератури, 2021. – 204 с.

25. Трофименко О.Г., Прокоп Ю.В., Задерейко О.В. Алгоритмізація та програмування : навч.-метод. посібник. Одеса: Фенікс, 2020. 310 с. URL: http://dspace.onua.edu.ua/handle/11300/12345.

26. С++. Алгоритмізація та програмування : підручник / О.Г. Трофименко, Ю.В. Прокоп, Н.І. Логінова, О.В. Задерейко. 2-ге вид. перероб. і доповн. Одеса : Фенікс, 2019. 477 с.

27. Stroustrup B. The C++ Programming Language, 4th Edition, 2013.

28. Злобін, Г.Г. Основи алгоритмізації та програмування мовою Сі : підручник. Львів : ЛНУ ім. І. Франка, 2022. 97 с. ISBN: 978-617-7593-75-0

29. Ярошко С. А., Ярошко О.С. Методи розробки алгоритмів. Програмування мовою С++: навч. посіб. Львів: ЛНУ імені Івана Франка, 2022. 248 с. URL: https://lnuittutor.github.io/

30. Рудий Т. В., Паранчук Я. С., Сеник В. В. Алгоритмізація та програмування. Ч. 1. Структурне програмування: навч. посіб. Львів: ЛДУВС, 2023. 240 с. http://surl.li/yegqcz

31. Прокоп Ю. В., Трофименко О. Г., Толокнов А. А., Дубовой Я. В. Комплексний аналіз підходів до викладання курсів CS1 і CS2 в університетах

світу та України. Вісник Черкаського державного технологічного університету. Технічні науки. 2021. № 2. С. 18-30. https://doi.org/10.24025/2306-4412.1.2021.229502.

32. Прокоп Ю.В., Трофименко О.Г., Задерейко О.В. Аналіз підходів у викладанні початкового курсу програмування в університетах. Системні технології. 2021. № 4 (135). С. 73-84. https://doi.org/10.34185/1562-9945-4-135-2021-08.

33. Трофименко О. Г., Савєльєва О. С., Прокоп Ю. В., Логінова Н.І., Дика А. І. Soft skills для розробників програмного забезпечення. Кібербезпека: освіта, наука, техніка. 2023. № 3(19). С. 6-19. https://doi.org/10.28925/2663-4023.2023.19.619.

34. Бандоріна Л.М., Климкович Т.О., Удачина К.О. Основи алгоритмізації та програмування : навч. посібник. УДУНТ, 2022. 158 с. http://surl.li/yjcgnv

35. Антонюк Л.Л. Компетентністний підхід у вищій освіті: світовий досвід навч. пос. Київ : КНЕУ, 2016.66 с.

36. Професійна педагогіка : Підручник / Авт. : О. В. Грабовський, Л. В.Коломієць, О. С. Савельева, А. В.Семенова, В. Ф.Яні; за заг. ред. А. В.Семенової. – Одеса : Бондаренко М. О., 2020. – 575 с.

37. Професійна педагогіка: навч. посібник для вищих навч. закладів/ В. І. Жигірь, О. Чернєга ; за ред. М. В. Вачевського. - Київ: К.: Кондор, 2016. – 336 с. 978-966-351-359-1. Код 233704.

38. Зайченко І. В. Теорія і методика професійного навчання: навч. посібник. 2-е вид., доповн. і переробл. К.: Видавництво Ліра-К, 2016. 580 с.

39. Методика формування пошуково-дослідницьких умінь майбутніх інженерів-педагогів у процесі професійної підготовки: колективна монографія / В.В.Кулешова, В.В.Мальована. – Артемівськ: ННППІ УІПА, 2012. – 264 с. (власний внесок: Р1; Р2 с.86-92; Р3; 9,8 д.а.).

40. Формування професійної компетентності викладачів технічних дисциплін: колективна монографія / В.В.Кулешова, В.В.Мальована,

Ю.С.Бобрикова. – Х., 2020. – 206 с. (власний внесок: Р1 с.8-94; Р2 с.95-100; Р3с.146-159; 6,5 д.а.).

41. Кулешова В.В., Мальована В.В. Особливості особистості викладача технічних дисциплін у вищих навчальних закладах/ Проблеми інженерно-педагогічної освіти. Збірник наукових праць. №50-51 Харків: УІПА,– 2016 р., – С.322-329

42. Кулешова В.В., Мальована В.В. Формування професійних методичних умінь у майбутніх інженерів-педагогів економічного профілю / Міжнародний науковий журнал «ІНТЕРНАУКА». №7 (29) Київ:– 2017 р., – С.26-29

43. Кулешова В.В. Формування креативної компетентності майбутніх інженерів у процесі професійної підготовки / Проблеми інженерно-педагогічної освіти. Збірник наукових праць. № 58 Харків: УІПА,– 2018 р., – С.21-26

44. Олійник В. В. Відкрита післядипломна педагогічна освіта: нові моделі та форми професійного розвитку / Освіта дорослих у перспективі змін: інновації, технології, прогнози: колективна монографія / За ред.. А. Василюк, А. Стоговського. – Ніжин: Видавець ПП Лисенко М.М., 2017. – 248 с. С. 93–108.

45. Теорія та методика викладання фахових дисциплін у ЗВО: навчально- методичний посібник / укладач І. В. Казанжи – Миколаїв : СПД Румянцева, 2018. – 154 с.

46. Ортинський В. Л. Педагогіка вищої школи : навч. посіб. / Ортинський В. Л. – Центр учбової літератури, 2017. – 472 с

47. Професійна освіта України на шляху до євроінтеграції (1992–2017) / науков. ред. Н.Г. Ничкало; упорядники: Л.В. Горбань, В.П. Тименко. – К.: ДП «Інформ.-аналіт. агенство», 2018. – 358 с.

48. Сисоєва С.О. Теорія і практика вищої освіти: навч. посібник / С.О. Сисоєва, І.В. Соколова. – К., 2016. – 338 с. – Режим доступу: http://lib.iitta.gov.ua/711948/1/Sysoieva_Socolova_2016_pos..pdf

49. Теорія і практика вищої професійної освіти в Україні : навч. посіб. для магістрантів зі спеціальності 011 «Освітні, педагогічні науки» / [авт.-укл.: Т.О.Дороніна]. – Кривий Ріг : КДПУ, 2018. – 250 с.

50. Методика професійного навчання: метод. вказ. по виконанню курсової роботи для здобувачів освіти освітнього ступеня «бакалавр» денної та заочної форми навч. інженерно-педагогічних спеціальностей / ННППІ Укр. інж.-пед. акад. ; упоряд. : В. В. Кулешова, В.В. Мальована, Ю.С. Бобрикова ; за заг. ред. д-ра пед. наук, проф.В. В. Кулешової. – Бахмут, 2022. – 92 с.

51. Методика професійного навчання : конспект лекцій для здобувачів вищої освіти ОС «бакалавр» денної та заоч. форм здобуття освіти спец. 015 Проф. освіта (за спеціалізаціями). Ч. 1 / О. Е. Коваленко, Н. О. Брюханова, Н. В. Корольова; Укр. інж.-пед. акад., Каф. педагогіки, методики та менеджменту освіти. - Харків: УІПА, 2020. - 200 с.

52. Методика професійного навчання: конспект лекцій для здобувачів вищої освіти ОС «бакалавр» денної та заоч. форм здобуття освіти спец. 015 Проф. освіта (за спеціалізаціями). Ч. 2/ О. Е. Коваленко, Н. О. Брюханова, Н. В. Корольова; Укр. інж.-пед. акад., Каф. педагогіки, методики та менеджменту освіти. - Харків: УІПА, 2020. - 180 с.

53. Коваленко О. Е., Брюханова Н. О., Корольова Н.В. Методика професійного навчання: дидактичне проектування: Підручник для студентів інженерно-педагогічних спеціальностей. – Харків: УІПА, 2019. – 204 с.

54. Коваленко О. Е., Брюханова Н. О., Корольова Н.В. Методика професійного навчання: основні технології навчання: Підручник для студентів інженерно-педагогічних спеціальностей. – Харків: УІПА, 2019. – 174 с.

55. Методика професійного навчання: дидактичне проектування: конспект лекцій для студ. денної та заоч. форм навч. спец. 015.06 "Проф. освіта. Електроніка, радіотехн. та телекомунікації" освітнього рівня "бакалавр"/ Н. Г. Кошелева; Укр. інж.-пед. акад. (Бахмут), ННППІ. - Бахмут: [б. в.], 2017. - 100 с.

56. Методика професійного навчання: основні технології навчання:

конспект лекцій для студ. денної та заоч. форм навч. спец. 015.06 "Проф. освіта. Електроніка, радіотехн. та телекомунікації" освітнього рівня "бакалавр"/ Н. Г. Кошелева; Укр. інж.-пед. акад. (Бахмут), ННППІ. - Бахмут: [б. в.], 2017. - 101 с.

57. Теорія і методика професійного навчання : навч. посібник. – 2-е вид., доповн. і переробл. / І.В.Зайченко. – К.: Видавництво Ліра-К, 2016. – 580 с.

58. Методика професійного навчання: навч. посібник для вищих навч. закладів інж.-пед. спец. для традиційної та дистанційної форм навчання. Ч. 1: Дидактичне проектування/ О. Е. Коваленко, Н.О. Брюханова, Н.В. Корольова; Укр. інж.- пед. академія. - 2-ге вид., перероб. та доп.. - Х.: ФОП Шевченко С. О., 2010. - 264 с.

ДОДАТОК А КОНТУРНИЙ КОНСПЕКТ НА ТЕМУ «ЗБІРКИ. РЕФЛЕКСІЯ. ДИНАМІЧНЕ ЗАВАНТАЖЕННЯ ЗБІРОК. ДОСЛІДЖЕННЯ ТИПУ»

Збірка (assembly) – базовий будівельний блок програми в .NET Framework і є двійковим файлом, який містить логічну групу з одного або декількох керованих модулів або файлів ресурсів, таких як растрові зображення і рядки. Збірка містить у собі:

- номер версії;
- метадані;
- інструкції на мові IL.

Коли компілятор платформи .NET створює виконуваним додатком «.exe» або динамічний модуль «.dll», їхній вміст є збіркою. Збірка – це повністю самодостатній і, швидше, логічний, ніж фізичний елемент. Це означає, що він може бути збережений більш ніж в одному файлі (хоча динамічні збірки зберігаються в пам'яті, а зовсім не в файлах). Якщо збірка зберігається у більш ніж одному файлі, то повинні бути один головний файл, що містить точку входу та описує інші файли.

Одна й таж структура збірки використовується як для виконуваного коду, так й для динамічних модулів. Єдина відмінність полягає в тому, що збірка містить головну точку входу програм, тоді як бібліотечна збірка – ні.

Схема отримання та використання збірки показана на рисунку 1.

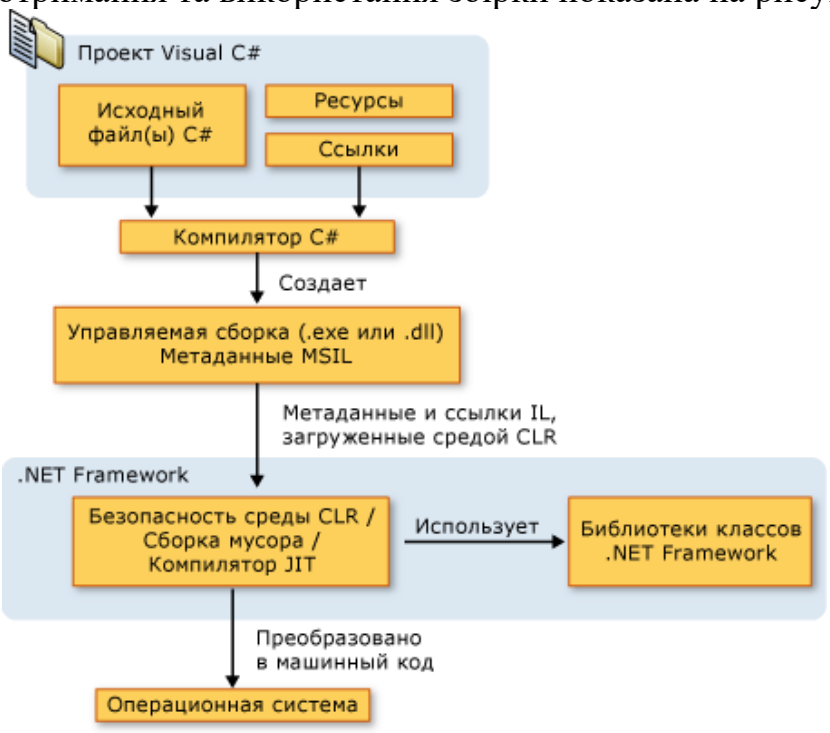

Рисунок 1 - Схема отримання та використання збірки

Рефлексія – це отримання інформації про типи під час виконання програми. Наприклад, можна отримати список всіх класів та інтерфейсів збірки, список елементів кожного класу, список параметрів кожного методу.

Виходячи з концепції уніфікації системи типів C# кожен тип в середовищі .NET пов'язаний зі спеціальним об'єктом-типом. Клас Type простору імен System грає ключову роль системі роботи з метаданими. Будь-яка діяльність із використання інформації про тип пов'язана із застосуванням цього класу.

Примірник класу Type дозволяє отримати повну інформацію про тип: інформацію про його методи, властивості, вкладені типи, інформацію про збірка та модуль, що містять даний тип, повне ім'я типу та багато іншого. Більше того, за допомогою класу Type можна динамічно створювати екземпляри описуваного ним типу (класу), а також виконувати методи та працювати з властивостями отриманого об'єкта.

Робота з типом починається з одержання відповідного екземпляра класу Type. Примірник типу Type може бути представлений будь-яким із наступних типів:

- класи;
- розмірні типи (Value types);
- масиви;
- інтерфейси;
- покажчики;
- індексатори (нумератори).

Посилання на об'єкт Type, що асоціюється з типом, може бути отримано одним з наступних способів:

- метод Object.GetType повертає об'єкт Type, який представляє тип заданого екземпляра об'єкта;
- статичний метод GetType повертає об'єкт Type, який представляє тип, вказаний у вигляді свого повного імені;
- методи Module.GetTypes, Module.GetType і Module.FindTypes повертають об'єкт Type, який представляє типи, визначені в модулі;
- метод GetTypes дозволяє отримати масив об'єктів для всіх загальнодоступних та захищених типів, визначених у модулі;
- метод FindInterfaces повертає фільтрований список інтерфейсів типів, які підтримуються даним типом;
- метод GetElementType повертає об'єкт Type, який представляє елемент;
- методи GetInterfaces та GetInterface повертають об'єкт Type, який представляє інтерфейс, що підтримується типом;
- метод GetTypeArray повертає масив об'єктів Type, які репрезентують типи, задані за допомогою набору об'єктів;
- методи GetTypeFromProgID та GetTypeFromCLSID повертають об'єкт Type;
- метод GetTypeFromHandle повертає об'єкт Type, який вказується за допомогою дескриптора (метод надається для сумісності);

– оператор typeof отримує об'єкт Type для зазначеного типу.

Динамічне завантаження збірок. Програма може завантажувати інші збірки з:

1. Каталогу програми (private-збірка).
2. Global Assembly Cache (GAC) (збірка, що розділяються).
3. Конкретний файл на диску.

Перші два способи ідентифікують збірку на ім'я.

Повне ім'я збірки містить, крім власне імені, версію збірки, інформацію про локалізації (Culture) та відкритий ключ (PublicKeyToken). Приклад імені:

```
System.Drawing, Version = 1.0.3300.0, Culture = neutral,
PublicKeyToken = b03f5f7f11d50a3a
```

Повне ім'я однозначно ідентифікує збірку і його можна завантажити:

```
Assembly asm = Assembly.Load("System.Drawing, Version=1.0.3300.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a");
```

Ім'я збірки може бути як повним, так і частковим. Часткове ім'я збірки дозволяє використовувати спеціальний механізм підтримки версій збірки. Метод LoadWithPartialName класу Assembly завантажує збірку за частковою інформацією неї.

```
Assembly asm = Assembly.LoadWithPartialName("System.Drawing");
```

Динамічне завантаження збірок з каталогу програми не вимагає вказівки повного імені збірки, досить короткого імені.

```
Assembly asm = Assembly.Load("MyAssembly"); // завантаження private-збірки
```

Функції Assembly.Load і Assembly.LoadWithPartialName при виборі відповідної збірки насамперед переглядають каталог програми, віддаючи перевагу private-збіркам.

Динамічна завантаження збірки повним шляхом до файлу дозволяє завантажити будь-яку збірку в системі:

```
Assembly a = Assembly.LoadFrom("D:\WINNT\Microsoft.NET\Framework\" +
    "v1.0.3705\System.Drawing.dll");
```

Однак цей спосіб недостатньо гнучкий і навряд чи варто широко його застосовувати.

Динамічне завантаження типів. Тепер, коли збірка завантажена, можна отримати інформацію про тип. Для цього необхідно використовувати так зване "кваліфіковане ім'я типу" (Assembly Qualified Type Name). Кваліфіковане ім'я типу складається з двох частин: повного імені типу та повного або часткового імені збірка. Для отримання опису метаданих певного типу його кваліфіковане ім'я передається у статичний метод GetType класу Type. У разі успіху, цей метод повертає екземпляр класу Type.

```
Assembly a = Assembly.LoadWithPartialName("System.Drawing");
string strAssemblyQualifiedTypeName = "System.Drawing.Rectangle, " +
a.FullName;
Type type = Type.GetType(strAssemblyQualifiedTypeName);
```

В даному випадку завантаження типу проведено у три етапи. Спочатку завантажена збірка, потім отримано повне ім'я і, лише потім, отримано об'єкт Type. Ці етапи можна поєднати. Якщо ви знаєте повне ім'я збірка, можна

використовувати його для збірка кваліфікованого імені типу, яке можна передати безпосередньо методу Type.GetType().

```
Type type = Type.GetType("System.Drawing.Rectangle"
+ ", System.Drawing" + ", Version=1.0.3300.0"
+ ", Culture=neutral"
+ ", PublicKeyToken=b03f5f7f11d50a3a");
```

Дослідження типу. Маючи доступ до об'єкту Type, можна починати досліджувати структуру типу, що він описує (перебирати поля, методи, події, властивості, вкладені типи).

Клас Type містить більше трьох десятків методів, що дозволяють отримувати інформацію про описаний тип. Так можна дізнатися про будь-яку властивість типу або наявність (і значення) певного атрибуту, з'ясувати належність типу до деякого сімейства.

Методи, що дозволяють отримувати інформацію про тип, наведено у таблиці 1.

Ось, наприклад, як можна за допомогою цих властивостей отримати опис сімейства типу:

```
string GetTypeDescription(Type type)
{
    return type.IsClass ? "class"
: type.IsInterface ? "interface"
: type.IsArray ? "array"
: type.IsEnum ? "enum"
: type.IsValueType ? "struct"
: "";
}
```

Таблиця 1 – Методи, що дозволяють отримувати інформацію про тип

IsAbstract	Чи є тип абстрактним класом чи інтерфейсом
IsClass	Чи клас це? (тобто не value Type і не інтерфейс)
IsSealed	Чи може мати тип спадкоємців?
IsInterface	Це інтерфейс? (Зверніть увагу, інтерфейс - це не клас!)
IsCOMObject	А чи не COM-об'єкт ховається під виглядом класу .NET?
IsValueType	Це значення типу (тип-значення)?
IsEnum	Це перелік?
IsPrimitive	Чи не є тип примітивним (тобто одним з bool, byte, sbyte, short, ushort, int, uint, long, ulong, char, double, float)?
IsArray	Чи є тип масивом. Тип елементів масиву можна отримати за допомогою ElementType.
IsPointer	Чи є тип посиланням? Тип об'єкта, що вказується, можна отримати за допомогою властивості ElementType.
IsByRef	А чи не тип це параметра, що передається за посиланням? Якщо так, то тип об'єкта, що передається, можна отримати за допомогою властивості ElementType.
HasElementType	HasElementType = IsArray || IsPointer || IsByRef. Тобто. Тип не є самостійним типом, лише " похідним від " ElementType.

ДОДАТОК Б ЛІСТИНГИ ПРОГРАМ

Головна форма додатку «Form1.cs» лабораторної роботи №1

```
using oop11;

namespace WinForms
{
    public partial class Form1 : Form
    {
        // Змінні для зберігання колонок DataGridView
        private DataGridViewColumn dataGridViewColumn1 = null;
        private DataGridViewColumn dataGridViewColumn2 = null;
        private DataGridViewColumn dataGridViewColumn3 = null;
        private DataGridViewColumn dataGridViewColumn4 = null;
        private DataGridViewColumn dataGridViewColumn5 = null;

        // Ініціалізація таблиці
        private void initDataGridView()
        {
            // Очищення джерела даних та додавання колонок у таблицю
            dataGridView1.DataSource = null;
            dataGridView1.Columns.Add(getDataGridViewColumn1());
            dataGridView1.Columns.Add(getDataGridViewColumn2());
            dataGridView1.Columns.Add(getDataGridViewColumn3());
            dataGridView1.Columns.Add(getDataGridViewColumn4());
            dataGridView1.Columns.Add(getDataGridViewColumn5());
            dataGridView1.AutoResizeColumns(); // Автоматична зміна розміру колонок
        }

        // Динамічне створення першої колонки у таблиці
        private DataGridViewColumn getDataGridViewColumn1()
        {
            if (dataGridViewColumn1 == null)
            {
                dataGridViewColumn1 = new DataGridViewTextBoxColumn();
                dataGridViewColumn1.Name = "";
                dataGridViewColumn1.HeaderText = "Shelf";
                dataGridViewColumn1.ValueType = typeof(string);
                // Задаємо ширину колонки
                dataGridViewColumn1.Width = dataGridView1.Width / 3;
            }
            return dataGridViewColumn1;
        }

        // Динамічне створення другої колонки у таблиці
        private DataGridViewColumn getDataGridViewColumn2()
        {
            if (dataGridViewColumn2 == null)
            {
                dataGridViewColumn2 = new DataGridViewTextBoxColumn();
                dataGridViewColumn2.Name = "";
                dataGridViewColumn2.HeaderText = "Product";
                dataGridViewColumn2.ValueType = typeof(string);
                dataGridViewColumn2.Width = dataGridView1.Width / 3;
            }
            return dataGridViewColumn2;
        }

        // Динамічне створення третьої колонки у таблиці
        private DataGridViewColumn getDataGridViewColumn3()
        {
            if (dataGridViewColumn3 == null)
```

```
    {
        dataGridViewColumn3 = new DataGridViewTextBoxColumn();
        dataGridViewColumn3.Name = "";
        dataGridViewColumn3.HeaderText = "Quantity";
        dataGridViewColumn3.ValueType = typeof(string);
        dataGridViewColumn3.Width = dataGridView1.Width / 3;
    }
    return dataGridViewColumn3;
}

// Динамічне створення четвертої колонки у таблиці
private DataGridViewColumn getDataGridViewColumn4()
{
    if (dataGridViewColumn4 == null)
    {
        dataGridViewColumn4 = new DataGridViewTextBoxColumn();
        dataGridViewColumn4.Name = "";
        dataGridViewColumn4.HeaderText = "Year";
        dataGridViewColumn4.ValueType = typeof(string);
        dataGridViewColumn4.Width = dataGridView1.Width / 3;
    }
    return dataGridViewColumn4;
}

// Динамічне створення п'ятої колонки у таблиці
private DataGridViewColumn getDataGridViewColumn5()
{
    if (dataGridViewColumn5 == null)
    {
        dataGridViewColumn5 = new DataGridViewTextBoxColumn();
        dataGridViewColumn5.Name = "";
        dataGridViewColumn5.HeaderText = "Price";
        dataGridViewColumn5.ValueType = typeof(string);
        dataGridViewColumn5.Width = dataGridView1.Width / 3;
    }
    return dataGridViewColumn5;
}

// Колекція об'єктів MediaMarket
private IList<MediaMarket> mediaMarketList = new List<MediaMarket>();

// Додавання продукту до колекції
private void addMediaMarket(string shelf, string product, string quantity,
                            string price, string year)
{
    MediaMarket mM = new MediaMarket(shelf, product, quantity, price, year);
    mediaMarketList.Add(mM);

    // Очищення текстових полів після додавання
    textBox1.Text = "";  textBox2.Text = ""; textBox3.Text = "";
    textBox4.Text = "";  textBox5.Text = "";
    showListInGrid(); // Оновлення відображення списку
}

// Видалення продукту з колекції за індексом
private void deleteMediaMarket(int elementIndex)
{
    mediaMarketList.RemoveAt(elementIndex);
    showListInGrid(); // Оновлення відображення списку
}

// Відображення колекції у таблиці
private void showListInGrid()
{
    dataGridView1.Rows.Clear(); // Очищення таблиці
```

```
            // Додавання кожного об'єкта з колекції у таблицю
            foreach (MediaMarket st in mediaMarketList)
            {
                DataGridViewRow row = new DataGridViewRow();
                DataGridViewTextBoxCell cell1 = new DataGridViewTextBoxCell();
                DataGridViewTextBoxCell cell2 = new DataGridViewTextBoxCell();
                DataGridViewTextBoxCell cell3 = new DataGridViewTextBoxCell();
                DataGridViewTextBoxCell cell4 = new DataGridViewTextBoxCell();
                DataGridViewTextBoxCell cell5 = new DataGridViewTextBoxCell();

                cell1.ValueType = typeof(string); cell1.Value = st.getShelf();
                cell2.ValueType = typeof(string); cell2.Value = st.getProduct();
                cell3.ValueType = typeof(string); cell3.Value = st.getQuantity();
                cell4.ValueType = typeof(string); cell4.Value = st.getPrice();
                cell5.ValueType = typeof(string); cell5.Value = st.getQuantity();

                row.Cells.Add(cell1); row.Cells.Add(cell2); row.Cells.Add(cell3);
                row.Cells.Add(cell4); row.Cells.Add(cell5);

                dataGridView1.Rows.Add(row); // Додавання рядка до таблиці
            }
        }

        // Обробник кнопки для додавання нового продукту
        private void button1_Click_1(object sender, EventArgs e)
        {
            addMediaMarket(textBox1.Text, textBox2.Text, textBox3.Text,
textBox4.Text, textBox5.Text);
        }

        // Обробник для видалення обраного продукту з таблиці
        private void видалитиToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            int selectedRow = dataGridView1.SelectedCells[0].RowIndex;
            DialogResult dr = MessageBox.Show("Видалити продукт?", "",
MessageBoxButtons.YesNo);

            if (dr == DialogResult.Yes)
            {
                try
                {
                    deleteMediaMarket(selectedRow);
                }
                catch (Exception)
                {
                    // Обробка винятків при видаленні
                }
            }
        }

        // Конструктор форми
        public Form1()
        {
            InitializeComponent();
            initDataGridView(); // Ініціалізація таблиці при запуску форми
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Метод для обробки події завантаження форми (поки не реалізований)
        }
    }
}
```

Файл з класом «MediaMarket.cs» лабораторної роботи №2

```
namespace WinForms
{
    // Клас MediaMarket для представлення товару на полицях магазину
    internal class MediaMarket
    {
        // Поле для зберігання інформації про полицю
        private string shelf;
        // Поле для зберігання інформації про продукт
        private string product;
        // Поле для зберігання кількості товару
        private string quantity;
        // Поле для зберігання ціни товару
        private string price;
        // Поле для зберігання року виробництва товару
        private string year;

        // Конструктор класу MediaMarket, що ініціалізує усі поля
        public MediaMarket(string shelf, string product, string quantity,
                           string price, string year)
        {
            this.shelf = shelf; this.product = product; this.quantity = quantity;
             this.price = price; this.year = year;
        }
        // Метод для отримання значення ціни товару
        public string getPrice()
        {
            return this.price;
        }
        // Метод для встановлення значення ціни товару
        public void setPrice(string price)
        {
            this.price = price;
        }
        // Метод для отримання року виробництва товару
        public string getYear()
        {
            return this.year;
        }
        // Метод для встановлення року виробництва товару
        public void setYear(string year)
        {
            this.year = year;
        }
        // Метод для отримання інформації про полицю
        public string getShelf()
        {
            return this.shelf;
        }
        // Метод для отримання інформації про продукт
        public string getProduct()
        {
            return this.product;
        }
        // Метод для отримання кількості товару
        public string getQuantity()
        {
            return this.quantity;
        }
        // Метод для встановлення інформації про полицю
        public void setShelf(string shelf)
        {
            this.shelf = shelf;
        }
```

```
        // Метод для встановлення інформації про продукт
        public void setProduct(string product)
        {
            this.product = product;
        }
        // Метод для встановлення кількості товару
        public void setQuantity(string quantity)
        {
            this.quantity = quantity;
        }
    }
}
```

ДОДАТОК В ПУБЛІКАЦІЇ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

ПОСТАНОВКА ЛАБОРАТОРНОГО ПРАКТИКУМУ «РЕАЛІЗАЦІЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА НА ПЛАТФОРМІ .NET»

Чикунов П.О., к.т.н., доц.
Павловський В.В., здобувач вищої освіти другого рівня
Бахмутський навчально-науковий професійно-педагогічний інститут ХНУ імені В. Н. Каразіна

Розробка програмного забезпечення для Windows має давню історію, яка зазнала значних змін та покращень протягом багатьох років. У сучасному світі інформаційних технологій роль розробників програмного забезпечення, особливо тих, хто спеціалізується на створенні Windows-застосунків з графічним інтерфейсом користувача (GUI), стає дедалі важливішою. Інтерфейс користувача – це перше, що бачить користувач, тому його дизайн має бути не лише естетичним, а й функціональним. Успішний розробник повинен мати глибоке розуміння принципів дизайну, знання сучасних мов програмування, а також вміти адаптувати застосунки під різні пристрої та екрани.

Платформа .NET призначена для розробки різноманітних застосунків для платформ Windows, macOS, Linux, Android, iOS [1]. Платформа .NET підтримує різні типи застосунків:

– десктопні застосунки з графічним інтерфейсом користувача [2],
– вебдодатки та мобільні застосунки,
– хмарні застосунки,
– інтернет речей та штучний інтелект.

Виконано огляд засобів розробки застосунків з графічним інтерфейсом для платформи .NET. Встановлено, що розробники застосовують три основні платформи для розробки застосунків з графічним інтерфейсом користувача для ОС Windows – це Windows Forms, Windows Presentation Foundation та Universal Windows Platform. Кожна з них має свої переваги та недоліки, тому розробники вибирають платформу залежно від специфіки майбутнього проєкту.

Windows Forms – це графічний інтерфейс користувача з відкритим кодом для .NET, стандартний набір бібліотек базових класів і API, що спрощують поширені завдання програми. За допомогою такого середовища розробки, як Visual Studio, можна створювати інтелектуальні клієнтські програми Windows Forms, які відображають інформацію, вимагають введення від користувачів та обмінюються даними з віддаленими комп'ютерами.

Windows Presentation Foundation (WPF) – це платформа інтерфейсу користувача для створення клієнтських програм для настільних систем. Платформа розробки WPF підтримує широкий набір функцій розробки програм, у тому числі: Модель програми, Ресурси, Елементи керування, Графіка, Макет, Прив'язка даних, Документи, Безпека. Це частина бібліотек, що належать до ASP.NET Core або Windows Forms. WPF використовує XAML для надання декларативної моделі програмування програм.

Universal Windows Platform (UWP) надає розробникам платформу для будь-яких пристроїв під керуванням Windows 10 або 11, будь-то настільний комп'ютер, ігрова приставка Xbox, гарнітура змішаної реальності або смартфон.

Вона дозволяє використовувати декларативний підхід для опису інтерфейсу на мові XAML, що спрощує розробку і покращує читабельність коду. UWP підтримує сучасний дизайн та інтеграцію з Windows Store, що дозволяє розробникам розповсюджувати свої програми більш широко. UWP забезпечує високий рівень безпеки завдяки обмеженому доступу до системних ресурсів.

Виконана постановка нового лабораторного практикуму для вивчення теми «Реалізація графічного інтерфейсу користувача на платформі .NET»:

1. реалізація графічного інтерфейсу користувача засобами Windows Forms для роботи з колекціями;

2 реалізація графічного інтерфейсу користувача засобами Windows Forms для перегляду вмісту збірки;

3. реалізація графічного інтерфейсу користувача засобами Windows Presentation Foundation;

4. реалізація графічного інтерфейсу користувача засобами Universal Windows Platform.

Розроблено завдання лабораторних робіт, наведено приклади виконання робіт та оформлення звітів, що опис етапів побудови застосунків у середовищі Visual Studio (рис. 1), програмний код на мові C#, приклад виконання програми.

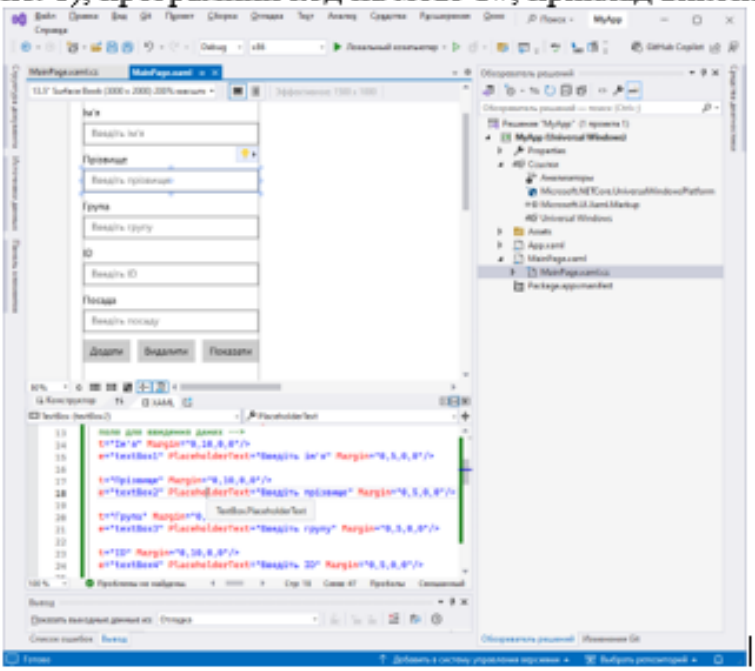

Рисунок 1 – Приклад розробки графічного інтерфейсу засобами UWP

Список використаних джерел

1. Ласкаво просимо до .NET. URL: https://learn.microsoft.com/dotnet/welcome. (дата звернення 12.11.2024).

2. Graphical user interface. URL: https://microsoft.fandom.com/wiki/Graphical user interface. (дата звернення 12.11.2024).

3. Чикунов П.О. Модернізація лабораторного практикуму з дисципліни «Об'єктно-орієнтоване програмування» / П.О. Чикунов, І.П. Чикунов // *Актуальні тенденції розвитку освіти, науки та технологій* : Матеріали VII Міжн. наук.-практ. конф. (м. Бахмут, м. Харків, 15 травня 2024 р.). : у 3-х ч. / За заг. ред. Г. Г. Михальченко. Бахмут – Харків: ННІПІ УІПА, 2024. Ч 2. С. 97-99.