

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система адміністратора освітніх курсів
на мові програмування C# з використанням СКБД MySQL»

на здобуття освітнього ступеня магістра
зі спеціальності F3 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

*Кваліфікаційна робота містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів мають посилання
на відповідне джерело*

(підпис)

Олександр МАКАРУК
(Ім'я, ПРІЗВИЩЕ здобувача)

Виконав:
здобувач вищої освіти
група КНДМ-62

Олександр МАКАРУК

Керівник:
науковий ступінь,
вчене звання

Олена ШИКУЛА
д.ф.-м.н., професор

Рецензент:
науковий ступінь,
вчене звання

(Ім'я, ПРІЗВИЩЕ)

Київ 2025

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність F3 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
« _____ » _____ 2025 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

_____ Макаруку Олександру Сергійовичу

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Інформаційна система адміністратора освітніх курсів на мові програмування C# з використанням СКБД MySQL

керівник кваліфікаційної роботи Олена ШИКУЛА д.ф.-м.н., професор,

(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467

2. Строк подання кваліфікаційної роботи «26» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, мова програмування C#, СКБД MySQL, середовище програмування Microsoft Visual Studio.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження ролі і завдання освітніх курсів в навчальному процесі та огляд існуючих аналогів систем адміністрування, виявлення їх переваг та недоліків.
Розробка інформаційної системи адміністратора освітніх курсів.

5. Перелік графічного матеріалу: *презентація*

1. Огляд існуючих систем для адміністрування.
2. Створення проекту в Microsoft Visual Studio.
3. Блок-схеми алгоритмів.
4. Схеми бази даних.
5. Екранні форми інформаційної системи для адміністратора освітніх курсів.
6. Фрагменти програмного коду.

6. Дата видачі завдання «16» вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.09-27.09.25	виконано
2	Огляд існуючих систем адміністрування	28.09-11.10.25	виконано
3	Визначення функціоналу та основних характеристик інформаційної системи	12.10-25.10.25	виконано
4	Вибір інструментів для розробки інформаційної системи	26.10-04.11.25	виконано
5	Створення бази даних для розроблюваної інформаційної системи	05.11-12.11.25	виконано
6	Розробка інформаційної системи адміністратора освітніх курсів	12.11-02.12.25	виконано
7	Оформлення роботи: вступ, висновки, реферат	02.12-08.12.25	виконано
8	Розробка демонстраційних матеріалів	09.12-13.12.25	виконано

Здобувач вищої освіти

(підпис)

Олександр МАКАРУК

(Ім'я, ПРІЗВИЩЕ)

Керівник

кваліфікаційної роботи

(підпис)

Олена ШИКУЛА

(Ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 79 стор., 5 табл., 55 рис., 63 джерела.

Наукове завдання – проведення огляду та аналіз програмних продуктів для адміністраторів, вибір оптимальних інструментів для створення інформаційної системи, створення бази даних системи адміністрування освітніми курсами, розробка зручного та інтуїтивно зрозумілого інтерфейсу користувача, розробка інформаційної системи для полегшення роботи адміністратора освітніх курсів, тестування роботи інформаційної системи.

Мета роботи – розробка інформаційної системи адміністратора освітніх курсів для полегшення його роботи. Інформаційна система повинна мати зручний та інтуїтивно зрозумілий інтерфейс і надавати можливість адміністратору переглядати та редагувати інформацію.

Об'єкт дослідження – інформаційна система адміністратора освітніх курсів.

Предмет дослідження – теоретичні та практичні проблеми створення та реалізації інформаційної системи адміністратора освітніх курсів.

Короткий зміст роботи:

Досліджено роль і завдання освітніх курсів в навчальному процесі, зроблено огляд існуючих аналогів систем адміністрування, виявлено їх переваги та недоліки.

За допомогою об'єктно-орієнтованої мови програмування C# з використанням СКБД MySQL в середовищі Microsoft Visual Studio було створено інформаційну систему адміністратора освітніх курсів із зручним та зрозумілим користувачеві інтерфейсом. Було протестовано роботу інформаційної системи.

Ключові слова: інформаційна система, адміністратор освітніх курсів, об'єктно-орієнтована мова програмування C#, СКБД MySQL, середовище Microsoft Visual Studio

ABSTRACT

Text part of the master's qualification work: 79 pages, 5 tables, 55 pictures, 63 sources.

The scientific task is to conduct a review and analysis of software products for administrators, select the optimal tools for creating an information system, create a database for the educational course administration system, develop a convenient and intuitive user interface, and develop an information system to facilitate the work of the educational course administrator, test the operation of the information system.

The purpose of the work is to develop an information system for the educational course administrator to facilitate his work. The information system should have a convenient and intuitive interface and allow the administrator to view and edit information.

The object of research is the information system for the educational course administrator.

The subject of research is the theoretical and practical problems of creating and implementing an information system for the educational course administrator.

Summary of the work: The role and tasks of educational courses in the educational process were studied, an overview of existing analogues of administration systems was made, their advantages and disadvantages were identified. Using the object-oriented programming language C# using the MySQL database in the Microsoft Visual Studio environment, an information system for the administrator of educational courses was created with a convenient and user-friendly interface. The operation of the information system was tested.

Keywords: testing systems, *information* system, educational course administrator, object-oriented programming language C#, MySQL database, Microsoft Visual Studio environment

ЗМІСТ

ВСТУП.....	9
1 ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ.....	12
1.1 Освітні курси як чинник оптимізації навчання.....	12
1.2 Аналіз існуючих аналогів інформаційних систем адміністратора	13
2 ШЛЯХИ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	26
2.1 Обґрунтування вибору мови програмування.....	26
2.2 Вибір середовища програмування.....	30
2.2.1 Обґрунтування вибору середовища програмування.....	30
2.2.2 Робота в середовищі програмування	31
2.3 Вибір СКБД	40
2.3.1 Проблема зберігання даних. Бази даних	40
2.3.2 Обґрунтування вибору СКБД.....	45
2.4 Моделювання інформаційної системи.....	50
2.4.1 Загальні вимоги до роботи інформаційної системи	50
2.4.2 Розробка алгоритмів створення інформаційної системи.....	53
3 ОПИС ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	58
АДМІНІСТРАТОРА ОСВІТНІХ КУРСОВ	58
3.1 Вимоги до програмного продукту.....	58
3.2 Структура бази даних	61
3.2 Створення макетів інформаційної системи.....	66
1.4 Екранні форми інформаційної систкми.....	68
3.5 Тестування операційної системи.....	77
ВИСНОВКИ.....	81
ПЕРЕЛІК ПОСИЛАНЬ	83
ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ.....	88
ДОДАТОК Б. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)	122

ВСТУП

Адміністратор – це спеціаліст, діяльність якого спрямована на забезпечення якісного та результативного обслуговування клієнтів, надання консультацій з питань, що стосуються товарів чи послуг, які пропонує підприємство чи організація [1].

У період, коли українські землі перебували у складі Речі Посполитої, термін «адміністратор» мав дещо інше значення. Під ним розуміли:

- управителя різних маєтностей (земельних володінь, господарств, підприємств тощо);
- особу, яка здійснювала керівництво певною територіальною одиницею католицької громади. Зокрема, адміністратор дієцезії призначався Папою Римським тимчасово – у випадку відсутності єпископа або неможливості виконання ним своїх обов’язків.

Зі збільшенням кількості торговельних закладів, магазинів, кафе та ресторанів потреба у фахівцях такого профілю лише зростала. Сьогодні діяльність практично жодного підприємства неможливо уявити без адміністратора. Він може відповідати як за певний напрям роботи компанії, так і за організацію діяльності персоналу. Наприклад, адміністратор ресторану повинен зустрічати гостей, супроводжувати їх до столика, пропонувати вільні місця, контролювати загальний порядок у закладі та координувати роботу офіціантів.

В умовах сучасних швидких змін управління організацією стає складним процесом, який неможливо реалізувати за допомогою лише стандартних схем менеджменту. Для цього потрібен висококваліфікований спеціаліст, здатний ефективно керувати як виробничими, так і організаційними процесами, раціонально розподіляти обов’язки між співробітниками та забезпечувати стабільність і розвиток підприємства. Адміністратор повинен мати ґрунтовні знання про всі внутрішні й зовнішні аспекти функціонування компанії.

Для успішної діяльності адміністратора необхідно мати низку особистісних якостей та професійних умінь. Насамперед це відповідальність, активність, рішучість, вміння налагоджувати комунікацію, витриманість і стресостійкість.

Важливим є також уміння чітко та грамотно висловлювати власні думки, адже професія передбачає постійний контакт з людьми. Бажаною вважається й охайна, приваблива зовнішність, оскільки адміністратор часто є «обличчям» установи чи закладу.

Рівень оплати праці в цій сфері, як правило, є вищим за середній показник заробітної плати в промисловості. Однак водночас професія адміністратора має певні ризики. У процесі роботи нерідко виникають проблемні ситуації, які потребують швидкого ухвалення рішень та пошуку компромісів, що влаштовують як клієнтів, так і персонал. Саме тому вміння застосовувати знання з психології, зокрема у врегулюванні конфліктів, є суттєвою перевагою для фахівця цієї галузі.

Попри те, що професія адміністратора відносно нова, вона швидко інтегрувалася у повсякденне життя й набула широкого поширення. Сьогодні практично жоден серйозний заклад – чи то ресторан, магазин, комп'ютерна фірма, чи база відпочинку – не обходиться без адміністратора. До основних функцій спеціаліста належать організація роботи закладу, контроль за діяльністю персоналу та підтримання належного рівня сервісу.

Варто зазначити, що професія адміністратора має декілька різновидів і відноситься до базової групи адміністративно-управлінського персоналу. Робочий графік здебільшого складається позмінно, з можливістю нічних чергувань, що потребує від працівника додаткової витривалості та здатності до адаптації.

У сучасних умовах спостерігається стрімке зростання попиту на діяльність адміністратора освітніх курсів. З метою оптимізації та полегшення його роботи створюються спеціалізовані програмні продукти. Проте на сьогодні бракує програмного забезпечення, яке б одночасно поєднувало простоту використання та зрозумілий функціонал і було безпосередньо орієнтоване на адміністрування освітніх курсів. Існуючі аналоги здебільшого розроблені для потреб шкіл або закладів вищої освіти, організаційна діяльність яких суттєво відрізняється від функціонування курсів додаткової освіти. Саме тому створення програмного продукту, призначеного для спрощення роботи адміністратора освітніх курсів, є *актуальним*.

Об'єкт дослідження – інформаційна система адміністратора освітніх курсів.

Предмет дослідження – теоретичні та практичні аспекти розроблення та впровадження інформаційної системи адміністратора освітніх курсів.

Мета дослідження полягає у створенні інформаційної системи адміністратора освітніх курсів, яка забезпечуватиме ефективне виконання його професійних обов'язків. Розроблена система повинна мати зручний, інтуїтивно зрозумілий інтерфейс та надавати адміністратору можливість переглядати, редагувати і систематизувати необхідну інформацію.

Для досягнення поставленої мети необхідно вирішити наступні *завдання*:

- Провести огляд та аналіз наявних програмних продуктів, орієнтованих на адміністрування.
- Визначити оптимальні інструменти для створення інформаційної системи.
- Розробити базу даних, призначену для адміністрування освітніх курсів.
- Створити інтерфейс користувача, що характеризуватиметься простотою використання та інтуїтивною зрозумілістю.
- Реалізувати інформаційну систему, яка сприятиме полегшенню роботи адміністратора освітніх курсів.
- Протестувати роботу інформаційної системи.

Практична цінність атестаційної роботи бакалавра полягає в створенні інформаційної системи адміністратора освітніх курсів з метою полегшенні його роботи. Розроблена інформаційна система виключить рутину роботи, а також буде запобігати людським помилкам.

1 ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ

1.1 Освітні курси як чинник оптимізації навчання

Курсове навчання суттєво відрізняється як від освітнього процесу у закладах вищої освіти, так і від інших форм підготовки. Основними його ознаками є:

- інтенсивні та динамічні темпи засвоєння матеріалу;
- орієнтація на формування практичних навичок.

Досить часто короткотривалі терміни навчання викликають у частини людей певні сумніви. Це пояснюється усталеним у суспільстві уявленням, що для оволодіння новими знаннями чи вміннями потрібен тривалий і ґрунтовний освітній процес. Подібне твердження є правильним лише частково. У традиційній системі навчання, яка ґрунтується на читанні, конспектуванні та запам'ятовуванні матеріалу, дійсно витрачається значний час, а знання, що не підкріплюються практикою, з часом можуть бути втрачені.

Разом із тим практика доводить, що навчальний процес може бути значно ефективнішим і більш результативним за умови інтеграції теорії з практикою. Використання отриманих знань у реальних завданнях уже з перших занять дозволяє засвоювати матеріал швидше, розширювати обсяг засвоєної інформації та, що особливо важливо, набувати навичок її практичного застосування.

Саме тому курси поєднують у собі як передачу теоретичного матеріалу, так і його закріплення на практиці, що є ключовим чинником оптимізації та скорочення тривалості навчання без втрати його якості.

До основних обов'язків адміністратора навчального центру належать:

- приймання телефонних дзвінків та консультування потенційних слухачів із організаційних питань;
- реєстрація учасників курсів;
- організація старту навчальних груп у разі набору необхідної кількості слухачів (від 4 осіб) за одним курсом і на одну зміну.

1.2 Аналіз існуючих аналогів інформаційних систем адміністратора

Зважаючи на значну щільність робочого графіка, а також на ймовірність виникнення різноманітних непередбачуваних ситуацій, що порушують запланований порядок роботи, навіть найбільш уважний та відповідальний працівник може припуститися помилок. Водночас похибки та недоліки у діяльності адміністратора є особливо небажаними для закладу розміщення. Такі помилки здатні вкрай негативно вплинути на репутацію підприємства, показники його відвідуваності, а відтак – і на кінцевий фінансовий результат.

З метою мінімізації ризику виникнення подібних інцидентів процес управління, який здійснюється адміністратором, доцільно автоматизувати шляхом використання спеціалізованого програмного забезпечення для адміністрування.

Програма для готелів, хостелів, міні-готелів, баз відпочинку і пансіонатів – jSolutions [2]

Система управління готелем, хостелом, санаторієм, базою відпочинку та іншими закладами готельного бізнесу jSolutions забезпечує комплексний контроль усіх бізнес-процесів закладу (див. рисунок 1.1). Однією з ключових переваг програмного забезпечення для готелів jSolutions є можливість інтеграції з додатковими модулями, серед яких:

- Модуль jSolutions для автоматизації ресторану, кафе та бару – дозволяє організувати процес замовлення харчування безпосередньо в номер готелю.
- Модуль jSolutions для клубів різної спеціалізації (розважальних, спортивних, фітнес-клубів, SPA-центрів, дитячих клубів) – забезпечує автоматичне нарахування оплати за користування додатковими послугами із включенням їх до загального рахунку за проживання.

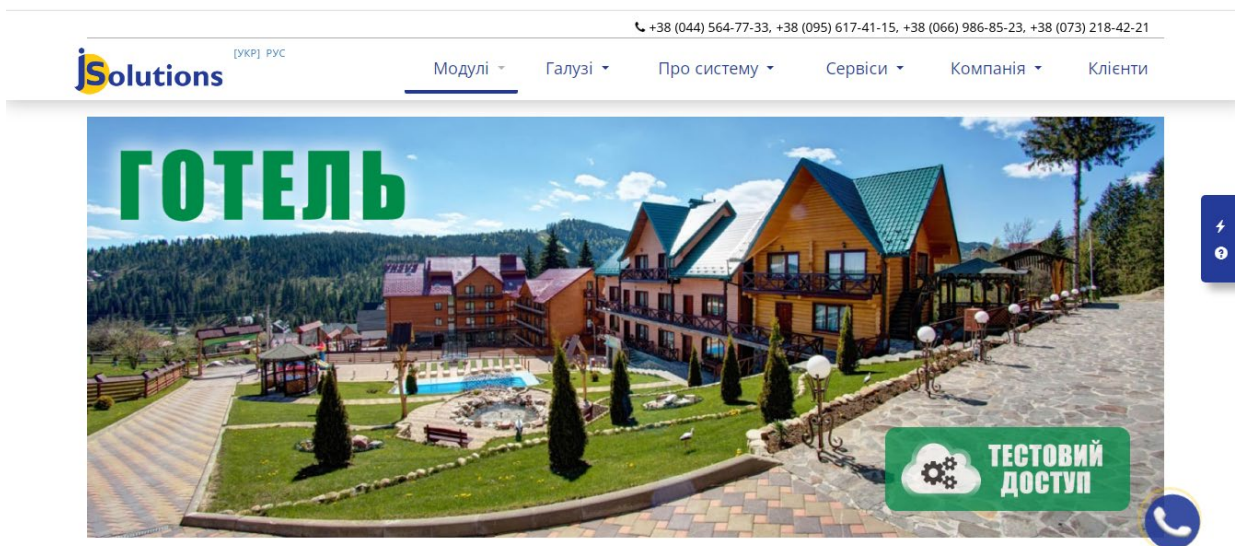


Рисунок 1.1 – Система управління jSolutions

Функціональні можливості хмарної системи jSolutions для управління готельним бізнесом:

Однією з ключових характеристик системи є управління номерним фондом. Програмне забезпечення дозволяє формувати структуру номерного фонду за різними типами, зокрема за моделлю хостелу (ліжко-місце) або класичного готелю (номер повністю). Передбачено можливість побудови схеми розташування номерів у будівлі за поверхами та приміщеннями, а також використання прикладів типових номерів. Для кожного номера користувач може додавати перелік характеристик у вигляді меблів та обладнання із зазначенням їхньої вартості. Крім того, існує функціонал щодо визначення індивідуальних параметрів для кожного номера, наприклад графіка прибирання, переліку необхідних товарів чи комплектації міні-бару.

Не менш важливим елементом є налаштування тарифів. У відповідному розділі системи передбачено можливість коригування вартості проживання залежно від сезону, кількості гостей та кількості додаткових місць. Користувач може вказувати перелік доступних схем харчування та додаткових послуг. Крім базових тарифів, система дозволяє створювати спеціальні пропозиції зі знижками, орієнтованими на певні категорії номерів або конкретні періоди проживання.

Окремий модуль відповідає за управління завантаженням номерного фонду. У розділі «Дані про місця» відображається актуальна інформація щодо наявності вільних номерів на обраний період, а також подається календар завантаженості з графічною візуалізацією у розрізі місяців. Процес бронювання включає можливість вибору тарифу, введення особистих даних гостя, зазначення інформації про номер рейсу та запланований час заїзду і виїзду. Система підтримує як поселення за попереднім бронюванням, так і без нього. Для оперативного введення персональних даних може застосовуватися сканер паспортних документів.

Також ключовим елементом системи є управління комунікаціями з клієнтами. Система забезпечує роботу з побажаннями, скаргами та зауваженнями клієнтів за допомогою механізму реклаमाцій. Гості мають можливість залишати заявки на додаткові послуги, зокрема замовлення дзвінка-будильника у визначений час, виклик таксі до моменту виїзду, замовлення сніданку в номер або користування послугами хімчистки.

Для раціональної організації роботи персоналу система передбачає планування обслуговування номерного фонду. Зокрема, можна визначати як плановий, так і фактичний час прибирання номерів із зазначенням необхідного переліку товарів для міні-бару. Це сприяє підвищенню ефективності роботи обслуговуючого персоналу та зменшує ризики виникнення недоліків у сервісі.

Система розрахована на створення звітів та аналітики. Система дозволяє формувати звіти різних типів, вести облік безготівкових платежів, здійснювати аналіз взаєморозрахунків, контролювати оплату і передоплату, виставляти рахунки та вести архів документації. Крім того, користувач може отримувати зведені звіти щодо фінансових результатів та наданих послуг, що забезпечує якісний моніторинг економічної діяльності закладу.

Особливості системи автоматизації готелів jSolutions:

- формування єдиної бази даних для будь-якої кількості готелів або корпусів;
- облік завантаження номерного фонду;
- автоматизація розрахунків з клієнтами та контрагентами;

- впровадження системи лояльності для гостей (дисконтні та подарункові картки);
- підтримка різних форм оплати: готівкових, банківськими картами, за допомогою дисконтних карток;
- створення та контроль завдань щодо прибирання приміщень;
- інтеграція з офіційним сайтом готелю та можливість роботи через web-інтерфейс;
- взаємодія з електронними замками, турнікетами, шлагбаумами та використання електронних ключів (брелоків, карток, браслетів);
- технічна підтримка користувачів на території всієї України.

Зручність роботи адміністратора. Робочий процес адміністратора організований через єдиний інтерфейс, у якому зібрано всі основні інструменти для виконання професійних обов'язків. До них належать: аналіз завантаженості номерного фонду, реєстрація гостей із внесенням їхніх даних до клієнтської бази, управління стандартними та додатковими послугами (харчування, пральня, паркування, замовлення таксі, доступ до високошвидкісного інтернету). Система дозволяє поєднувати ці послуги з основним рахунком за проживання, здійснювати облік передоплати, виконувати бронювання з урахуванням індивідуальних побажань клієнта, а також швидко аналізувати завантаженість на майбутні періоди.

Використання хмарної системи jSolutions дає змогу значно підвищити ефективність організації роботи адміністратора, спростити процес обслуговування гостей і зробити управління закладом готельного бізнесу більш прозорим та результативним.

Програмне забезпечення для ресторанного бізнесу SERVIO [3]

Програма SERVIO є комплексним програмним інструментом, призначеним для автоматизації та управління процесами у закладах громадського харчування (див. рисунок 1.2). Система забезпечує виконання широкого спектра завдань, серед яких: організація обслуговування гостей, проведення банкетів, ведення складського та управлінського обліку, формування калькуляційних і технологічних

карт, управління фінансами, організація дисконтних програм, акцій, а також контроль робочого часу персоналу.

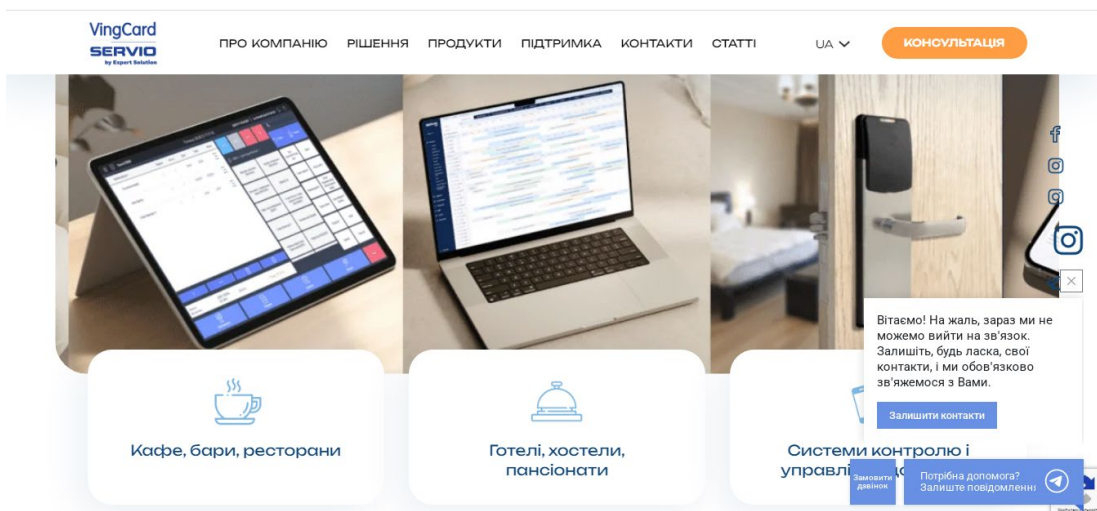


Рисунок 1.2 – Програмне забезпечення SERVIO

Функціонал SERVIO налаштовується відповідно до специфіки закладу незалежно від його формату: ресторан, кафе, бар, кав'ярня, заклад швидкого харчування або їдальня. Основною метою впровадження програми є не лише автоматизація, а й оптимізація бізнес-процесів, підвищення ефективності роботи підприємства та усунення виявлених недоліків.

Підключення системи SERVIO дозволяє:

- Інтегрувати всі бізнес-процеси закладу в єдину систему управління.
- Забезпечити оптимізацію роботи залу, бару, кухні, складу та каси.
- Контролювати всі етапи обслуговування клієнтів – від моменту прийому замовлення до формування розрахункового документа.
- Виявляти та усувати слабкі місця у роботі підприємства.
- Підвищувати продуктивність праці персоналу та здійснювати контроль за дотриманням робочої дисципліни.
- Вести облік робочого часу та впроваджувати прозору систему нарахування заробітної плати і бонусів відповідно до відпрацьованих годин.
- Використовувати механізми лояльності для клієнтів (дисконтні та бонусні програми).

- Налагоджувати комунікацію з постійними відвідувачами за допомогою електронної пошти, мобільних застосунків або інших інструментів розсилки.
- Здійснювати збір і аналіз статистичних даних про відвідувачів, зокрема щодо часу перебування у закладі та індивідуальних смакових уподобань.
- Проводити інвентаризацію та вести детальний складський облік.
- Формувати різні види звітності, необхідні для управлінських і фінансових потреб.

Використання програми SERVIO у закладах ресторанного бізнесу дозволяє досягти таких результатів:

- підвищення рівня лояльності клієнтів;
- зростання прибутковості підприємства;
- зменшення операційних витрат;
- прискорення процесів обслуговування;
- мінімізація ймовірності помилок у роботі персоналу.

Програмне забезпечення Salon Manager [4]

Програма Salon Manager призначена для комплексної автоматизації діяльності закладів індустрії краси (див. рисунок 1.3). Її функціональні можливості охоплюють ведення історії клієнтів, облік витратних матеріалів і препаратів, управління фінансами та заробітною платою персоналу, а також інші важливі операції, необхідні для ефективного функціонування салону.

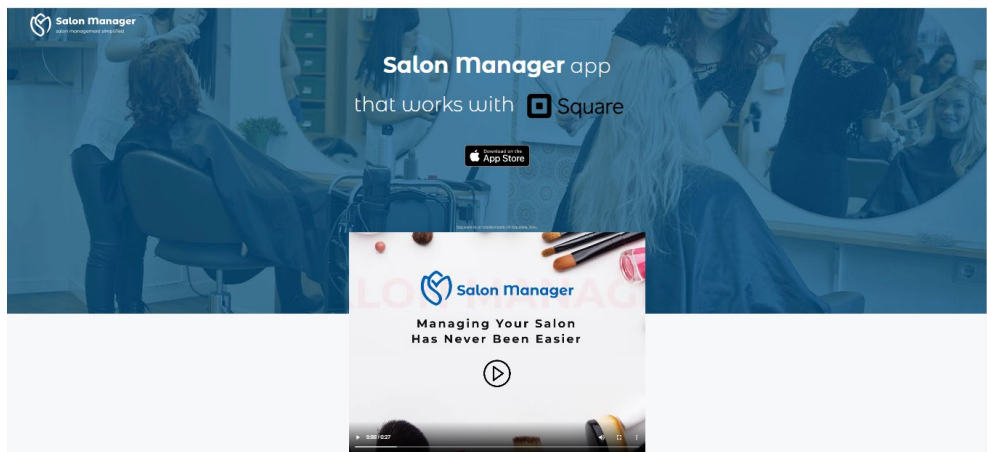


Рисунок 1.3 – Програмне забезпечення Salon Manager

До основних можливостей системи належать:

1. Управління клієнтською базою:

- ведення детальної історії клієнтів (запис на послуги, історія оплат, використані препарати, покупки, сертифікати);
- нагадування про дні народження клієнтів із можливістю надання знижок у святкові дні;
- формування програм лояльності (дисконтні картки, акції, бонусні програми, абонементи, подарунки).

2. Організація роботи салону:

- електронний журнал запису з можливістю автоматичного резервування;
- формування графіка роботи співробітників;
- управління правами доступу користувачів, ведення журналу змін і ключових операцій;
- робота з локальною мережею або через Інтернет.

3. Складський та фінансовий облік:

- контроль залишків препаратів і матеріалів;
- проведення операцій приходу, продажу, списання та внутрішнього переміщення товарів;
- формування дозамовлень на основі аналізу залишків;
- підтримка роботи зі сканером штрих-кодів;
- імпорт та експорт даних (клієнти, складська номенклатура, звіти).

4. Управління фінансами та нарахуванням заробітної плати:

- нарахування відсотків за виконані послуги та реалізацію товарів;
- встановлення фіксованих надбавок або утримань;
- можливість різних варіантів оплати клієнтських рахунків (готівка, депозит, бонуси з доплатою готівкою);
- ведення обліку боргів без обмеження їх суми;
- визначення максимально допустимого відсотка оплати бонусами.

5. Аналітика та звітність:

- формування фінансових та аналітичних звітів (оборотні відомості, валовий дохід, популярність препаратів);

- статистика щодо відвідуваності та уподобань клієнтів;
- контроль діяльності персоналу та ефективності роботи салону.

Таким чином, система «Адміністратор Салону» виступає універсальним інструментом для автоматизації бізнес-процесів у закладах індустрії краси, забезпечуючи зручність управління клієнтською базою, оптимізацію складського обліку, прозорість фінансових операцій та підвищення якості обслуговування клієнтів.

Програмне забезпечення адміністрування FossLook [5]

Для управління системою FossLook використовується спеціальна програма «FossLook Адміністратор» (див. рисунок 1.4). Вона забезпечує повний контроль за налаштуванням та функціонуванням системи документообігу і містить широкий спектр інструментів для адміністрування.



Рисунок 1.4 – Програма адміністрування FossLook

Основні можливості FossLook:

1. Управління організаційною структурою. Система дозволяє створювати та адмініструвати підрозділи, користувачів і групи користувачів. Кожному користувачеві можуть призначатися права доступу відповідно до його функціональних обов'язків.

2. Робота з документами. У FossLook можна створювати власні типи документів, організовувати їх зберігання у бібліотеках та папках, а також керувати доступом до них. Система підтримує можливість використання електронного цифрового підпису (ЕЦП) як для окремих полів документа, так і для вкладених файлів. Для цього можуть застосовуватися різні криптопровайдери, які адміністратор обирає у відповідному розділі налаштувань.

3. Контроль об'єктів та доступу. Програма відображає список усіх заблокованих об'єктів у системі. Блокування документа здійснюється під час його редагування, а адміністратор має можливість у будь-який момент розблокувати потрібний об'єкт.

4. Маршрутизація документів. У системі реалізована підтримка створення шаблонів маршрутів руху документів між користувачами. Це дозволяє як заздалегідь налаштовувати маршрути, так і створювати їх безпосередньо в процесі відправки документа. Кількість маршрутів не обмежена.

5. Керування сесіями користувачів. Адміністратор має змогу переглядати активні сесії користувачів та відключати їх від сервера за потреби.

6. Підключення зовнішніх модулів. Функціональні можливості FossLook можна розширювати за рахунок інтеграції із зовнішніми модулями.

7. Звіти, статистика та історія. Система дозволяє вести статистику роботи користувачів із документами, зокрема переглядати час роботи, кількість звернень та обсяг внесених змін. Крім того, в історії документа зберігається інформація про всі зміни: дата, автор змін та попередні і нові значення полів.

8. Поштові сервіси та мережеві налаштування. FossLook підтримує підключення до внутрішнього поштового сервера, а також інтеграцію із зовнішніми поштовими сервісами (наприклад, ukr.net, gmail.com). Для роботи через

веб-інтерфейс адміністратор може налаштувати параметри веб-сервера, зокрема порт і адресу.

Таким чином, FossLook є багатофункціональною системою електронного документообігу, що забезпечує централізоване управління користувачами, документами, маршрутами їх руху, а також підтримує інтеграцію із зовнішніми сервісами і модулями. Її використання сприяє підвищенню прозорості управлінських процесів, контролю доступу та ефективності роботи організації.

Програмне забезпечення «Ресепшен» [6]

Додаток «Ресепшен» призначений для автоматизації процесу запису клієнтів та управління їхнім обслуговуванням у різних сферах діяльності. Програму можуть використовувати медичні працівники, косметологи, стоматологи, масажисти, перукарі, майстри манікюру, а також менеджери барбершопів, ресторанів і готелів (див. рисунок 1.5). Основною метою застосунку є усунення необхідності ручного ведення записів клієнтів, що значно полегшує організацію роботи фахівців і підприємств сфери послуг.

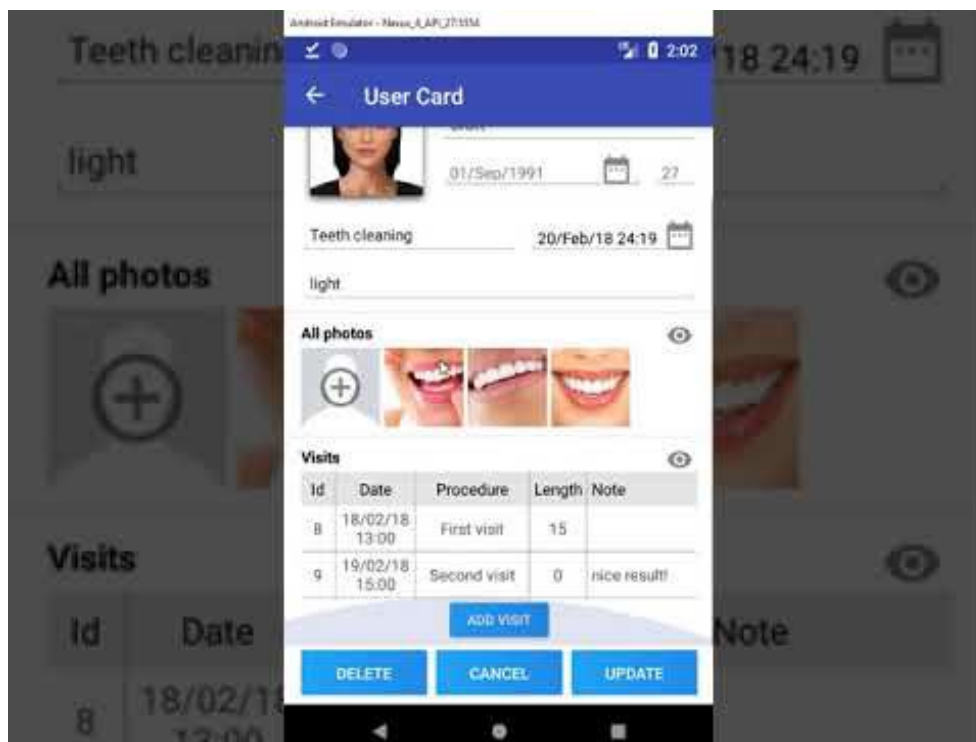


Рисунок 1.5 – Програма «Ресепшен»

До ключових можливостей системи належать:

- ведення бази клієнтів із можливістю прикріплення фотографій, що дозволяє відстежувати прогрес наданих послуг;
- організація розрахунків із клієнтами за здійснені візити та формування фінансових звітів у форматі PDF (за день, тиждень або місяць);
- швидкий пошук інформації через рядок пошуку на головній сторінці, що охоплює всі записи;
- зручне сортування клієнтів і візитів за будь-яким обраним критерієм.

Таким чином, додаток «Ресепшен» виступає універсальним інструментом для управління записами клієнтів, фінансовим обліком та організацією робочих процесів у сфері послуг. Простота у використанні та можливість роботи з мобільних пристроїв роблять його зручним рішенням як для індивідуальних фахівців, так і для малих підприємств.

Vumrix – програмне забезпечення для організації запису клієнтів [7]

Vumrix є сучасною програмою, призначеною для ведення електронного запису клієнтів (див. рисунок 1.6). Завдяки простому та інтуїтивно зрозумілому інтерфейсу, наявності сервісу для онлайн-запису, а також додатковим інструментам, програма значно підвищує зручність роботи та сприяє зростанню продуктивності спеціалістів.

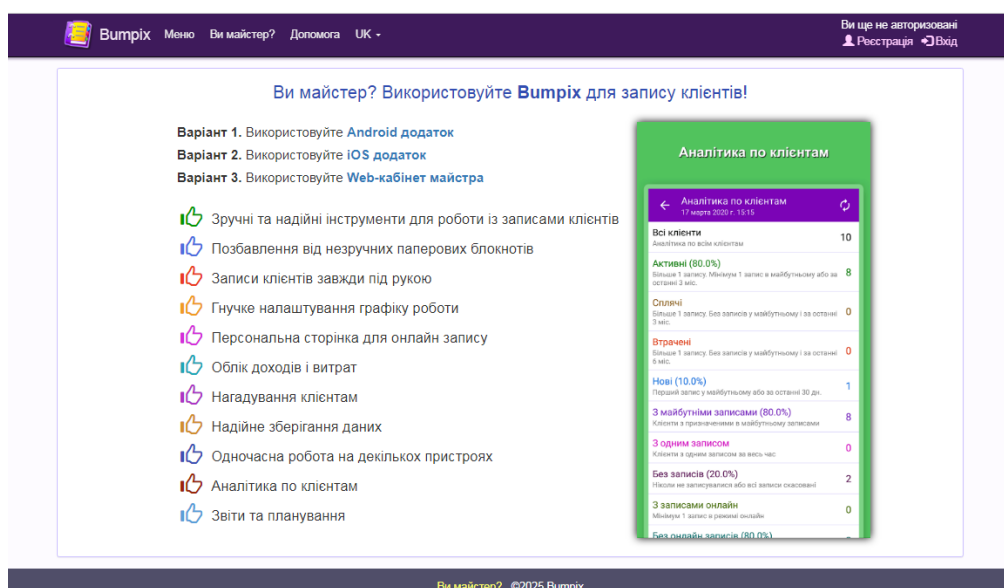


Рисунок 1.6 – Програма Vumrix

До основних функціональних можливостей програмного забезпечення належать:

- робота без обов'язкового підключення до мережі Інтернет;
- зручне та наочне відображення вільного часу для запису клієнтів;
- автоматичне надсилання нагадувань клієнтам про заплановані візити;
- ведення обліку доходів та витрат;
- підтримка роботи кількох спеціалістів в межах однієї програми;
- надання персональної сторінки для здійснення онлайн-запису;
- гнучке налаштування робочого графіка з можливістю створення індивідуального розкладу на кожен день;
- безпечне зберігання даних на сервері з подальшою можливістю їх відновлення у разі необхідності;
- синхронна робота на декількох пристроях;
- створення не лише записів клієнтів, а й особистих подій у календарі.

Програмне забезпечення Vumprіx можна застосовувати у різних сферах обслуговування. Воно підходить для організації запису клієнтів на процедури манікюру, послуги перукарів, візажистів, стилістів, бровистів, лікарів та інших спеціалістів, які надають індивідуальні послуги.

Слід зазначити, що Vumprіx створено не лише для приватних майстрів. Це також ефективний інструмент для невеликих салонів краси, оскільки він дозволяє вести базу клієнтів, організовувати робочий процес майстрів та управляти записами безпосередньо з мобільного телефона чи планшета.

Автоматизація процесів при адмініструванні готельного, ресторанного та сервісного бізнесу є важливим інструментом для підвищення ефективності роботи, мінімізації помилок персоналу та покращення якості обслуговування клієнтів. Сучасні програмні рішення (jSolutions, SERVIO, Salon Manager, FossLook, «Ресепшен», Vumprіx) забезпечують інтегрований контроль бізнес-процесів, управління клієнтською базою, фінансовим та складським обліком, комунікацією з гостями, а також створення аналітики й звітності.

Впровадження сучасного програмного забезпечення в різних сферах послуг забезпечує:

- зниження ймовірності помилок персоналу;
- оптимізацію бізнес-процесів і підвищення продуктивності;
- зручність управління фінансами, клієнтською базою та складським обліком;
- формування лояльності клієнтів і покращення якості обслуговування;
- зростання прибутковості підприємств.

Таким чином, автоматизація є не просто додатковим інструментом, а стратегічним чинником успіху підприємств готельно-ресторанної та сервісної сфери, який дозволяє їм бути конкурентоспроможними та відповідати сучасним вимогам ринку.

2 ШЛЯХИ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Обґрунтування вибору мови програмування

Сучасний розвиток інформаційних технологій визначається зростанням попиту на продуктивне, безпечне та універсальне програмне забезпечення. У цьому контексті особливу роль відіграють мови програмування, що поєднують простоту використання, гнучкість та широкі можливості для створення додатків.

Однією з таких мов є C#, створена компанією Microsoft як багатофункціональний інструмент для програмістів. Завдяки багатопарадигмальності, інтеграції з платформою .NET та підтримці сучасних технологій, C# зайняла провідні позиції у сфері програмної інженерії.

Ця мова активно застосовується у різних напрямках: від бізнес-рішень і мобільних додатків до ігор та систем машинного навчання. Це зумовлює потребу в аналізі її історії, характеристик і практичного значення.

Історія та особливості C#.

C# («сі шарп») – сучасна кросплатформна мова програмування з відкритим вихідним кодом. Вона була створена наприкінці 1990-х – на початку 2000-х років командою Microsoft під керівництвом Андерса Хейлсберга, відомого також як автор Turbo Pascal і Delphi. Основне завдання – створення мови, яка б поєднувала потужність C++ і простоту Java та забезпечувала ефективну розробку для платформи .NET Framework, а пізніше й .NET Core.

У 2001 році C# офіційно увійшла до складу .NET 1.0 і швидко здобула популярність. Згодом C# отримала стандартизацію ECMA-334 та ISO/IEC 23270, що забезпечило її відкритість і незалежність від конкретних версій продуктів Microsoft та закріпило її статус у світі розробки.

З часом C# активно еволюціонувала. Кожна нова версія додавала принципово нові можливості:

- C# 3.0 – інтегровані запити LINQ;
- C# 5.0 – підтримка асинхронного програмування (async/await);

- C# 7.0 – покращені кортежі, шаблони та робота з пам'яттю;
- C# 9.0 – рекордні типи (records) та сучасні шаблони;
- C# 10–12 – розширені можливості глобальних using, вдосконалені лямбди, нові типи та синтаксичні скорочення.

Сьогодні C# є ключовим елементом не тільки .NET Framework, але й сучасного .NET (Core, 5, 6, 7, 8), що підтримує кросплатформність і дозволяє запускати програми на Windows, Linux, macOS, Android, iOS та навіть у браузері завдяки технології Blazor. C# належить до найпопулярніших мов завдяки зручному синтаксису, широким бібліотекам і можливості працювати з різними пристроями – від IoT до хмарних сервісів.

Основні особливості мови.

C# відносять до багатопарадигмальних мов, адже вона підтримує об'єктно-орієнтоване програмування, функціональний підхід та елементи процедурного стилю. Її синтаксис близький до мов C, C++ та Java, що полегшує перехід для розробників із досвідом.

Ключові особливості мови:

- Статична типізація та суворий контроль за типами даних.
- Автоматичне керування пам'яттю через збирач сміття (garbage collector).
- Потужна стандартна бібліотека, інтегрована з .NET.
- LINQ для роботи з колекціями та базами даних у декларативному стилі.
- Асинхронність і підтримка багатопоточності.
- Високий рівень безпеки – зведення до мінімуму можливостей для помилок управління пам'яттю.

Важливо, що C# постійно вдосконалюється: нові версії додають сучасні можливості, роблячи мову більш виразною, лаконічною та зручною у використанні.

Переваги та недоліки мови.

Переваги:

- Простота та зрозумілий синтаксис – полегшує як створення, так і підтримку коду.
- Тісна інтеграція з .NET і величезна кількість готових бібліотек.
- Кросплатформність – завдяки сучасному .NET програми на C# працюють на різних ОС.
- Популярність та затребуваність – велика спільнота і високий попит на спеціалістів.
- Низький поріг входження – підходить навіть початківцям.
- Широкі можливості застосування – від простих програм до масштабних корпоративних систем.

Недоліки:

- Вимогливість до ресурсів (особливо у великих проєктах).
- Прив'язаність до екосистеми Microsoft (хоча останніми роками вона значно послабилася).
- Менша швидкодія, ніж у C++ при низькорівневих завданнях.

Синтаксичні та концептуальні особливості.

C# успадкувала багато рис від C, C++ та Java. Синтаксис схожий із цими мовами, що робить її зрозумілою для програмістів із досвідом. Вона підтримує статичну типізацію, об'єктно-орієнтовану парадигму (спадкування, інкапсуляцію, поліморфізм), а також функціональні елементи.

Особливістю є автоматичне керування пам'яттю (garbage collector), що зменшує кількість помилок. Подальші версії додали асинхронність, динамічне зв'язування, LINQ та узагальнені типи.

Порівняння з іншими мовами.

C++ і C#: У C++ програміст має прямий доступ до пам'яті, що дозволяє досягти максимальної продуктивності, але підвищує ризик помилок. У C# управління пам'яттю автоматизоване, що робить розробку безпечнішою, проте трохи менш продуктивною. C++ надає низькорівневий контроль і високу швидкодію, C# ж орієнтована на безпеку, автоматизоване управління пам'яттю та

простоту розробки. Тому C++ часто використовують у системному програмуванні, тоді як C# – у бізнес-додатках, вебсервісах і іграх.

C# і Java: обидві мови мають спільні ідеї та схожий синтаксис, але Java історично незалежна від конкретної платформи, тоді як C# тісно пов'язана з екосистемою Microsoft, хоча з .NET Core стала по-справжньому кросплатформною.

Галузі застосування.

C# вважається універсальною і використовується у багатьох напрямках:

- Десктопні та веб-додатки: створення програм для Windows, macOS і вебсервісів на базі .NET.
- Мобільні застосунки: завдяки Xamarin один код можна запускати на iOS та Android.
- Ігрова індустрія: рушій Unity зробив C# стандартом для розробки 2D- і 3D-ігор.
- Машинне навчання та штучний інтелект – бібліотека ML.NET дозволяє створювати моделі машинного навчання у середовищі .NET.
- Хмарні рішення – інтеграція з Microsoft Azure дає можливість розробляти масштабовані системи, мікросервіси та DevOps-рішення.
- Кібербезпека: створення антивірусних програм та інструментів захисту інформаційних систем.

Отже, мова C# однаково ефективна для бізнесу, науки та індустрії розваг.

Таким чином, C# – це сучасна, багатопарадигмальна мова програмування, яка поєднує простоту синтаксису та потужний інструментарій. Вона увібрала кращі риси мов сімейства C, але позбавлена їхніх ключових недоліків. Сьогодні C# є однією з найбільш затребуваних мов у світі, завдяки універсальності, кросплатформності та тісній інтеграції з .NET. Вона успішно використовується у бізнесі, освіті, ігровій індустрії, наукових дослідженнях і розробці хмарних систем.

Подальший розвиток C# і платформи .NET визначатиме ключові тенденції програмної інженерії, зокрема у сферах DevOps, мікросервісної архітектури та штучного інтелекту.

2.2 Вибір середовища програмування

2.2.1 Обґрунтування вибору середовища програмування

Загальна характеристика Microsoft Visual Studio.

Інтегроване середовище розробки (IDE) – це програмний комплекс, що поєднує у собі редактор коду, компілятор, відлагоджувач, інструменти для керування проєктами та додаткові модулі, які допомагають у процесі програмування. Головна його мета – створити для розробника зручний простір, де можна писати, перевіряти та вдосконалювати програмне забезпечення.

Microsoft Visual Studio з'явилася у 1997 році й відтоді перетворилася на одне з провідних середовищ програмування спочатку для Windows, а згодом і для інших платформ. Вона підтримує значну кількість мов: C#, C++, Visual Basic, Python, JavaScript, F#, а також дозволяє створювати застосунки для різних операційних систем і пристроїв.

Продукт існує у кількох редакціях: Community (безкоштовна для навчання та невеликих команд), Professional і Enterprise, орієнтовані на комерційне використання та розширений функціонал.

Основні можливості та інструменти Visual Studio.

Visual Studio – це потужне середовище, яке надає розробникам широкий набір засобів, зокрема:

- редактор коду з автодоповненням, підсвічуванням синтаксису та контекстними підказками;
- відлагоджувач для покрокового аналізу роботи програм;
- засоби створення графічних інтерфейсів (Windows Forms, WPF, GUI);
- модулі для роботи з базами даних (SQL Server, MySQL тощо);
- інтеграцію з системами контролю версій (Git, Azure DevOps);
- можливість встановлення розширень (*Extensions*), які додають нові функції за потреби.

Мова програмування C# у Visual Studio

Однією з ключових мов, які підтримує Visual Studio, є C#. Вона була створена компанією Microsoft у 2000 році та тісно пов'язана з платформою .NET, що забезпечує велику кількість готових бібліотек для розробки додатків.

Завдяки Visual Studio програмісти мають змогу швидко розробляти застосунки різного типу:

- настільні програми для Windows;
- веб-додатки за допомогою ASP.NET;
- мобільні застосунки (через Xamarin або .NET MAUI);
- комп'ютерні ігри (у рушії Unity, який базується на C#).

Visual Studio однаково активно використовується як у навчальному процесі, так і в професійній діяльності. Студенти застосовують його для вивчення програмування та створення навчальних проєктів, а компанії – для розробки корпоративних систем, вебсервісів, хмарних застосунків та ігор.

Переваги та недоліки Visual Studio.

Переваги:

- широкий набір інструментів у межах одного середовища;
- тісна інтеграція з Windows і сервісами Microsoft;
- підтримка різних мов програмування;
- зручна робота з C# та платформою .NET.

Недоліки:

- висока потреба в обчислювальних ресурсах;
- складність для новачків через велику кількість функцій;
- деякі версії потребують оплати.

Отже, Microsoft Visual Studio – одне з найпотужніших сучасних середовищ для розробки програмного забезпечення. Воно об'єднує всі необхідні інструменти для написання, тестування та вдосконалення програмних продуктів, а особливо ефективно працює з мовою C#. Завдяки своїй гнучкості та багатофункціональності Visual Studio широко використовується як у навчальних цілях, так і у створенні великих комерційних проєктів.

2.2.2 Робота в середовищі програмування

Для створення додатку для адміністратора освітніх курсів було вибрано середовище розробки Microsoft Visual Studio [8].

Visual Studio – це інтегроване середовище розробки програмного забезпечення, створене корпорацією Microsoft, яке надає розробникам потужний набір інструментів для створення різноманітних програмних продуктів. За допомогою цього середовища можна розробляти як прості консольні програми, так і застосунки з графічним інтерфейсом, зокрема класичні Windows Forms чи сучасні WPF-додатки. Крім того, Visual Studio підтримує створення веб-додатків, хмарних сервісів та навіть ігор, що робить його універсальним рішенням для програмістів [9].

Особливістю середовища є підтримка кількох мов програмування. Серед них найбільш поширеними є Visual C#, Visual Basic, Visual F#, Visual C++, а також Python та інші. Завдяки цьому розробники можуть обирати оптимальний інструмент залежно від специфіки завдання (див. рисунок 2.1).

Важливим є те, що Visual Studio не обмежується створенням програм під Windows. Завдяки сучасним технологіям, таким як .NET MAUI та інтеграції з мобільними платформами, воно дозволяє розробляти додатки і для Android, і для iOS, забезпечуючи кросплатформність і зручність у реалізації проєктів.

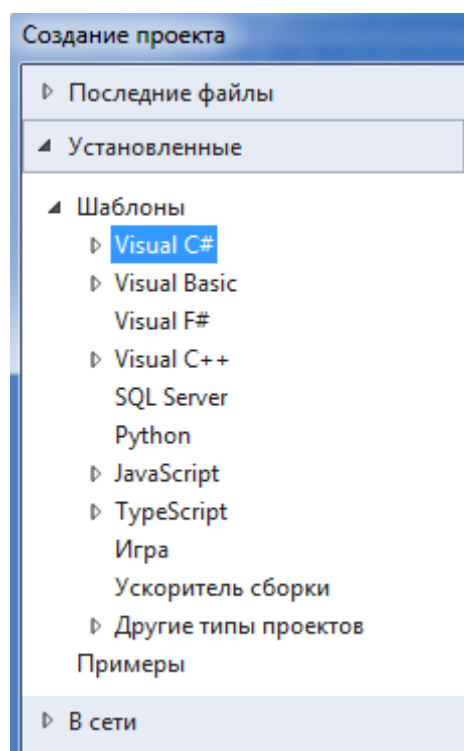


Рисунок 2.1 – Вибір мови програмування

Visual Studio Community є безкоштовною редакцією середовища, яка орієнтована на учнів, студентів, викладачів, а також на незалежних розробників та авторів програм із відкритим вихідним кодом [10]. Це робить її доступним інструментом для навчання та реалізації індивідуальних проєктів без фінансових витрат.

Створення програми на C# у Visual Studio

У середовищі Visual Studio передбачено кілька підходів до розробки програм. Найпоширеніший полягає у створенні окремих проєктів для кожної задачі [11]. Це дозволяє впорядкувати роботу та забезпечує просте керування файлами й компонентами.

Алгоритм створення нового проєкту можна подати так (див. рисунок 2.2):

1. Запустити Microsoft Visual Studio.
2. На початковій сторінці обрати пункт «Створити проєкт...», після чого відкривається вікно налаштування.
3. У верхній частині вікна необхідно вибрати мову Visual C# та тип застосунку:
 - Windows Forms App – для створення застосунку з графічним інтерфейсом;
 - Console App – для розробки консольної програми.
4. У нижній частині вікна вказується назва проєкту (поле Ім'я:), а також шлях до місця його збереження (кнопка Огляд у полі Розташування:). Варто також перевірити наявність позначки біля пункту Створити каталог для рішення.
5. Після підтвердження натисканням кнопки ОК середовище автоматично створить базову структуру проєкту, готову до написання коду [12].

Таким чином, Visual Studio не лише спрощує сам процес створення програм, але й надає інтуїтивно зрозумілий інтерфейс, що особливо корисно як для початківців, так і для досвідчених розробників.

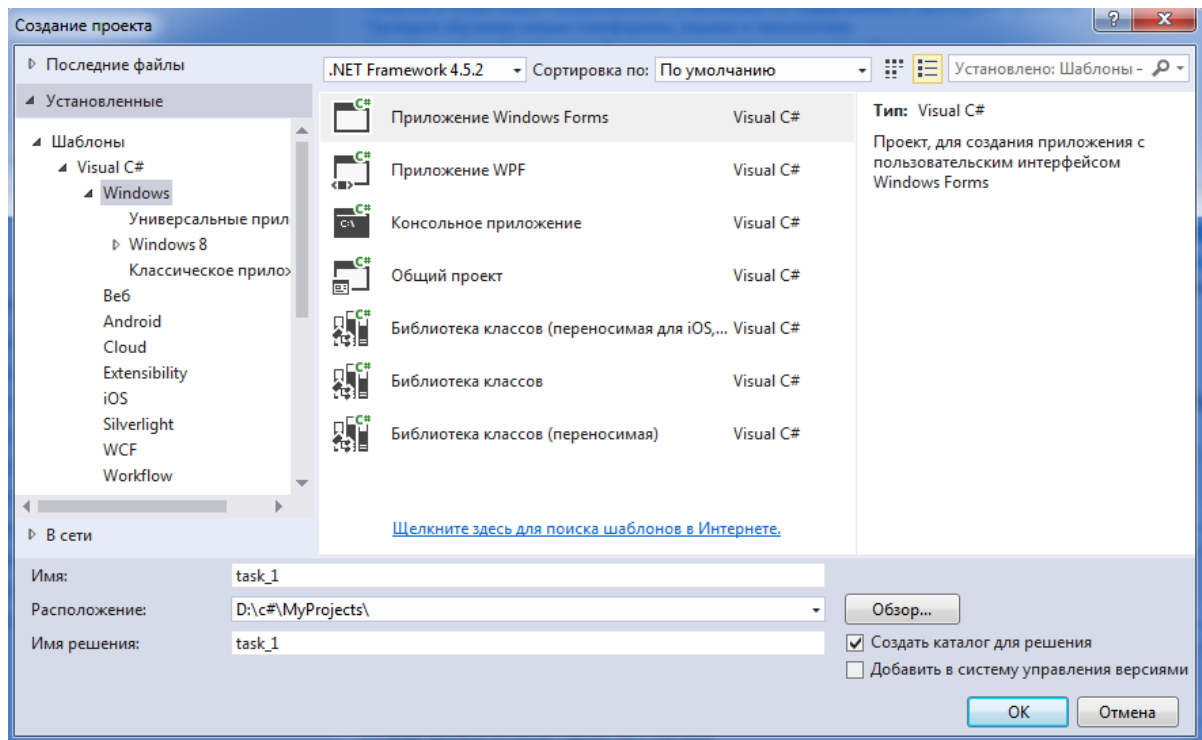


Рисунок 2.2 – Вибір каталогу

Результатом виконання описаного алгоритму буде створення в папці MyProjects окремого каталогу Task_1, у якому зберігатимуться всі файли відповідного проєкту. Таким чином, кожна нова задача матиме власну структуру з унікальною директорією та проєктними файлами. Такий підхід є зручним для роботи з невеликими завданнями, однак у разі потреби реалізувати кілька пов'язаних програм він може виявитися менш ефективним.

У цьому випадку доцільно застосовувати більш універсальний спосіб організації роботи – об'єднання кількох проєктів у єдине Рішення (Solution). Visual Studio підтримує цю можливість, дозволяючи групувати програми в рамках одного середовища. Завдяки цьому розробник отримує впорядковану структуру, де всі задачі зберігаються разом, але водночас залишаються незалежними проєктами.

Наприклад, для виконання трьох завдань можна створити таку ієрархію:

Рішення Lesson_1

– Проєкт Task_1

– Проєкт Task_2

– Проєкт Task_3

Алгоритм створення рішення виглядає наступним чином:

1. Запустити Microsoft Visual Studio.
2. На початковій сторінці обрати пункт «Створити проєкт...», після чого відкриється вікно налаштування.
3. У верхній частині вікна вказати мову Visual C# та визначити тип застосунку:
 - Windows Forms App – для розробки застосунку з графічним інтерфейсом;
 - Console App – для створення консольної програми.
4. У нижній частині вікна вписати назву першої задачі (наприклад, Task_1), задати шлях до збереження за допомогою кнопки Огляд (поле Розташування:), а в полі Ім'я рішення (Solution name) ввести назву загального рішення (Lesson_1). Варто також перевірити, щоб була активною опція Створити каталог для рішення.
5. Натискаємо ОК (див. рисунок 2.3).

Після підтвердження налаштувань Visual Studio створить структуру, у якій зберігатиметься як саме рішення, так і перший проєкт. Наступні задачі (Task_2, Task_3) можна буде додати до цього рішення за допомогою опції «Додати новий проєкт», що дозволяє ефективно керувати кількома програмами в межах однієї логічної групи.

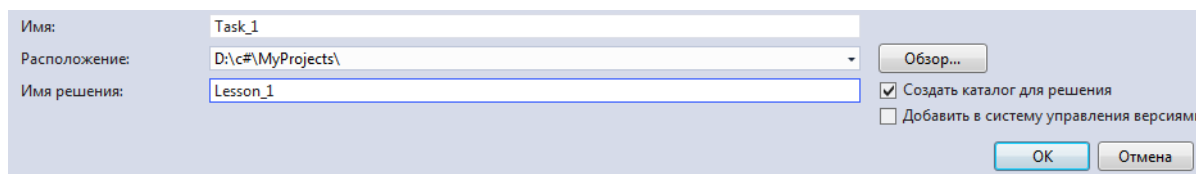


Рисунок 2.3 – Створення проєкту

Ми вже створили рішення Lesson_1 та перший проєкт Task_1 у його складі. Щоб додати нові проєкти до цього ж рішення, необхідно виконати такі кроки:

1. Вибрати пункт меню Файл → Створити → Проект... або скористатися комбінацією клавіш [Ctrl] + [Shift] + [N]. Відкриється знайоме вікно Створення проєкту.

2. Подальші дії виконуються аналогічно до попередніх, за винятком кроку 4. Тут у полі Ім'я: потрібно ввести назву нового проєкту, наприклад, Task_2.

3. У полі Розташування: обов'язково обрати опцію Додати до рішення (див. рис. 2.4).

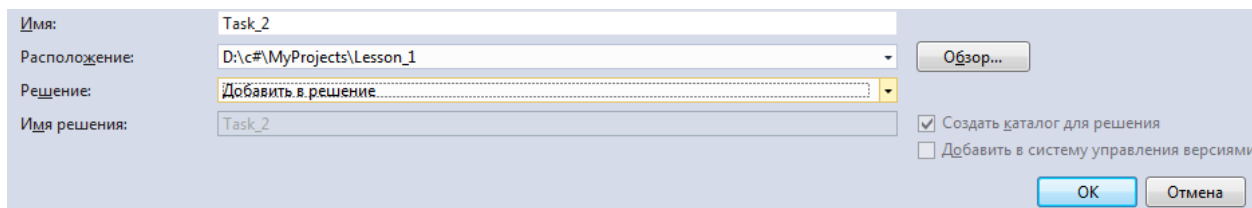


Рисунок 2.4 – Обрання імені та розміщення проекту

Щоб повторно зберегти проєкт, скористайтесь командою Файл→ Зберегти все або комбінацією клавіш [Ctrl] + [Shift] + [S].

Створення першого проєкту. Форма та її властивості

Після створення у Visual Studio нового проєкту типу Windows Forms (віконний застосунок) одразу відкривається головне вікно програми – форма (за замовчуванням Form1). У подальшому саме на цій формі ми будемо розташовувати різноманітні елементи інтерфейсу.

Цікаво, що навіть без жодних змін у щойно створеному проєкті його вже можна запустити. Для цього достатньо натиснути кнопку Пуск на головній панелі Visual Studio або клавішу [F5]. У результаті побачимо порожнє вікно з назвою Form1 (див. рисунок 2.5).

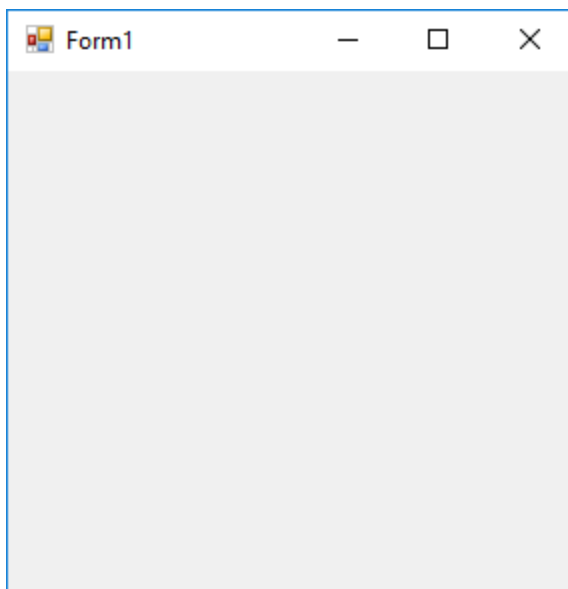


Рисунок 2.5 – Пустий проєкт

Після завершення перегляду запущену програму слід закрити, натиснувши кнопку Закрити у вікні застосунку.

Якщо ж у складі Рішення створено кілька проєктів, перед запуском необхідно в головному вікні Visual Studio вибрати саме той проєкт, який ви хочете протестувати (див. рисунок 2.6).

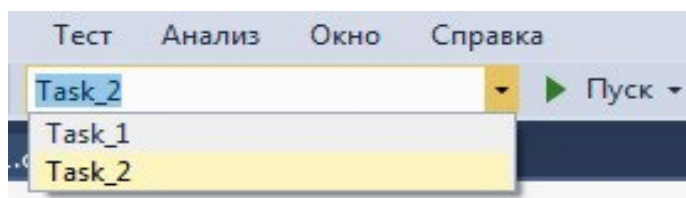


Рисунок 2.6 – Вибір проєкту

Одним із важливих елементів середовища Visual Studio є вікно властивостей об'єкта – Свойства (див. рисунок 2.7). У цьому вікні відображається перелік усіх властивостей обраного елемента та значення, які вони приймають.

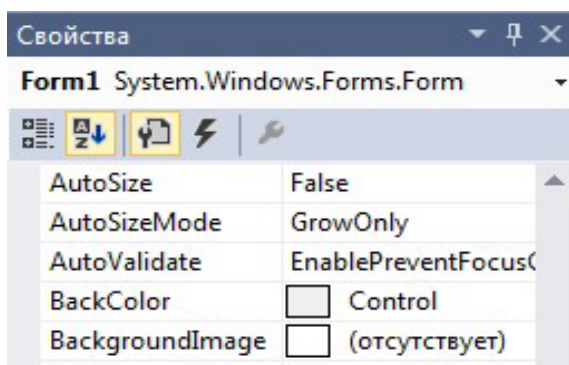


Рисунок 2.7 – Перелік властивостей

Розглянемо деякі властивості Форми (див. рисунок 2.8):

Властивість	Пояснення
Text	Заголовок вікна
Size - Width - Height	Розмір форми (ширина; висота)*
BackColor	Колір фону форми
BackgroundImage	Фоновий малюнок форми
BackgroundImageLayout	Спосіб відображення фонового малюнку форми: - Zoom (зображення приймає розміри форми, зберігаючи пропорції); - Stretch (зображення заповнює всю форму, при цьому пропорції не зберігаються)
StartPosition	Позиція форми на екрані після запуску: - Manual (лівий верхній кут форми знаходиться в координатах, заданих у властивості Location); - CenterScreen (форма з'являється в центрі екрана)
Location - X - Y	Координати лівого верхнього кута форми по відношенню до лівого верхнього кута екрану. Дана властивість працює, якщо значення властивості StartPosition= Manual
Opacity	Відсоток непрозорості форми

Рисунок 2.8 – Властивості форми

Якщо не потрібно задавати точні розміри об'єкта, їх можна легко скоригувати за допомогою миші, змінюючи межі форми безпосередньо на екрані.

Відкриття рішення

Збережене раніше рішення або проект можна відкрити кількома способами:

- знайти та запустити файл із розширенням **.sln** (див. рисунок 2.9);
- у головному меню Visual Studio обрати команду Файл → Відкрити → Рішення або проект...;
- скористатися комбінацією клавіш [Ctrl]+[Shift]+[O].

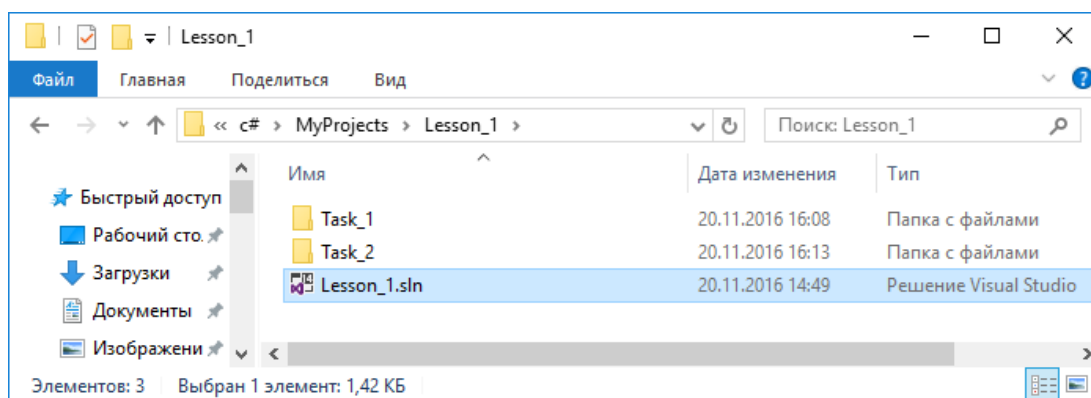


Рисунок 2.9 – Відкриття проекту

Якщо після відкриття рішення у Visual Studio не з'являється вікно Конструктора форми (див. рисунок 2.10), слід у панелі Обозреватель решений двічі клацнути по файлу **Form1.cs** (див. рисунок 2.11).

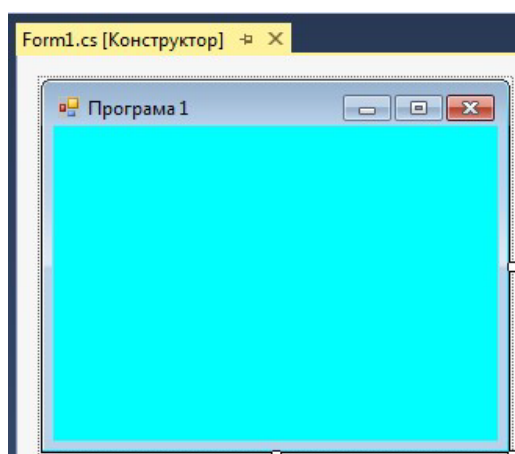


Рисунок 2.10 – Конструктор форм

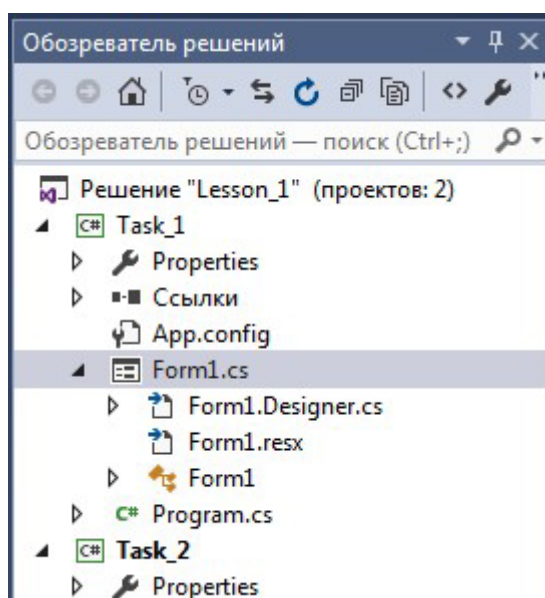


Рисунок 2.11 – Огляд проекту

Налаштування інтерфейсу Microsoft Visual Studio.

Середовище розробки Visual Studio має зручний багатовіконний інтерфейс (див. рисунок 2.12), який легко адаптувати до потреб користувача.

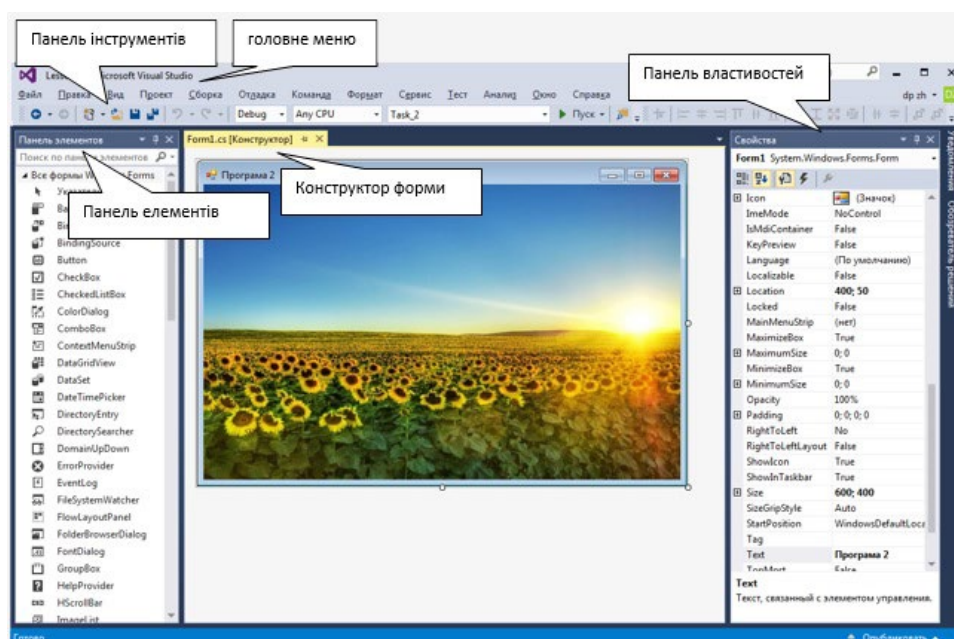


Рисунок 2.12 – Интерфейс

Панелі можна вільно переміщувати та змінювати їх кількість. Деякі з них можна згорнути, і тоді вони відображатимуться у вигляді вкладок зліва або справа. Наприклад, так працює панель Обозреватель решений (див. рисунок 2.13).

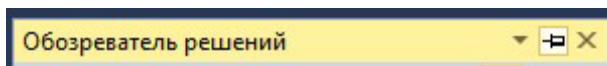


Рисунок 2.13 – Панель огляду рішень

Згорнуті панелі у Visual Studio можна відкрити, просто клацнувши на їхній назві. Водночас після завершення роботи вони автоматично згортаються знову. Щоб залишити панель відкритою, слід натиснути на значок «канцелярської кнопки» (див. рисунок 2.14).

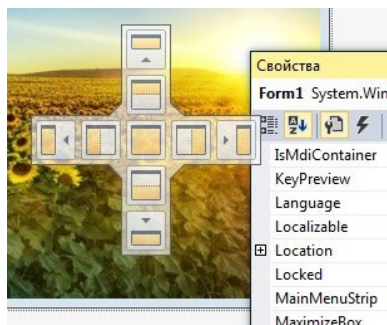


Рисунок 2.14 – Канцелярські кнопки

Якщо потрібна панель відсутня у вікні Visual Studio, її можна відобразити через головне меню, обравши пункт Вид.

Для переміщення панелі в іншу частину екрана використовується спеціальна схема розташування, яка з'являється одразу після початку перетягування.

У вікні Свойства властивості компонентів можуть бути згруповані за категоріями або ж показані одним списком, відсортованим за алфавітом.

Змінити кольорову схему середовища Visual Studio можна за допомогою команди: Сервіс → Параметри → Общие → Цветовая тема.

2.3 Вибір СКБД

2.3.1 Проблема зберігання даних. Бази даних

Поняття бази даних.

База даних (БД) – це впорядкований набір логічно пов'язаних даних, призначений для задоволення інформаційних потреб користувачів [13, 28, 29].

Іншими словами, БД можна визначити як колекцію таблиць, схем, хешів, представлень (переглядів) та інших об'єктів, характерних для систем збереження даних.

Види баз даних.

Бази даних можна класифікувати за різними ознаками:

- За структурою:
 - ✓ ієрархічні – дані зберігаються у вигляді «дерева» (наприклад, IBM Information Management System);
 - ✓ мережеві – дозволяють елементу мати кілька зв'язків (наприклад, Integrated Data Store);
 - ✓ об'єктно-орієнтовані – поєднують принципи БД і об'єктно-орієнтованого програмування (наприклад, db4o, ObjectDB);
 - ✓ реляційні – найпоширеніші, будуються на основі таблиць (наприклад, MySQL, PostgreSQL, Oracle);
 - ✓ нереляційні (NoSQL) – використовують інші моделі збереження: документи, графи, ключ-значення (наприклад, MongoDB, Neo4j, Redis).
- За організацією:
 - ✓ локальні – зберігаються на одному комп'ютері (наприклад, Access);
 - ✓ розподілені – дані розміщені на кількох вузлах мережі (наприклад, Cassandra).
- За ступенем упорядкованості:
 - ✓ структуровані – мають чітко визначені схеми (наприклад, SQL-бази даних);
 - ✓ неструктуровані – працюють із текстами, мультимедіа тощо (наприклад, Elasticsearch).
- За змістом:
 - ✓ географічні (ГІС-системи),
 - ✓ історичні (бази архівних документів),
 - ✓ наукові (бази експериментальних даних, наприклад, GenBank у біології).

Архітектура та доступ до БД

За способом доступу розрізняють:

- бази даних з локальним доступом – користувач працює з БД безпосередньо на своєму ПК;
- бази даних з віддаленим (мережевим) доступом – дані розміщені на сервері, а доступ здійснюється через мережу.

Для мережових БД використовують різні архітектури [14]:

- файл-сервер – сервер зберігає файли БД, а клієнти отримують до них доступ напряду (наприклад, MS Access у спільному доступі);
- клієнт-сервер – клієнт надсилає запити, а сервер (наприклад, MySQL чи MS SQL Server) обробляє їх і повертає результат.

Клієнт-серверна архітектура є однією з найпоширеніших моделей, на основі якої функціонує сучасний інтернет. Саме тому важливо розуміти її сутність та особливості. Основна ідея концепції полягає в тому, що центральний комп'ютер (сервер бази даних) не лише зберігає інформацію у централізованій БД, а й виконує основну частину операцій з обробки даних. Коли клієнт (робоча станція) надсилає запит, сервер здійснює пошук і відбір необхідної інформації, після чого результати передаються мережею назад клієнту. У випадку створення нових записів дані зберігаються або оновлюються у базі даних веб-додатку. Важливою особливістю клієнт-серверної архітектури є використання мови запитів SQL.

Перед тим як визначити поняття репозиторій, варто згадати базову аналогію. Якщо одна база даних виконує роль простого сховища інформації, то об'єднання двох і більше БД формує сховище даних. Репозиторій у цьому контексті можна охарактеризувати як масштабну базу даних, що має розширені функціональні можливості.

На відміну від звичайних БД, репозиторії даних слугують не лише для зберігання інформації, але й для управління як самими даними, так і метаданими, тобто відомостями про їхню структуру. Крім того, у репозиторіях підтримується існування кількох варіантів представлення однієї й тієї ж інформації (наприклад, у вигляді знімків, блокчейнів тощо).

Одним із найпоширеніших прикладів репозиторію є платформа GitHub [15].

Тестування баз даних спрямоване на перевірку їхньої цілісності та достовірності. Його метою є впевнитися, що дані, які відображаються на рівні Front-end, відповідають тим, що зберігаються у Back-end [16].

Комплексне тестування БД має включати:

- валідацію даних;
- перевірку відсутності втрат записів;
- правильність відображення зв'язків між даними;
- контроль за коректним збереженням і оновленням зв'язків (наприклад, під час видалення foreign key, primary key, батьківських чи дочірніх елементів);
- повне оновлення даних у БД для забезпечення їхнього однакового відображення у всіх інтерфейсах;
- перевірку коректності значень.

Реляційна база даних визначається як впорядкована сукупність елементів інформації [17], що організовані у вигляді набору формально описаних таблиць. Дані в такій системі можуть бути отримані або комбіновані різними способами без необхідності зміни структури самих таблиць.

Іншими словами, реляційна БД – це набір таблиць та взаємозв'язків між ними. Кожна таблиця складається зі стовпців і рядків: стовпці (поля) формують структуру таблиці, а рядки репрезентують її дані. Окрема клітинка, яка розташована на перетині стовпця та рядка, містить одну одиницю інформації – конкретне значення або порожній показник (NULL). Рядок таблиці також називають записом (record), тоді як кожне поле або стовпець має визначений тип даних.

Нереляційні бази даних становлять протилежне поняття реляційним. Якщо у реляційних БД інформація зберігається у вигляді взаємопов'язаних таблиць, то в нереляційних – дані можуть існувати як єдина сутність. Організація таких даних зазвичай здійснюється за принципом ключ–значення або у вигляді ієрархічних структур. Структуровані бази даних орієнтовані переважно на відображення

інформації про конкретні об'єкти (наприклад, людей, товари, ціни, заробітну плату тощо), тоді як нереляційні системи здебільшого застосовуються у сфері документообігу.

Зростання популярності нереляційних БД збіглося з активним розвитком інтернету, оскільки саме вони забезпечують ефективну організацію даних у масштабних проєктах, зокрема в пошукових системах на зразок Google. Такі БД дозволяють розв'язувати проблеми масштабованості, підвищення продуктивності та забезпечення гнучкості системи. Саме тому поняття “нереляційні БД” часто асоціюється з терміном NoSQL, а також напряду пов'язане з сучасною галуззю Big Data.

Система управління базами даних (СУБД) [18] для реляційних БД – це програмно-технічний комплекс, що забезпечує визначення взаємозв'язків між таблицями, створення, модифікацію, контроль і управління даними відповідно до міжнародного стандарту ISO/IEC 2382:2015. Програмне забезпечення для роботи з БД може бути інтегрованою частиною СУБД або функціонувати автономно.

Серед найпоширеніших систем управління базами даних варто відзначити MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, Interbase, Firebird та IBM DB2. Використання таких інструментів дозволяє ефективно опрацьовувати великі обсяги даних, що практично неможливо здійснити вручну.

Система управління базами даних повинна забезпечувати користувачам доступ через декілька різних інтерфейсів. Вони можуть бути як графічними, так і текстовими. Графічний інтерфейс передбачає введення даних за допомогою клавіатури або миші, а результати відображаються на екрані у візуально зручному форматі. Одним із різновидів текстового інтерфейсу, який часто застосовується у СУБД, є інтерфейс командного рядка. У цьому випадку взаємодія здійснюється шляхом введення команд із клавіатури, а система повертає результат у текстовому вигляді. Такий підхід забезпечує зрозуміле, хоча й менш візуально привабливе, ніж у графічному інтерфейсі, відображення даних.

Головною відмінністю між СУБД та реляційною системою управління базами даних (РСУБД) [19] є спосіб організації зберігання інформації. У РСУБД

дані структуруються у вигляді таблиць із чітко визначеними зв'язками між ними, тоді як у нереляційних системах такі зв'язки відсутні. Дані можуть зберігатися у файловому вигляді або в ієрархічних структурах.

Для забезпечення ефективної роботи будь-яка СУБД повинна надавати користувачам такі функціональні можливості:

- підтримку різних інтерфейсів взаємодії;
- фізичну незалежність даних (зміни у фізичному представленні не повинні впливати на архітектуру БД);
- логічну незалежність даних;
- оптимізацію запитів;
- забезпечення цілісності даних;
- управління паралельним доступом;
- резервне копіювання та відновлення даних;
- захист інформації та безпеку бази даних.

2.3.2 Обґрунтування вибору СКБД

Загальна характеристика СКБД MySQL

MySQL – одна з найпопулярніших у світі систем керування реляційними базами даних з відкритим вихідним кодом. Вона була розроблена у 1995 році шведською компанією MySQL AB, а згодом придбана компанією Sun Microsystems, яка у 2010 році увійшла до складу корпорації Oracle. Нині MySQL продовжує активно розвиватися та підтримується великою спільнотою розробників і користувачів.

Основне призначення MySQL полягає у створенні, зберіганні, модифікації та управлінні даними в реляційній моделі. СКБД побудована за принципами клієнт–серверної архітектури: серверна частина відповідає за обробку запитів і керування даними, а клієнтська частина – за взаємодію користувача чи застосунку з сервером.

MySQL є кросплатформним програмним продуктом: вона працює під керуванням Windows, Linux, macOS, FreeBSD, Solaris та інших операційних систем.

Завдяки цьому розробники можуть інтегрувати її практично з будь-якими сучасними застосунками.

Архітектура MySQL

MySQL побудована на основі класичної клієнт–серверної архітектури. Серверна частина відповідає за керування даними, виконання запитів, контроль транзакцій та забезпечення безпеки. Клієнтська частина може бути представлена у вигляді командного рядка, графічних утиліт або ж сторонніх застосунків, що взаємодіють із сервером через мережеві протоколи. Такий підхід дозволяє відокремити обробку даних від прикладного рівня та забезпечити ефективний розподіл навантаження.

Ядро MySQL складається з кількох основних компонентів. По-перше, це модуль оптимізатора запитів, який аналізує SQL-інструкції та формує найбільш ефективний план їх виконання. По-друге, система керування буферами та кешами, що зберігає проміжні результати й часто використовувані дані у пам'яті, скорочуючи кількість звернень до диску. По-третє, механізми зберігання даних, серед яких найпоширенішим є InnoDB, що забезпечує підтримку транзакцій, зовнішніх ключів та багатокористувацького доступу з високим рівнем захисту. Завдяки використанню різних механізмів зберігання MySQL може бути оптимізованою як для систем, де пріоритетом є швидкість читання, так і для застосунків, де першочерговим є забезпечення цілісності даних.

Окремо варто згадати можливості реплікації. MySQL дозволяє налаштовувати передачу змін між кількома серверами, що сприяє створенню відмовостійких систем і балансуванню навантаження. Це особливо актуально для проєктів із великою кількістю одночасних користувачів.

Основні можливості MySQL

До ключових функціональних характеристик MySQL належать

- Підтримка мови SQL (Structured Query Language) – стандартної мови для роботи з реляційними базами даних.

- Розширюваність і масштабованість – можливість обробки як невеликих баз для локальних проєктів, так і високонавантажених систем з мільйонами записів.
- Робота з транзакціями – забезпечення цілісності даних за допомогою властивостей ACID (атомарність, узгодженість, ізолюваність, довговічність).
- Підтримка кількох типів таблиць – зокрема InnoDB (основний механізм з підтримкою транзакцій та зовнішніх ключів) та MyISAM (швидкий механізм для читання й аналітики).
- Можливість створення збережених процедур, тригерів і подань – для гнучкого управління логікою обробки даних.
- Підтримка реплікації – що дає змогу будувати відмовостійкі системи та розподіляти навантаження між кількома серверами.
- Високий рівень безпеки – завдяки системі користувачів і ролей, контролю доступу та можливості шифрування з'єднань.

Практичне використання MySQL

На практиці MySQL широко застосовується у найрізноманітніших сферах. Однією з найбільш відомих є веб-розробка. Завдяки інтеграції з мовою PHP MySQL стала стандартним рішенням для динамічних вебсайтів та інтернет-магазинів. Популярні системи управління контентом, такі як WordPress, Joomla та Drupal, за замовчуванням використовують MySQL як основну базу даних. Це забезпечує легкість розгортання сайтів будь-якої складності – від особистих блогів до великих порталів.

Не менш активно MySQL використовується в електронній комерції. Багато платформ для створення інтернет-магазинів (Magento, OpenCart, PrestaShop) побудовані саме на цій СКБД. Це пояснюється тим, що MySQL добре справляється з великою кількістю одночасних транзакцій, дозволяє ефективно зберігати інформацію про замовлення, користувачів, товари та фінансові операції.

В освітніх та наукових проєктах MySQL цінується за простоту, безкоштовність і доступність. Її часто використовують як навчальну базу для

студентів, які вивчають реляційні бази даних і принципи роботи SQL. У багатьох університетах саме MySQL є першою СКБД, з якою стикаються майбутні програмісти та аналітики.

На державному рівні MySQL також знаходить застосування. В Україні її використовують у різних інформаційних системах для зберігання статистики, організації електронного документообігу та створення сервісів для громадян. Завдяки відкритій ліцензії це дозволяє зменшувати витрати на програмне забезпечення та водночас забезпечувати достатній рівень функціональності.

Переваги використання MySQL

MySQL займає провідні позиції серед реляційних СКБД завдяки низці суттєвих переваг:

- Відкритий вихідний код і безкоштовна ліцензія (GPL) – дає змогу вільно використовувати систему для більшості проєктів без додаткових витрат.
- Висока продуктивність – оптимізовані алгоритми обробки запитів забезпечують швидкий доступ до даних.
- Масштабованість – придатність як для невеликих сайтів, так і для глобальних високонавантажених систем.
- Простота встановлення та адміністрування – наявність великої кількості інструментів (phpMyAdmin, MySQL Workbench тощо).
- Широка підтримка спільнотою – активні форуми, документація, навчальні матеріали.
- Інтеграція з популярними мовами програмування – PHP, Java, Python, C#, JavaScript.

Недоліки MySQL

Попри значні переваги, система має й певні обмеження, які необхідно враховувати:

- Обмежена підтримка складних аналітичних запитів у порівнянні з такими СКБД, як Oracle чи PostgreSQL.
- Менша кількість вбудованих засобів для роботи з великими обсягами даних у бізнес-аналітиці.

- Історично деякі типи таблиць (наприклад, MyISAM) не підтримували транзакцій, хоча зараз основний механізм InnoDB компенсує цей недолік.

Практичне використання MySQL

На практиці MySQL широко застосовується у найрізноманітніших сферах. Однією з найбільш відомих є веб-розробка. Завдяки інтеграції з мовою PHP MySQL стала стандартним рішенням для динамічних вебсайтів та інтернет-магазинів. Популярні системи управління контентом, такі як WordPress, Joomla та Drupal, за замовчуванням використовують MySQL як основну базу даних. Це забезпечує легкість розгортання сайтів будь-якої складності – від особистих блогів до великих порталів.

Не менш активно MySQL використовується в електронній комерції. Багато платформ для створення інтернет-магазинів (Magento, OpenCart, PrestaShop) побудовані саме на цій СКБД. Це пояснюється тим, що MySQL добре справляється з великою кількістю одночасних транзакцій, дозволяє ефективно зберігати інформацію про замовлення, користувачів, товари та фінансові операції.

В освітніх та наукових проєктах MySQL цінується за простоту, безкоштовність і доступність. Її часто використовують як навчальну базу для студентів, які вивчають реляційні бази даних і принципи роботи SQL. У багатьох університетах саме MySQL є першою СКБД, з якою стикаються майбутні програмісти та аналітики.

На державному рівні MySQL також знаходить застосування. В Україні її використовують у різних інформаційних системах для зберігання статистики, організації електронного документообігу та створення сервісів для громадян. Завдяки відкритій ліцензії це дозволяє зменшувати витрати на програмне забезпечення та водночас забезпечувати достатній рівень функціональності.

Обґрунтування доцільності вибору MySQL

Вибір СКБД є критичним етапом у процесі розробки будь-якої інформаційної системи. MySQL доцільно обирати з огляду на такі фактори:

- Економічна ефективність – відсутність витрат на ліцензії робить MySQL оптимальним рішенням для навчальних, наукових та комерційних проєктів в Україні.

- Простота використання – зручність у роботі дає змогу швидко навчати нових фахівців та інтегрувати базу даних у проєкт.
- Надійність і стабільність – MySQL використовується у відомих світових продуктах (Wikipedia, WordPress, YouTube), що підтверджує її життєздатність у реальних умовах.
- Гнучкість і масштабованість – підходить як для невеликих локальних систем, так і для проєктів національного масштабу.
- Підтримка різних мов програмування – робить систему універсальною для розробників.
- Актуальність для ринку праці – MySQL залишається однією з найчастіше вживаних СКБД, а знання роботи з нею є затребуваним серед роботодавців.

Таким чином, MySQL є однією з найбільш збалансованих систем керування реляційними базами даних, яка поєднує у собі простоту, надійність, гнучкість та економічну доступність. Для дипломного проєкту її використання є цілком доцільним, оскільки система відповідає вимогам до сучасних СКБД, забезпечує належний рівень продуктивності та безпеки, а також має широку підтримку серед розробників.

2.4 Моделювання інформаційної системи

2.4.1 Загальні вимоги до роботи інформаційної системи

Для створення інформаційної системи адміністратора освітніх курсів буде застосовано об'єктно-орієнтовану мову програмування C# у поєднанні з інтегрованим середовищем розробки Microsoft Visual Studio та системою керування базами даних MySQL. Такий вибір зумовлений необхідністю забезпечення надійності, масштабованості та зручності у використанні майбутнього програмного продукту.

Проблема вибору тематики полягає у відсутності достатньо простого та інтуїтивно зрозумілого програмного забезпечення для адміністрування освітніх курсів [20]. Існуючі аналоги здебільшого орієнтовані на потреби загальноосвітніх шкіл та закладів вищої освіти, специфіка роботи яких суттєво відрізняється від

діяльності навчальних курсів і тренінгових центрів. Як наслідок, адміністраторам курсів доводиться використовувати або надто складні системи, що містять багато зайвих функцій, або ж поєднувати кілька різних інструментів, що призводить до дублювання даних і втрати часу.

Запропонована програма призначена для автоматизації щоденної роботи адміністратора освітніх курсів. Вона дозволить зберігати й обробляти інформацію про слухачів і навчальні програми. Адміністратор матиме можливість вносити нові дані про слухачів, редагувати їх та видаляти, переглядати актуальні списки та отримувати доступ до контактних даних. Наявність таких можливостей є важливою, оскільки вони забезпечують зручність у роботі навчального центру, зокрема дають змогу оперативно зв'язатися зі слухачем для уточнення організаційних питань.

Обов'язковим атрибутом у програмі виступатиме електронна пошта слухача. Це необхідно для надсилання інформації, пов'язаної з навчальним процесом: нагадувань про заняття, повідомлень про зміни в розкладі, результатів тестування та інших повідомлень. Крім того, у системі передбачається можливість створення нотаток, які дозволять фіксувати важливі деталі, особливі побажання чи індивідуальні потреби слухачів.

Реалізація функцій програми

У межах розробки інформаційної системи адміністратора освітніх курсів передбачено реалізацію низки ключових функціональних можливостей, що забезпечують повноцінну роботу з даними та зручність користування програмним продуктом.

Однією з основних функцій є запис і виведення інформації про слухачів курсів. Адміністратор повинен мати можливість додавати нових слухачів, редагувати дані про них та за потреби видаляти записи. Це дозволяє підтримувати актуальність інформації й забезпечує ефективний контроль за навчальним процесом. Крім того, реалізовано механізм перегляду збережених відомостей, що значно полегшує доступ до необхідних даних у процесі роботи.

Подібним чином потрібно організувати запис і виведення інформації про викладачів. Система повинна дозволяти вести облік педагогічного складу, зберігати контактні дані та інші відомості, необхідні для управління навчальними курсами. Це сприяє централізації даних і зменшує ризик їх втрати чи дублювання.

Важливою складовою є функція ведення нотаток. Вона дає змогу адміністратору фіксувати додаткову інформацію про слухачів чи викладачів, занотовувати особливі побажання, важливі коментарі або організаційні деталі. Наявність такого інструменту робить систему більш гнучкою і зручною у практичному використанні.

Не менш важливим є механізм реєстрації та входу користувачів у програму. Для цього буде реалізовано авторизацію за допомогою логіну та пароля. Такий підхід забезпечує розмежування доступу до інформації, підвищує рівень безпеки й дозволяє контролювати дії користувачів у системі.

Додатково у програмі буде реалізовано функцію калькулятора, яка може бути використана для виконання необхідних розрахунків у процесі роботи адміністратора. Це дозволяє уникнути потреби у сторонніх інструментах та підвищує зручність користування програмним забезпеченням.

Уся робота з даними слухачів і викладачів буде реалізована через систему керування базами даних MySQL. Саме ця СКБД відповідає за збереження, обробку та виведення даних, забезпечуючи їхню цілісність і надійність. Використання MySQL робить систему більш гнучкою та масштабованою, що є важливим для потенційного розширення функціоналу у майбутньому.

Окрему увагу буде приділено реалізації алгоритму пошуку. Користувач програми зможе швидко знайти потрібну інформацію за заданими критеріями, що значно підвищує продуктивність роботи адміністратора. Наявність ефективного пошуку є критично важливою функцією, адже зростання кількості даних у системі ускладнює їх ручний перегляд.

Таким чином, будуть реалізовані функціональні можливості, які охоплюють усі ключові аспекти роботи адміністратора освітніх курсів: від збереження й обробки

даних до забезпечення зручного доступу, безпеки та додаткових інструментів для підвищення ефективності роботи.

2.4.2 Розробка алгоритмів створення інформаційної системи

Реалізація запису та виведення інформації про слухачів, викладачів та нотаток.

Реалізація запису інформації про слухачів, викладачів та нотаток (див. рисунок 2.15).

Представимо алгоритм [21] у вигляді команд [22]:

- При введенні даних слухачів, викладачів та нотаток перевіряти підключення до бази даних. Якщо воно є – переходити до наступного пункту, якщо його немає – запис не реалізується.
- Дані заносяться в базу даних для подальшої роботи з ними.
- Закінчити роботу.

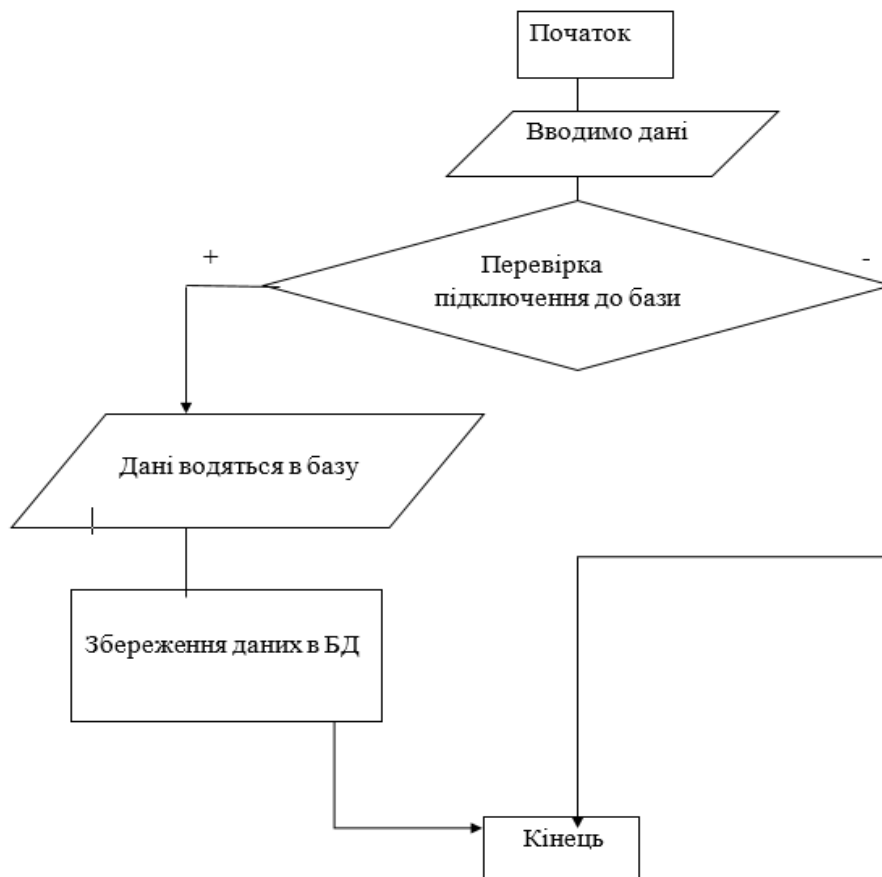


Рисунок 2.15 – Блок-схема реалізації запису інформації про слухачів, викладачів, нотаток.

Реалізація виведення інформації про слухачів, викладачів та нотаток (див. рисунок 2.16).

Представимо алгоритм у вигляді команд:

- При виведенні даних слухачів, викладачів та нотаток перевіряти підключення до бази даних. Якщо воно є – переходимо до наступного пункту, якщо його немає – запис не реалізується.
- Вивести дані на форму
- Закінчити роботу.

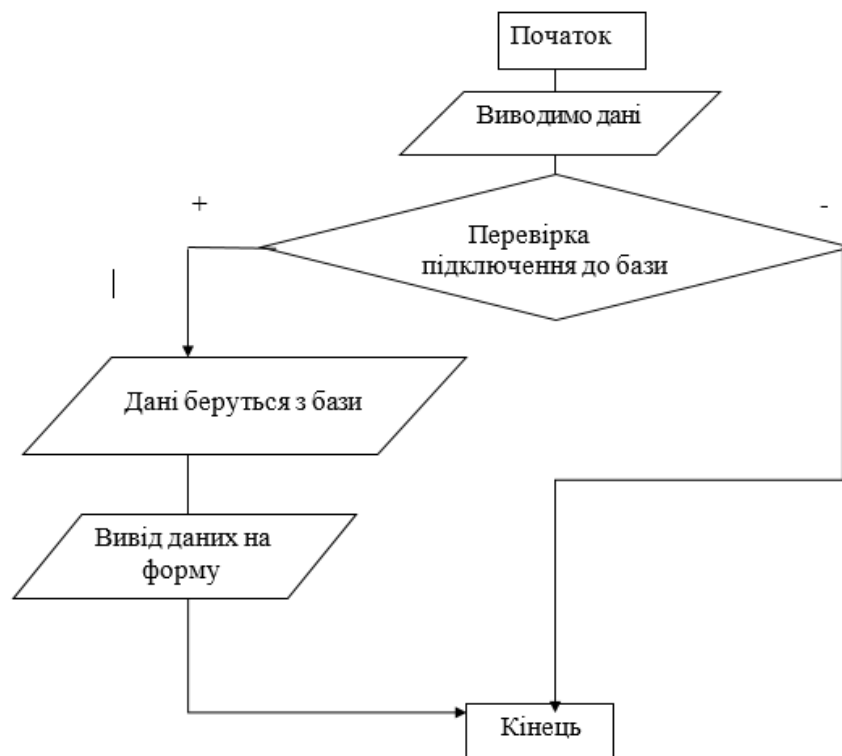


Рисунок 2.16 – Блок-схема реалізації виведення інформації про слухачів, викладачів, нотаток.

Реалізація реєстрації та входу користувача в програму за допомогою логіну та паролю.

Реалізація реєстрації користувача (див. рисунок 2.17).

Представимо алгоритм у вигляді команд:

- При введенні даних, потрібних для реєстрації перевіряється пароль та підтвердження паролю. Якщо вони співпадають, то дані записуються до бази

даних та ми переходимо до наступного пункту, якщо вони не співпадають, з'являється повідомлення про помилку.

- Перевіряємо підключення до бази даних, якщо воно існує, переходимо до наступного пункту, якщо його немає – запис не реалізується.
- Дані заносяться в базу даних для подальшої роботи з ними.
- Закінчити роботу.

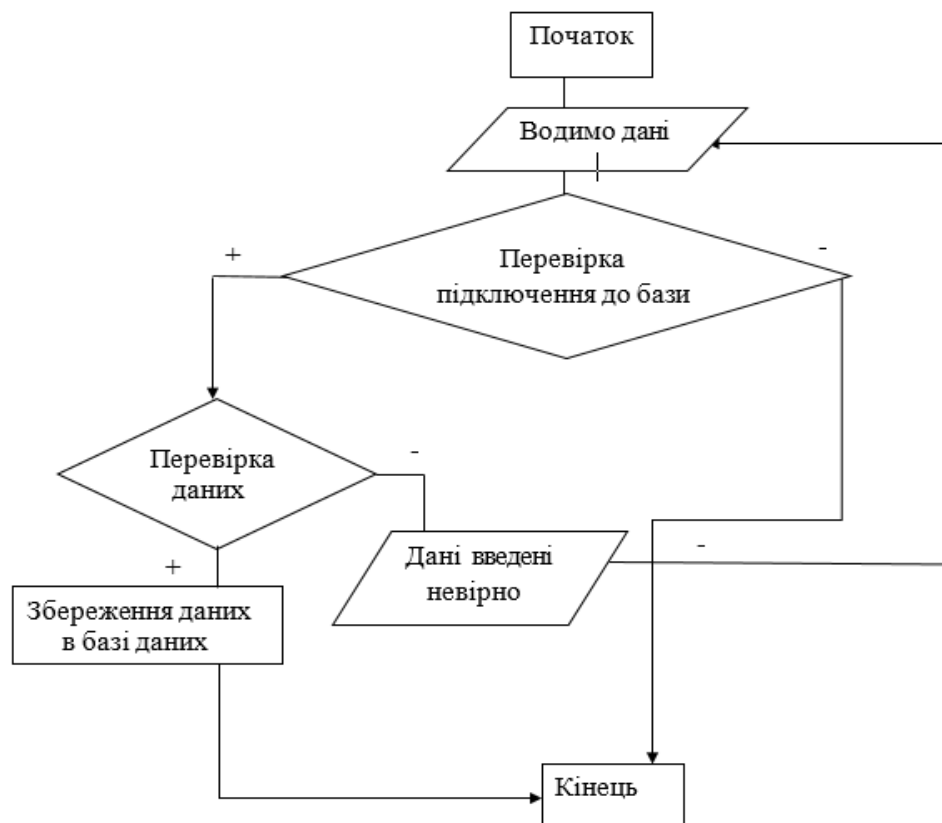


Рисунок 2.17 – Блок-схема реалізації реєстрації користувача

Реалізація входу користувача в програму за допомогою логіну та паролю
(див. рисунок 2.18).

Представимо алгоритм у вигляді команд:

- При виведенні даних для входу перевіряється підключення до бази даних. Якщо воно є – переходимо до наступного пункту, якщо його немає – виведення не відбудеться.
- В базі даних перевіряється наявність акаунту користувача, якщо він існує – переходимо до наступного пункту, якщо його немає – з'явиться повідомлення, що дані введені невірно .

- Користувачу стають доступними всі функції програми.
- Закінчити роботу.

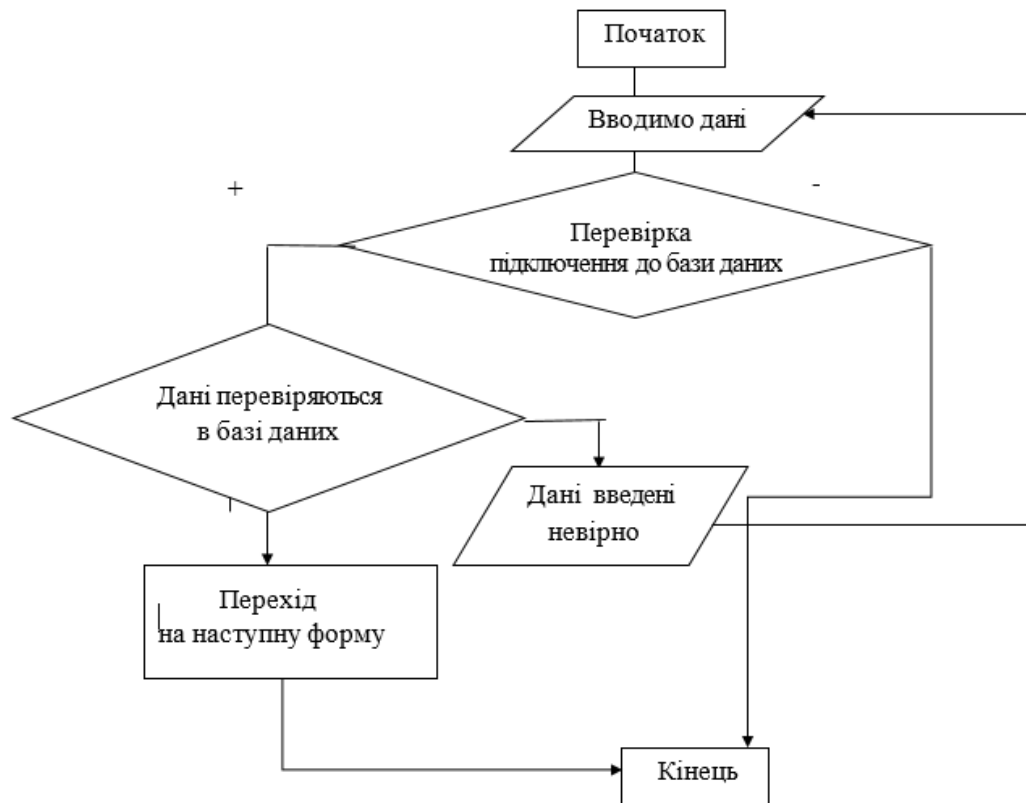


Рисунок 2.18 – Блок-схема реалізації входу в програму

Реалізація алгоритму калькулятор для підрахунків (див. рисунок 2.19).

Представимо алгоритм у вигляді команд:

- Користувач вводить перше число. Зберігаємо його в змінну A.
- Користувач вводить дію у вигляді символів.
+ (додавання),
- (віднімання),
* (множення),
/ (ділення).
- Запам'ятаємо команду в строкову змінну resultValue (дія).
- Користувач вводить друге число. Зберігаємо його в змінну B.
- Перевіряємо, яку дію потрібно виконати на основі вибору користувача:
якщо resultValue = додавання, то виконаємо $C = A + B$.
якщо resultValue = віднімання, то виконаємо $C = A - B$.

якщо $resultValue = \text{множення}$, то виконаємо $C = A * B$.

якщо $resultValue = \text{ділення}$, то виконаємо перевірку: чи можемо ми виконувати таке ділення:

якщо $B = 0$, виводимо повідомлення про помилку.

якщо $B \neq 0$, то виконуємо ділення $C = A / B$.

– Виводимо результат операції на екран.

– Запитуємо, закінчив користувач чи ні:

якщо закінчив, то виходимо з програми;

якщо не закінчив, то рахуємо наступне число.

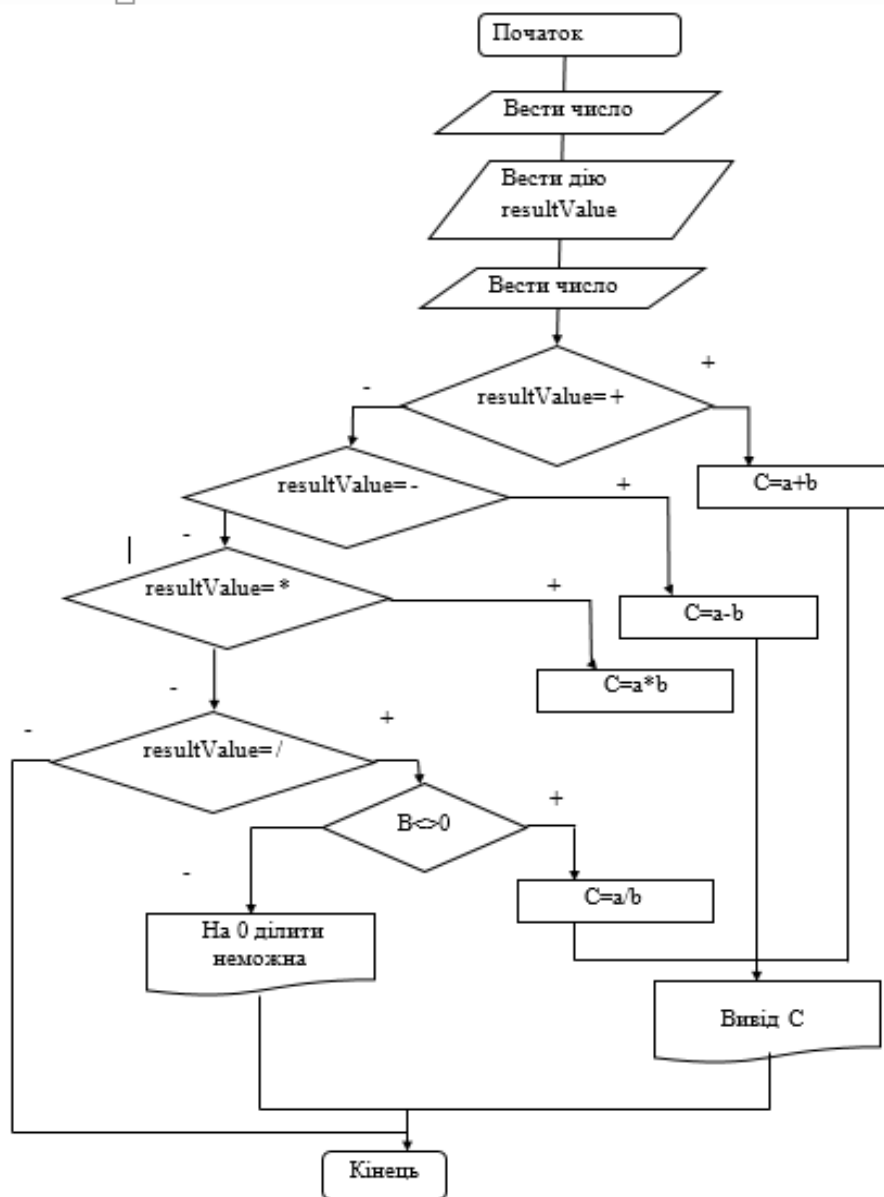


Рисунок 2.19 – Блок-схема реалізації алгоритму калькулятор для підрахунків

3 ОПИС ІНФОРМАЦІЙНОЇ СИСТЕМИ АДМІНІСТРАТОРА ОСВІТНІХ КУРСОВ

3.1 Вимоги до програмного продукту

Вимоги до програмного забезпечення – це сукупність характеристик, властивостей і функцій, якими має володіти створювана або вже розроблювана програмна система [24]. Вони формуються у процесі аналізу потреб і закріплюються у відповідних документах: специфікаціях вимог, діаграмах прецедентів та інших артефактах, що супроводжують етапи проєктування та розробки.

Процес формування вимог зазвичай поділяється на кілька послідовних етапів:

1. виявлення вимог (збір і визначення очікувань та потреб користувачів і зацікавлених сторін);
2. аналіз вимог (перевірка їх повноти, несуперечливості та узгодженості);
3. специфікація (формалізоване документування вимог);
4. верифікація та тестування (оцінка правильності сформульованих вимог).

Загалом можна виокремити дві основні категорії вимог: функціональні та нефункціональні. Функціональні вимоги описують внутрішню логіку системи, її поведінку та можливості, тобто визначають, що саме повинна виконувати програма [25].

Нефункціональні вимоги – це вимоги до програмного забезпечення, які визначають критерії оцінки його якості та ефективності. Якщо функціональні вимоги описують, що саме повинна виконувати система, то нефункціональні – вказують, якою вона має бути [26].

Бізнес-вимоги [27]:

Відомості про компанію: назва, перелік товарів і послуг, сфера діяльності, рік заснування, нагороди та здобутки, ключові конкуренти.

Цільова аудиторія:

- адміністратори освітніх курсів;
- чоловіки та жінки віком від 18 до 55 років;

- особи з вищою освітою;
- працівники, що активно взаємодіють із клієнтами;
- працюють у стандартному графіку з понеділка по п'ятницю з 9:00 до 18:00.

Основна проблема цільової аудиторії: необхідність автоматизації та спрощення роботи адміністратора.

Запропоноване рішення: використання функціоналу для запису та редагування даних користувачів дозволить уникнути потреби утримувати інформацію про слухачів у пам'яті.

Головні цілі програмного продукту:

1. Реалізація можливості створення записів і редагування інформації про слухачів.
2. Пошук даних про слухачів та додавання нотаток.

Орієнтовна структура форм

- Форма авторизації: реєстрація нового користувача та вхід у систему.
- Навігаційне меню: перехід між основними формами.
- Форми курсів: відображення інформації про навчальні програми.
- Керування користувачами: додавання та редагування даних слухачів.
- Модуль нотаток: створення та збереження коментарів.
- Калькулятор: допоміжний інструмент для обчислень.

Вимоги до дизайну та оформлення

- Стиль: оформлення в єдиній корпоративній концепції.
- Кольори: основні корпоративні – фіолетовий та ніжно-рожевий; додаткові – голубий і зелений для піктограм; білий використовується як базовий колір компонентів.
- Графіка: якісні та насичені фотографії, що ілюструють навчальні курси.
- Шрифти: перевага надається Arial, Microsoft Sans Serif або їхнім аналогам.
- Відображення: підтримка моніторів від 19 до 27 дюймів та ноутбуків із діагоналлю від 15,6 до 17,3 дюйма.

Приблизний набір модулів

- Фільтрація слухачів (за вартістю курсу, в алфавітному порядку).
- Форми курсів, розділені на два блоки:
 - ✓ Слухачі
 - ✓ Потенційні слухачі
- Калькулятор.
- Нотатки.
- Викладачі (з можливістю додавання та редагування інформації).

Інтеграція. Система повинна підтримувати інтеграцію з базою даних MySQL.

Вимоги до функціональності

Програмне забезпечення повинно забезпечувати такі можливості:

- реєстрація нового користувача;
- авторизація за індивідуальним паролем;
- перегляд інформації про курси;
- відображення даних слухачів і викладачів;
- додавання та редагування даних користувачів (слухачів і викладачів);
- створення та редагування нотаток;
- робота з калькулятором;
- пошук слухачів;
- сортування даних за алфавітом.

Мета створення програми. Розроблюване програмне забезпечення має на меті:

- забезпечити збір та первинну обробку інформації, необхідної для роботи адміністратора освітніх курсів;
- сформувати єдину систему звітності за ключовими показниками діяльності;
- підвищити якість даних за такими параметрами: повнота, точність, достовірність, своєчасність та узгодженість;
- скоротити час, необхідний на збір і початкову обробку інформації;

- зменшити кількість окремих інформаційних систем, які застосовуються для формування аналітичних звітів;
- оптимізувати витрати часу на інформаційно-аналітичну діяльність.

Функціональні підсистеми. У межах системи пропонується виділити такі підсистеми:

- Підсистема збору, обробки та завантаження даних – відповідає за отримання даних із зовнішніх джерел, їх трансформацію та підготовку для збереження у внутрішній базі.
- Підсистема зберігання даних – забезпечує структурування та збереження інформації для подальшого використання у процесі прийняття рішень.
- Підсистема формування та візуалізації звітності – призначена для створення бізнес-орієнтованих звітів та інтерактивних інформаційних панелей.

Режими функціонування системи. Система повинна підтримувати два основні режими роботи:

- Основний режим – стандартна робота з повним виконанням усіх функцій.
- Профілактичний режим – один або кілька модулів тимчасово не виконують свої завдання у зв'язку з технічним обслуговуванням.

Функціонування у основному режимі. У штатному режимі система повинна забезпечувати:

- безперервну роботу користувачів за принципом 24/7 (24 години на добу, 7 днів на тиждень);
- виконання всіх ключових функцій, включно зі збором, обробкою, завантаженням і збереженням даних.

3.2 Структура бази даних

База даних інформаційної системи адміністратора освітніх курсів складається з кількох основних таблиць, які забезпечують зберігання та обробку необхідної інформації:

- «Слухачі»

- «Курси»
- «Авторизація»
- «Нотатки»
- «Викладачі»

Таблиця «Слухачі» містить дані про користувачів-слухачів курсів:

- ідентифікаційний номер;
- прізвище, ім'я, по батькові (ПІБ);
- дата реєстрації або народження (уточнюється контекстом);
- електронна пошта;
- номер телефону;
- назва або код курсу;
- вартість курсу;
- додаткова інформація;
- бажаний час навчання;
- статус актуальності.

Таблиця «Курси» містить опис навчальних програм:

- ідентифікаційний номер курсу;
- назва курсу;
- тривалість навчання (кількість годин).

Таблиця «Авторизація» використовується для зберігання даних користувачів системи:

- код авторизації;
- ім'я користувача;
- прізвище користувача;
- електронна пошта;
- нікнейм;
- пароль.

Таблиця «Нотатки» зберігає додаткову інформацію, яку вводить користувач:

- код нотатки;
- текстова інформація;

- дата створення.

Таблиця «Викладачі» містить інформацію про викладачів курсів:

- код викладача;
- ім'я та прізвище викладача;
- стаж роботи;
- код курсу (зв'язок із таблицею «Курси»);
- номер телефону;
- електронна пошта.

Нижче представлені таблиці бази даних (див. таблиці 3.1-3.5)

Таблиця 3.1 – «Курси» (Courses)

Назва поля	Тип поля	Опис поля
IdCourses.	Int	Код курсу
NameCourses	Char	Назва курсу
TrainingTim	Int	Кількість годин навчання

Таблиця 3.2 – «Слухачі» (Clients)

Назва поля	Тип поля	Опис поля
IdCarat.	Int	Ідентифікаційний номер слухача
Name	Char	ПІБ
Date	Date	Дата реєстрації або народження
Phone	Numeric	Телефон слухача
Email	Char	Електронна пошта слухача
TrainingTime	Char	Бажаний час навчання
Courses	Int	Код курсу
Information	Text	Додаткова інформація
Topicality	Char	Актуальність
Price	Money	Ціна курсу грн.

Таблиця 3.3 – «Авторизація» (LoginTbl)

Назва поля	Тип поля	Опис поля
Auto_ID.	Int	Код авторизації
First_Name	nvarchar	Ім'я користувача
Last_Name	nvarchar	Прізвище користувача
Email_Address	nvarchar	Електрона пошта користувача
Username	nvarchar	Нікнейм користувача
Password	nvarchar	Пароль користувача

Таблиця 3.4 – «Нотатки» (NoteBook)

Назва поля	Тип поля	Опис поля
IdNoteBook	Int	Код нотатки
TextNoteBook	Text	Інформація
Date	Date	Дата створення

Таблиця 3.5 – «Викладачі» (Teachers)

Назва поля	Тип поля	Опис поля
IdTeacher	Int	Код викладача
NameTeache	char	Ім'я та прізвище викладача
Experience	Int	Стаж роботи
IdCourses	Int	Код курсу
Phone	Numeric	Телефон викладача
Email	Char	Електрона пошта викладача

Схема бази даних [30] зображена на рисунку 3.1.

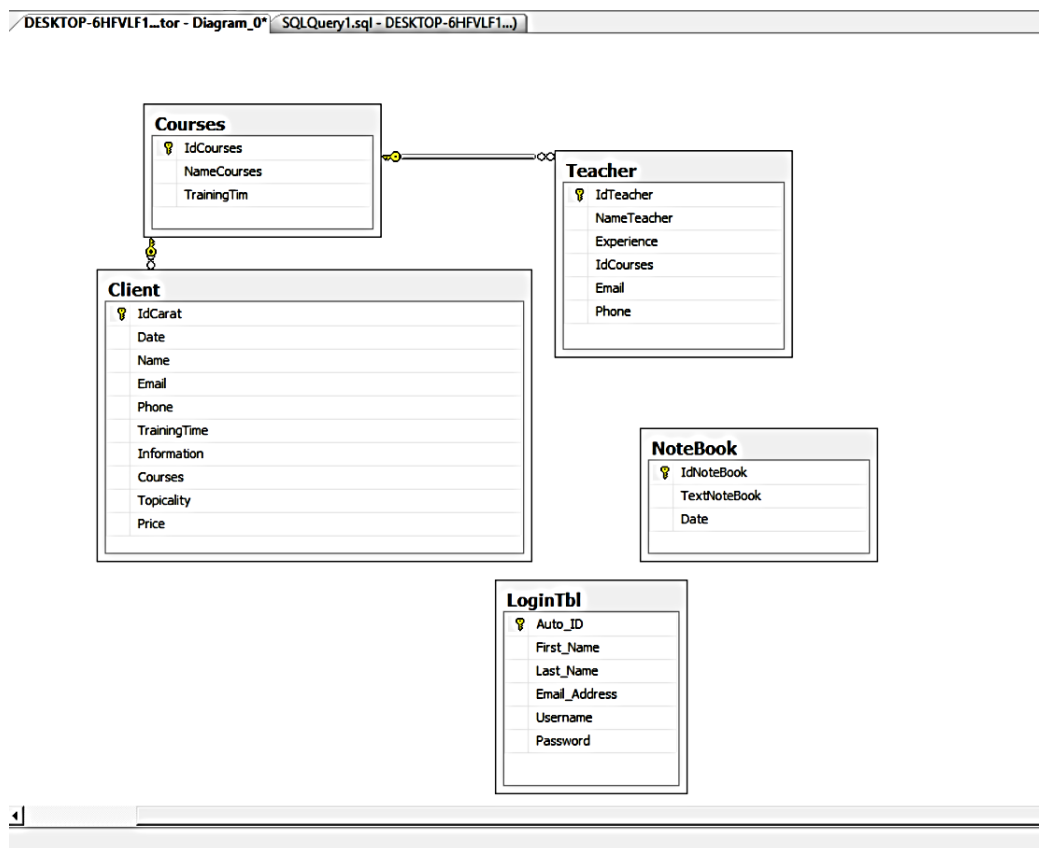


Рисунок 3.1 – Схема бази даних

У межах програмного продукту формується картотека, що містить відомості про такі об'єкти:

- слухачі курсів;
- навчальні курси;
- користувачі системи;
- вартість навчання;
- бажаний час відвідування занять;
- додаткова інформація про слухачів;
- викладачі;
- робочі записи (нотатки).

Опис вихідних даних. Програма передбачає можливість перегляду статистичної інформації, зокрема:

- дані про слухачів;
- дані про курси;

- інформацію щодо бажаного часу навчання слухачів;
- відомості про вартість навчання;
- додаткову інформацію, зазначену користувачами;
- інформацію про викладачів;
- дані про робочі записи.

3.2 Створення макетів інформаційної системи

На рисунках 3.2-3.5 наведені макети інформаційної системи адміністратора освітніх курсів.

На макеті Форма реєстрації (див. рисунок 3.2) зображено:

1. Елементи введення даних користувача – поля для внесення особистої інформації (ПІБ, електронна пошта, телефон, бажаний час навчання та інші необхідні параметри).
2. Елементи керування – функціональні кнопки для роботи з даними (збереження введеної інформації, очищення текстових полів, повернення на форму авторизації або вхід у програму).
3. Таблиця відображення даних користувачів – інтерфейсний елемент, що забезпечує перегляд зібраної інформації у зручному табличному форматі.

The wireframe shows a registration form layout. At the top, labeled '1', is a group of six text input fields arranged in two columns of three. Below this, labeled '2', is a row of four rectangular buttons. At the bottom, labeled '3', is a large rectangular area representing a table for displaying user data.

Рисунок 3.2 – Макет Форми реєстрації

Макет Входу в інформаційну систему (див. рисунок 3.3)

1. Форма введення даних для авторизації – поля для введення логіна та пароля користувача.
2. Елементи навігації – кнопки для входу до системи та переходу до форми реєстрації нового користувача.

 A mockup of a login form. It consists of two rectangular input fields stacked vertically, each containing the number '1'. Below these fields are two rectangular buttons, each containing the number '2'. The entire form is set against a light gray background.

Рисунок 3.3 – Макет Входу в інформаційну систему

Макет Форми головного меню (див. рисунок 3.4)

1. Система навігації між формами – забезпечує швидкий доступ до основних розділів програми.
2. Механізм переходу між курсами – дозволяє користувачу переглядати доступні навчальні програми та обирати потрібний курс.

 A mockup of a main menu interface. On the left is a vertical sidebar with a blue header bar and a list of five items, each labeled with the number '1'. To the right of the sidebar is a main content area containing four rectangular buttons arranged in a 2x2 grid, each labeled with the number '2'. The interface is presented on a light gray background.

Рисунок 3.4– Макет Форми головного меню

Макети Форм для слухачів, викладачів, нотатки (див. рисунок 3.5)

1. Реалізація навігації між формами.
2. Виведення та обробка даних із бази SQL.
3. Представлення текстової інформації.

The diagram illustrates a web form layout with three numbered components:

- 1**: A vertical sidebar on the left containing a list of four items, each represented by a light gray rectangle with the number '1' inside.
- 2**: A large rectangular area in the top right, representing a main content or data display area.
- 3**: Four small rectangular input fields arranged in a 2x2 grid at the bottom right, each containing the number '3'.

Рисунок 3.5 – Макети Форм для слухачів, викладачів, нотаток

1.4 Екранні форми інформаційної систкми

Форма «Реєстрація у системі».

Дана форма забезпечує можливість створення нового облікового запису користувачем для подальшої роботи в системі (див. рисунок 3.6). Реалізація здійснена шляхом взаємодії з базою даних, у яку заносяться відомості про нового користувача. Передбачено обов'язкове заповнення всіх текстових полів, а також перевірку відповідності пароля та його підтвердження. Крім того, передбачена функція видалення запису.

The screenshot shows a web application window titled "Реєстрація у системі". It contains a registration form with the following fields: "Ім'я:" (Name), "Ім'я користувача" (Username), "Прізвище:" (Surname), "Пароль:" (Password), "E-mail:", and "Повторно введіть:" (Repeat). Below the form is a toolbar with icons and labels: "Видалити" (Delete), "Очистити" (Clear), "Зберегти" (Save), and "Закрити" (Close). The main content area below the toolbar is empty.

Рисунок 3.6 – Форма «Реєстрація у системі»

Форма «Вхід в систему».

Ця форма надає можливість входу до системи за допомогою логіна та пароля (див. рисунок 3.7). У випадку невідповідності введених даних записам у базі, користувач отримує повідомлення про помилку. Також реалізована опція перегляду введеного пароля для зручності.

The screenshot shows a web application window titled "Вхід у аккаунт". It contains a login form with the following fields: "Ім'я користувача" (Username) with the value "Ольга", and "Пароль" (Password) with masked characters. There is a checkbox labeled "Показати пароль" (Show password). Below the form are two buttons: "Увійти" (Login) and "Вихід" (Logout). At the bottom, there is a link: "У вас ще немає облікового запису? Створити аккаунт" (Don't you have an account yet? Create an account). The form is decorated with an illustration of a person sitting at a desk with a laptop and a smartphone.

Рисунок 3.7 – Форма «Вхід в систему»

Форма «Головне меню».

У цьому діалоговому вікні користувач може здійснювати вибір перегляду слухачів за курсами (див. рисунок 3.8). Панель навігації забезпечує доступ до додаткових функцій: нотаток, калькулятора, перегляду викладачів, а також виходу з облікового запису чи завершення роботи програми.

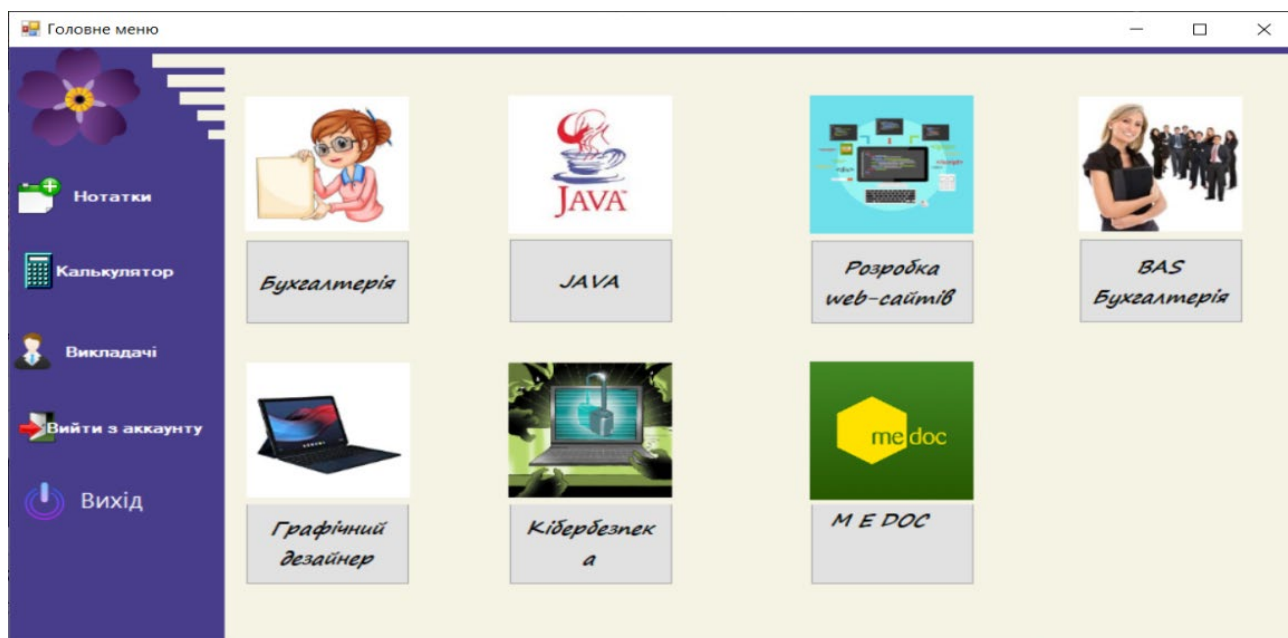


Рисунок 3.8 – Форма «Головне меню»

Форма курсів

У діалоговому вікні перегляду слухачів передбачено відображення таблиці зі списком слухачів, яка структурована на три розділи: «Потенційні слухачі», «Слухачі» та «Всі слухачі». Така класифікація забезпечує зручність у навігації та швидкий доступ до потрібних даних.

Для кращого ознайомлення користувача з навчальним процесом у вікні додатково представлена коротка інформація про відповідний курс.

Панель навігації надає можливість виконувати основні операції: перехід до вікон редагування, додавання та пошуку слухачів. Також реалізовані функції повернення до попереднього вікна та завершення роботи із системою.

На рисунках 3.9 – 3.15 наведені приклади форм для різних курсів.

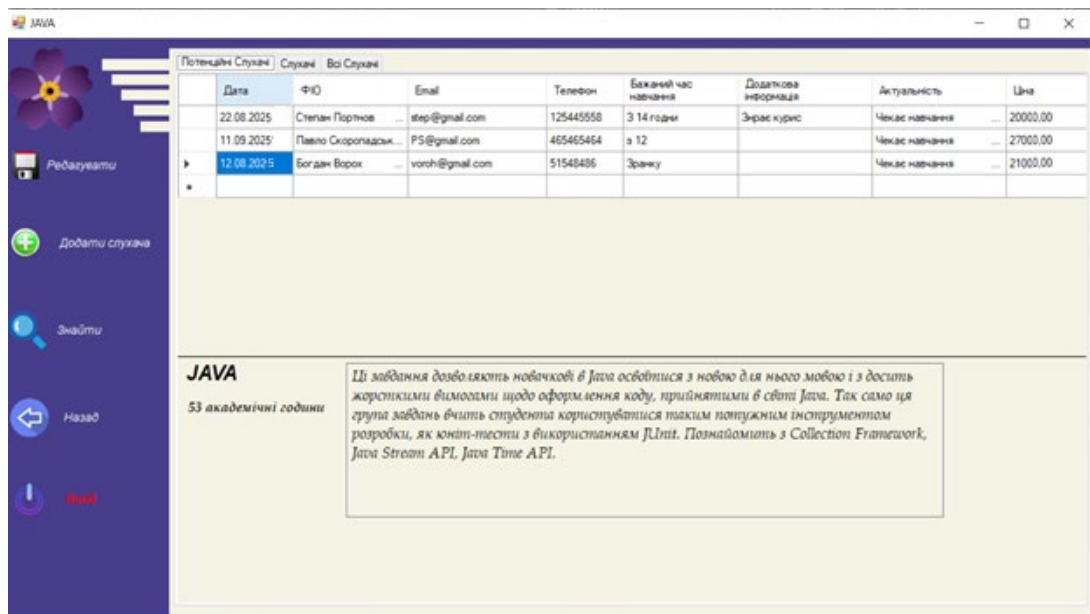


Рисунок 3.9 – Форма курсів «Java»

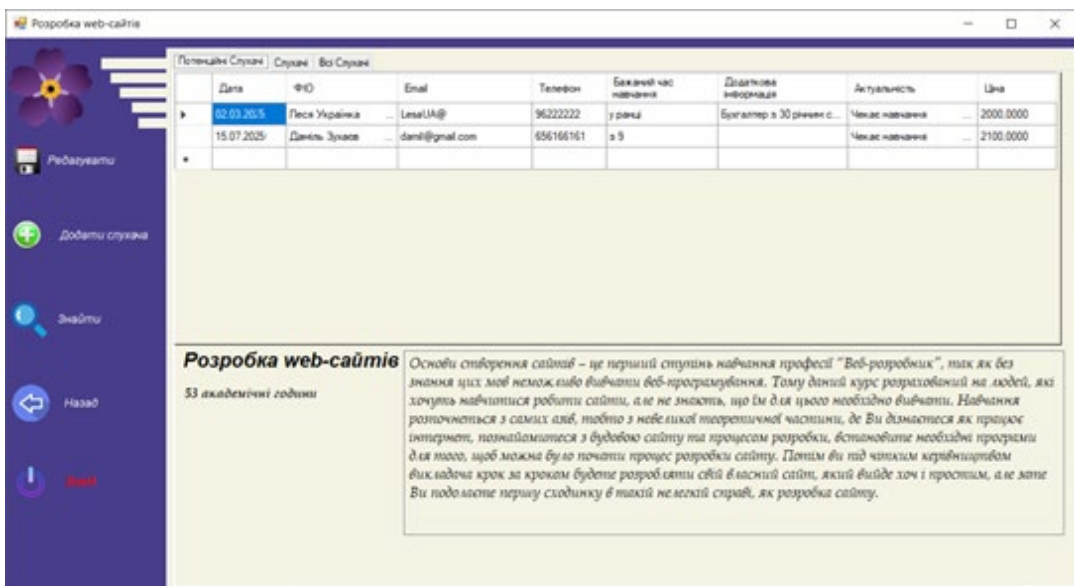


Рисунок 3.10 – Форма курсів «Розробка web-сайтів»

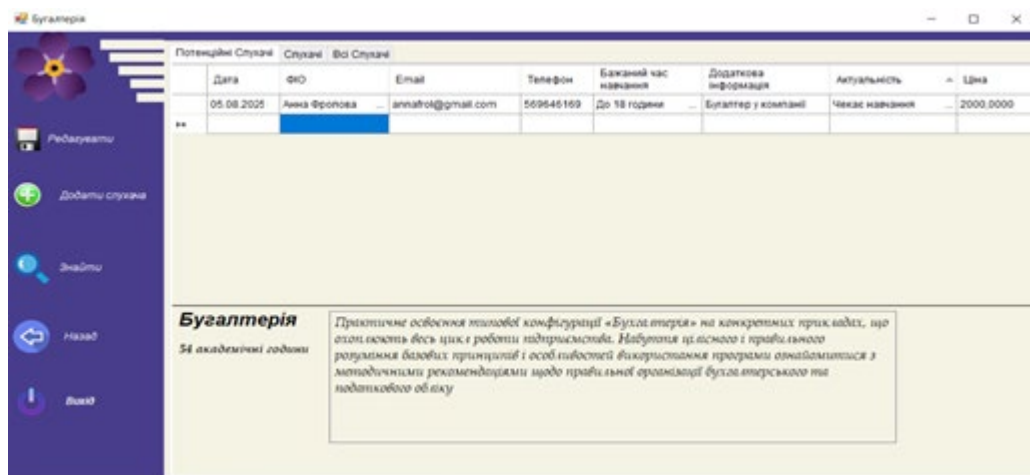


Рисунок 3.11 – Форма курсів «Бухгалтерія»

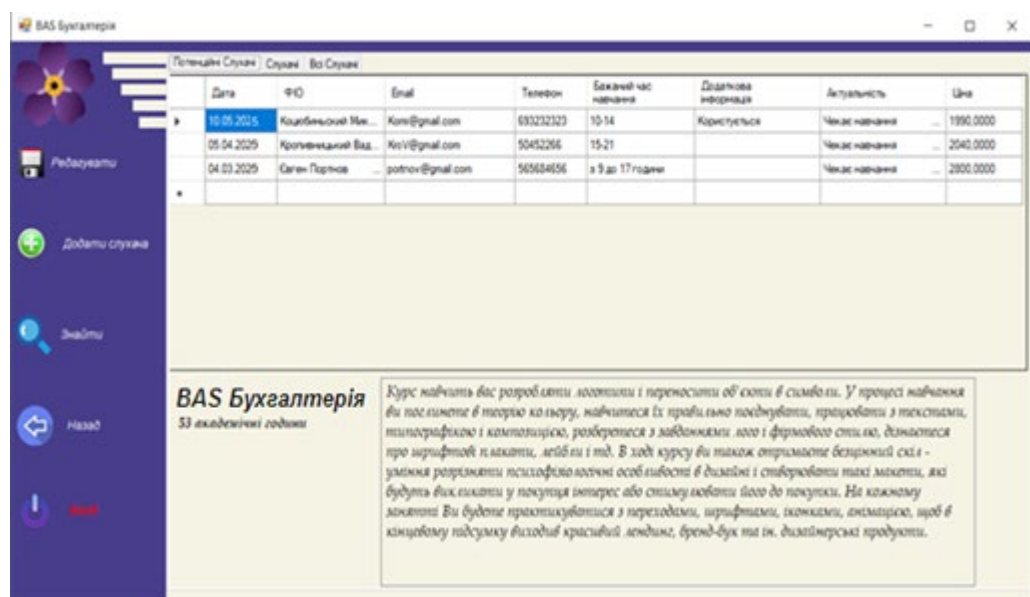


Рисунок 3.12 – Форма курсів «BAS Бухгалтерія»

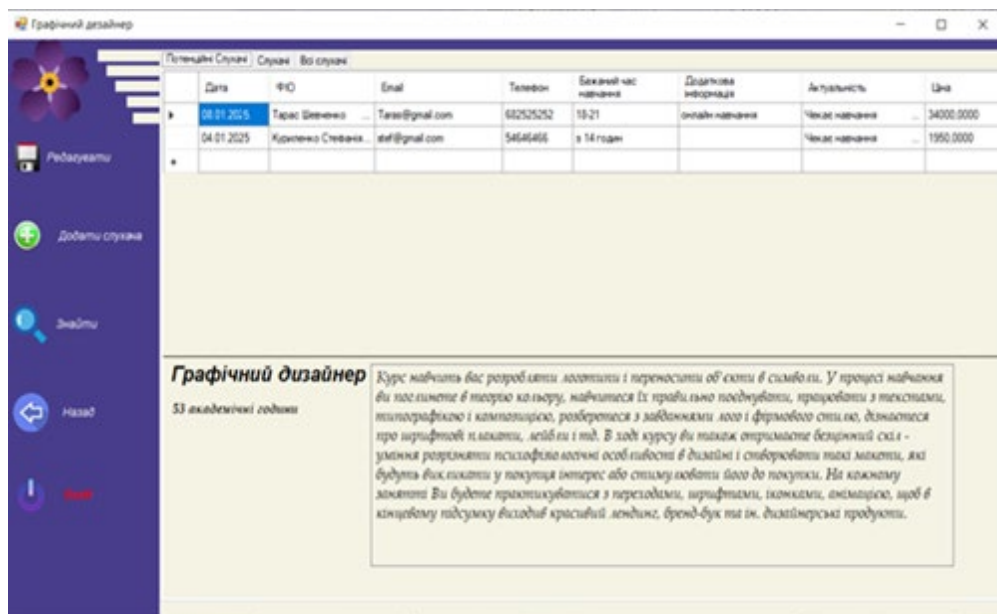


Рисунок 3.13 – Форма курсів «Графічний дизайнер»

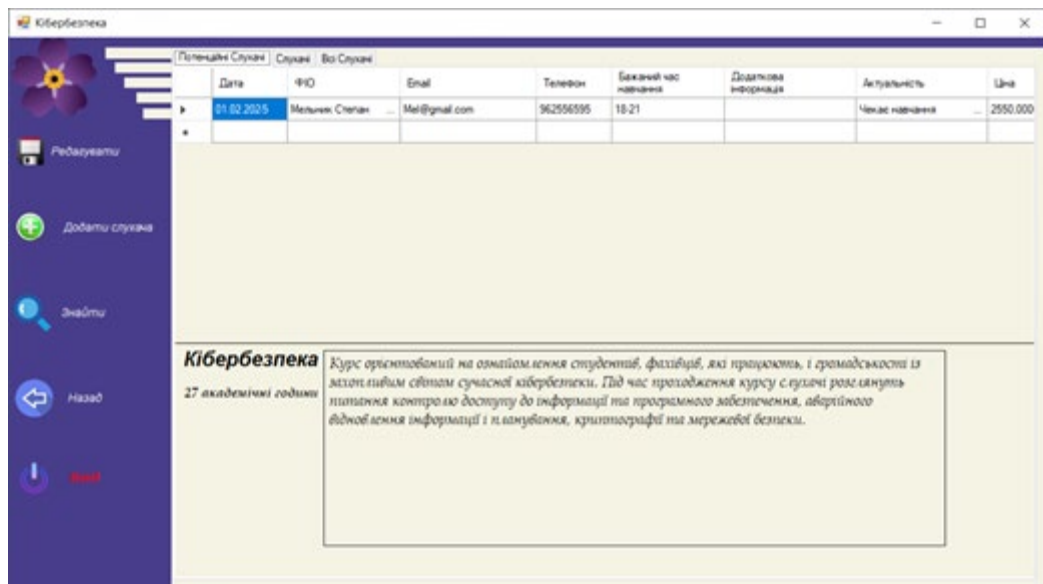


Рисунок 3.14 – Форма курсів «Кибербезпека»

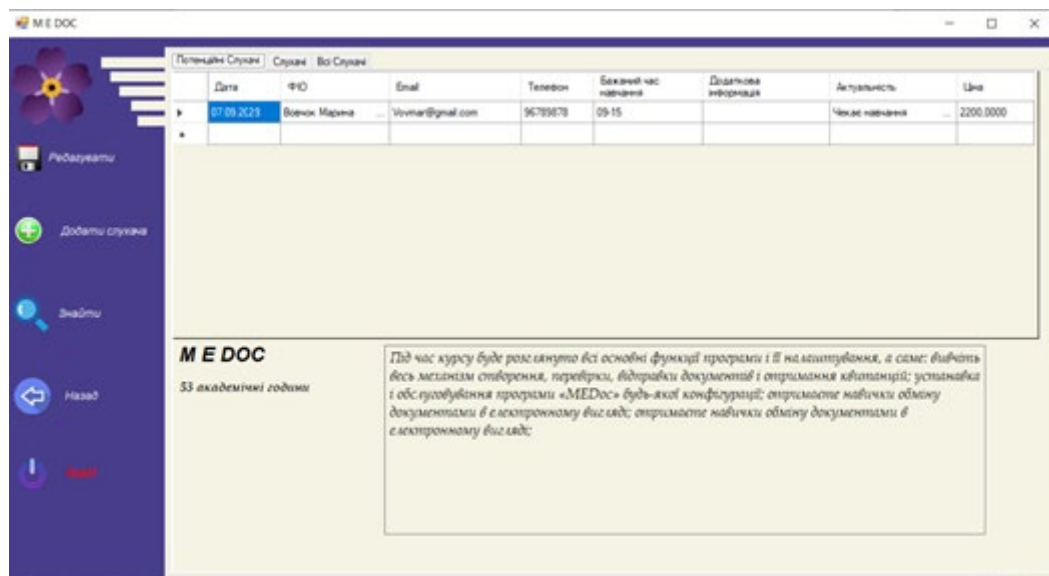


Рисунок 3.15 – Форма курсів «M.E.DOC»

Форма «Редагування/Додавання»

У формі роботи зі слухачами реалізовано можливість керування даними про слухачів: редагування існуючих записів, додавання нових користувачів та видалення непотрібних (див. рисунок 3.16).

Панель навігації забезпечує доступ до основних функцій системи, зокрема: редагування, додавання та пошуку слухачів. Також передбачені опції повернення до попереднього вікна та виходу з програми.

The screenshot shows a software window titled 'Редагування' (Editing). It features a table with 10 columns: IdCard, Date, Name, Email, Phone, TrainingTime, Information, Courses, and Topicality. The table contains 14 rows of student data. Below the table is a form for adding or editing a student, with fields for Code, Date, Actuality, FIO, Email, Phone, Course, and Price. The form also includes a section for 'Додаткова інформація' (Additional information) with a text area.

IdCard	Date	Name	Email	Phone	TrainingTime	Information	Courses	Topicality
1	05.08.2025	Анна Фролова ...	annafo@gmail.c...	569646169	До 18 годени ...	Бухгалтер у комп...	1	Чекас навчання... 2
3	22.08.2025	Степан Портнов...	step@gmail.com ...	125445558	3 14 годени ...	Зв'язок курси...	2	Чекас навчання... 2
6	02.01.2025	Карпенко Карий...	kar@gmail.com ...	51546466	з 12	апр	1	Навчання... 2
7	15.03.2025	Валентин Том...	wala@gmail.com ...	56564656	3 10 до 17 год...	Длмас про дода...	2	Навчання... 2
9	02.03.2025	Лєся Українка	LesiaUA@...	96222222	у ранці	Бухгалтер з 30 ...	3	Чекас навчання... 2
10	08.01.2025	Тарас Шевченк...	Taras@gmail.co...	682525252	18-21	онлайн навчання	4	Чекас навчання... 3
11	10.05.2025	Козобинський ...	Koti@gmail.com ...	693232323	10-14	Користується	5	Чекас навчання... 1
12	07.09.2025	Вовчок Марина ...	Vovmar@gmail.c...	96789878	09-15		6	Чекас навчання... 2
13	01.02.2025	Мельник Степа...	Mel@gmail.com ...	96256595	18-21		7	Чекас навчання... 2
14	09.04.2025	Кристаліна Кат...	Kris@gmail.com ...	50457812	9-16	Група вищогого ...	8	Чекас навчання... 3

Рисунок 3.16 – Форма «Редагування/Додавання»

Вікно «Знайти слухача»

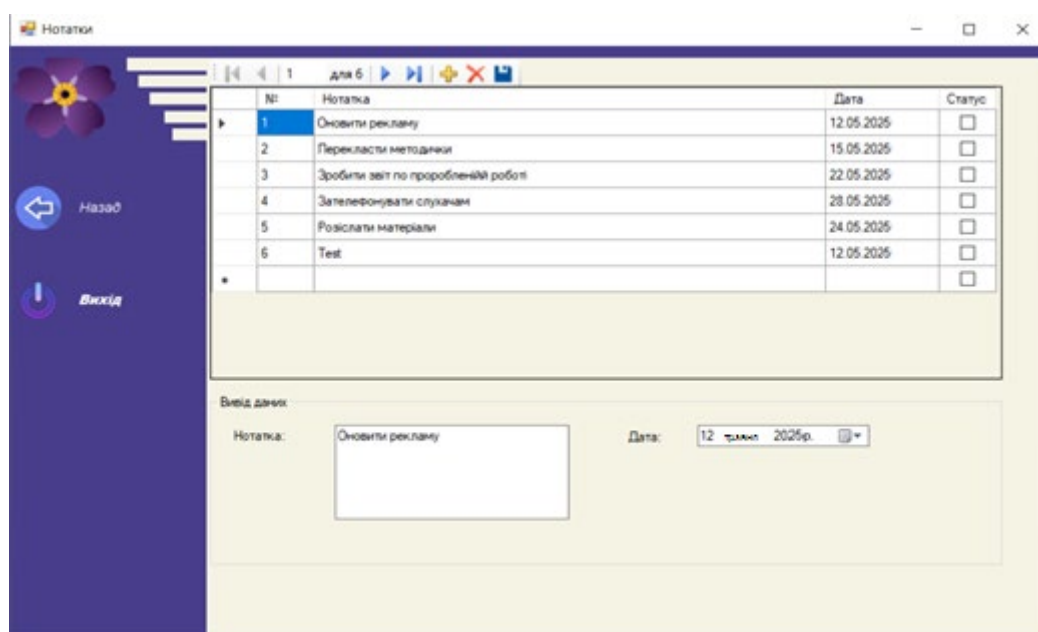
Дане вікно призначене для здійснення пошуку конкретного слухача серед даних, відображених на поточній формі. Реалізовано механізм автоматичного виділення знайденого елемента, що підвищує зручність користування системою та швидкість доступу до потрібної інформації (див. рисунок 3.17).

The screenshot shows a window titled 'SearchForm'. It contains a search form with the title 'Пошук' (Search) and the instruction 'Введіть дані для пошуку' (Enter data for search). There is a text input field for the search query. Below the input field are two buttons: 'Знайти' (Find) and 'Закрити' (Close).

Рисунок 3.17 – Вікно «Знайти слухача»

Форма «Нотатки»

Дана форма призначена для збереження текстової інформації у базі даних з подальшою можливістю її використання адміністраторами системи (див. рисунок 3.18).



№	Нотатка	Дата	Статус
1	Оновити рекламу	12.05.2025	☐
2	Перекласти методички	15.05.2025	☐
3	Зробити звіт по проробленій роботі	22.05.2025	☐
4	Зателефонувати слухачам	28.05.2025	☐
5	Розіслати матеріали	24.05.2025	☐
6	Test	12.05.2025	☐
*			☐

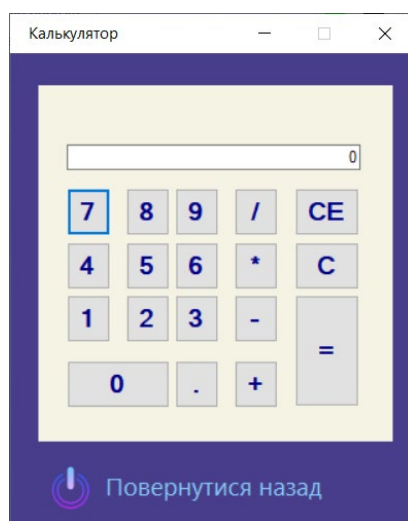
Вивід даних:

Нотатка: Дата:

Рисунок 3.18 – Форма «Нотатки»

Форма «Калькулятор»

Форма реалізує базові арифметичні операції, необхідні для виконання простих обчислень у межах системи (див. рисунок 3.19).



Калькулятор

0

7 8 9 / CE

4 5 6 * C

1 2 3 - =

0 . +

Повернутися назад

Рисунок 3.19 – Форма «Калькулятор»

Форма «Викладачі»

У цій формі реалізовано роботу з даними про викладачів. Користувач має можливість додавати нові записи, редагувати існуючі та видаляти непотрібні. Усі відомості зберігаються в базі даних. Додатково реалізовано функцію пошуку викладача за його ідентифікатором (див. рисунок 3.20).

Код	ФІО викладача	Стаж	Курс	Email	Телефон
1	Демченко Вал...	4	1	dem@gmail.com ...	2888245
2	Котенко Сергій ...	5	3	Cot@gmail.com ...	50287168
3	Пичар Валенти...	10	5	Vfn@gmail.com ...	962585282

Код	1
ФІО	Демченко Валерій
Стаж	4
Курс	1
Email	dem@gmail.com
Телефон	2888245

Рисунок 3.20 – Форма «Викладачі»

Описані у даному розділі форми забезпечують повноцінну взаємодію користувача із системою та реалізують усі необхідні функціональні можливості. Форма «Реєстрація» та «Авторизація» гарантують безпечний доступ до системи, забезпечуючи збереження даних користувачів. Головне меню виступає центральним елементом навігації, що поєднує всі ключові модулі.

Форми, пов'язані з роботою із слухачами та викладачами, дозволяють ефективно здійснювати пошук, додавання, редагування та видалення записів, а також структурувати дані у базі. Додаткові можливості, такі як «Нотатки» та «Калькулятор», розширюють функціональність системи та підвищують зручність її використання.

У сукупності реалізовані форми створюють інтуїтивно зрозумілий та функціонально насичений інтерфейс, що відповідає вимогам користувача і забезпечує комплексне управління інформаційними процесами.

3.5 Тестування операційної системи

В випадку, якщо адміністратор не введе ім'я (див. рисунок 3.21),

Full Name	Username
TANIN SANGGAM	TaninS
SANGGAM TANIN	SangnagamT
Sofi Dovg	Sofi
Ольга Довгенко	Ольга

Рисунок 3.21 – Введення даних без імені адміністратора

програма видає повідомлення про необхідність ввести ім'я (див. рисунок 3.22)

Рисунок 3.22 – Повідомлення про необхідність ввести ім'я

В випадку, коли паролі не співпадають (див. рисунок 3.23),

Full Name	Username
TANIN SANGGAM	TaninS
SANGGAM TANIN	SangnagamT
Sofi Dovg	Sofi
Ольга Довгенко	Ольга

Рисунок 3.23 – Введення паролів, що не співпадають

з'являється повідомлення, що пароль та пароль підтвердження не збігаються (див. рисунок 3.24).

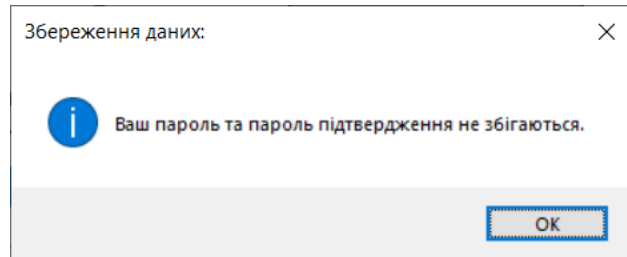


Рисунок 3.24 – Повідомлення про паролі

При вірно введених даних (див. рисунок 3.25)

Full Name	Username
TANIN SANGNGAM	TaninS
SANGNGAM TANIN	SangngamT
Sofi Dovg	Sofi
Ольга Довгенко	Ольга

Рисунок 3.25 – Введення вірно всіх даних

з'являється повідомлення з питанням, чи зберігати запис (див. рисунок 3.26), та повідомлення, що запис успішно збережено (див. рисунок 3.27), а також проводиться реєстрація в системі (див. рисунок 3.28).

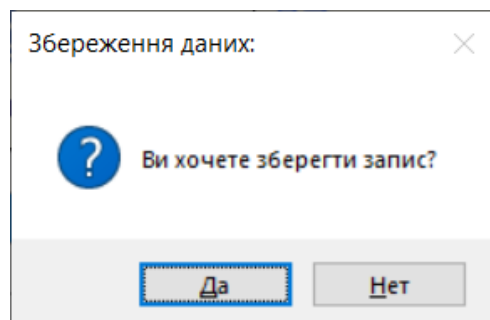


Рисунок 3.26 – Повідомлення з питанням, чи зберігати запис

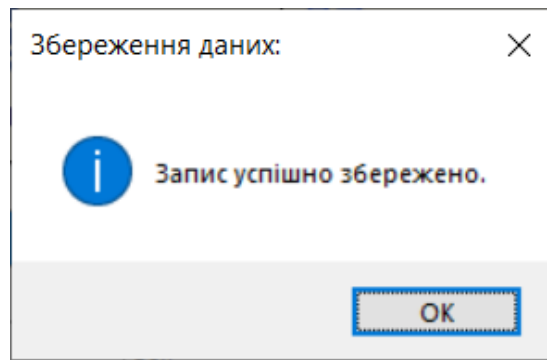


Рисунок 3.27 – Повідомлення, що запис успішно збережено

Реєстрація у системі

Ім'я: Ім'я користувача

Прізвище: Пароль:

E-mail: Повторно введіть:

 Видалити
  Очистити
  Зберегти
  Закрити

	Full Name	Username
	TANIN SANGNGAM	TaninS
	SANGNGAM TANIN	SangnagamT
	Sofi Dovg	Sofi
	Ольга Довгенко	Ольга
▶	Інна Коваленко	Кова

Рисунок 3.28 – Проведення реєстрації в системі

Було протестовано також створення нової нотатки (див. рисунок 3.29)

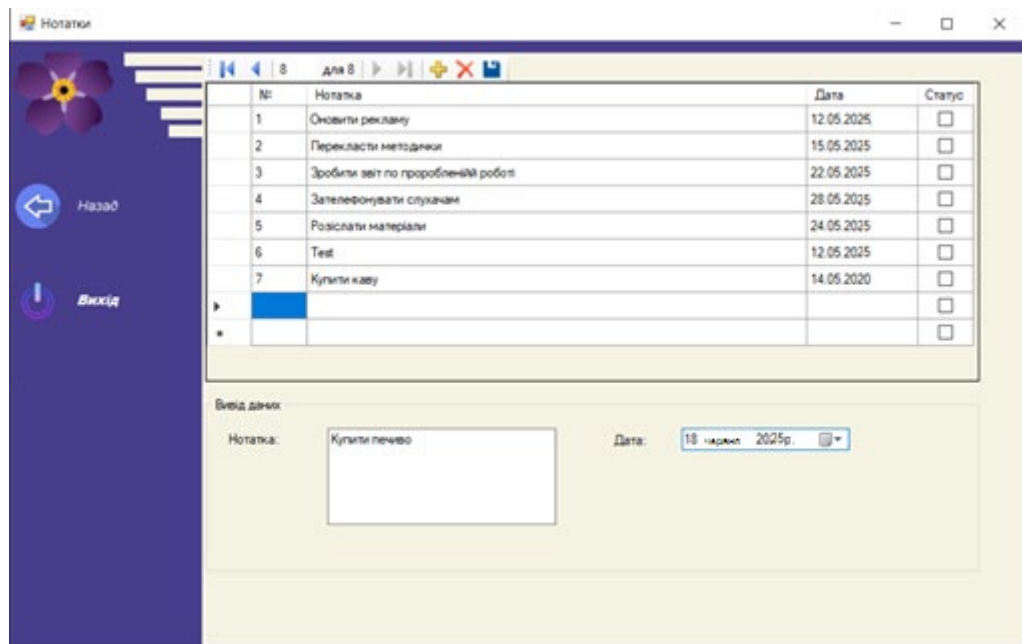


Рисунок 3.29 – Нова нотатка

Збереження даних в нотатці (див. рисунок 3.30)

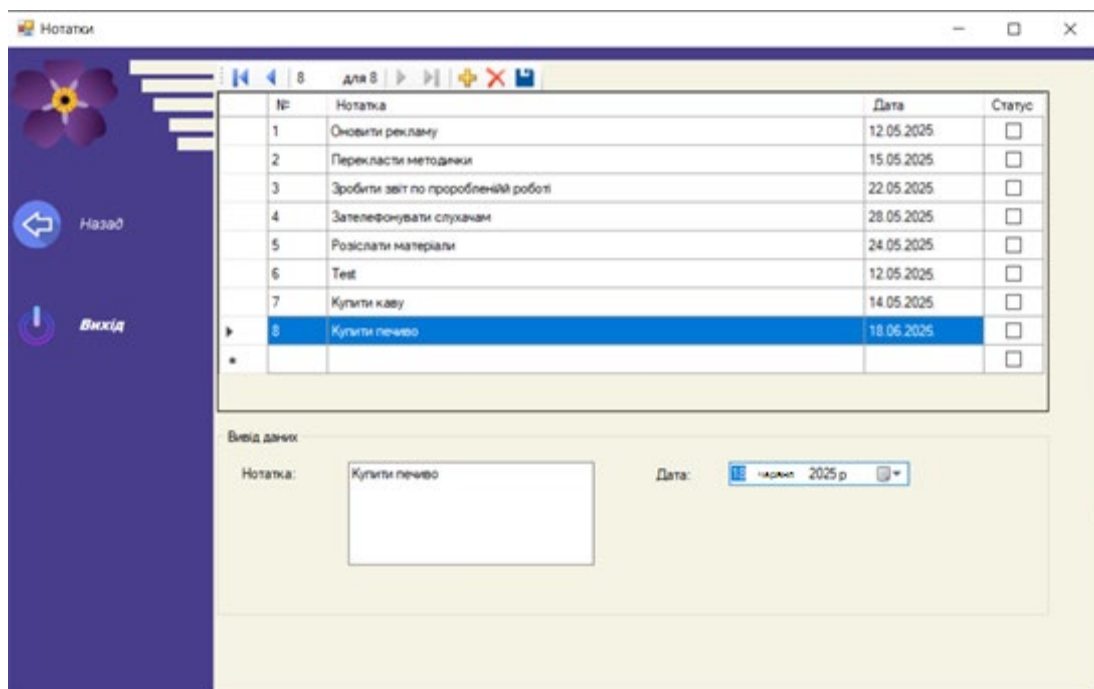


Рисунок 3.30 – Збережені дані в нотатці

ВИСНОВКИ

Мета атестаційної роботи магістра полягала в розробці інформаційної системи адміністратора освітніх курсів для полегшення його роботи. Інформаційна система повинна має зручний та інтуїтивно зрозумілий інтерфейс і надає можливість адміністратору переглядати та редагувати інформацію.

Для її реалізації було зроблено:

- Досліджено роль і завдання освітніх курсів в навчальному процесі. Проведено огляд та аналіз наявних програмних продуктів, орієнтованих на адміністрування, виявлено їх функціонал, переваги та недоліки.
- За допомогою об'єктно-орієнтованої мови програмування C# з використанням СКБД MySQL в середовищі Microsoft Visual Studio було створено інформаційну систему адміністратора освітніх курсів із інтуїтивно зрозумілим і дружнім інтерфейсом, що робить її зручною для будь-якої категорії користувачів, які займаються адмініструванням освітніх курсів. Ця система дає адміністратору можливість додавати нові дані про слухачів, редагувати та видаляти їх, а також здійснювати перегляд інформації. Завдяки цьому адміністратор навчального центру може ефективно працювати з базою слухачів, оперативно зв'язуватися з ними для уточнення організаційних питань. У процесі розробки були реалізовані модулі автентифікації, внесення та збереження даних, перегляду інформації та пошуку по слухачах. Кожен із цих модулів було адаптовано під специфіку роботи адміністратора та інтегровано в систему.
- Протестестована робота інформаційної системи. Результати тестування підтвердили працездатність продукту та його готовність до подальшого застосування.

Практична цінність атестаційної роботи бакалавра полягає в створенні інформаційної системи адміністратора освітніх курсів з метою полегшенні його роботи. Розроблена інформаційна система виключить рутину роботи, а також буде запобігати людським помилкам.

У перспективі вдосконалення програмного забезпечення можливим напрямком є розробка модулів управління доступом, а також покращення інтерфейсу сайту для підвищення його інформативності, привабливості та зручності використання.

ПЕРЕЛІК ПОСИЛАНЬ

1. Загальна характеристика професії "АДМІНІСТРАТОР". [Електронний ресурс]. – URL: <https://kha.dcz.gov.ua/publikaciya/zagalna-harakterystyka-profesiyi-administrator>
2. jSolutions. [Електронний ресурс]. – URL: <https://gigacloud.ua/partners/jsolutions>
3. Автоматизація. [Електронний ресурс]. – URL: <http://servio.ru/automation-of-hotels-guest-houses->
4. Адміністратор салону. [Електронний ресурс]. – URL: <http://salonmanager.com/>
5. Платформа для автоматизації бізнесу. [Електронний ресурс]. – URL: <https://fosslook.com.ua/>
6. Ресепшен. [Електронний ресурс]. – URL: <https://play.google.com/store/apps/details?id=com.experimentalui.apps.reception&hl=uk>
7. Bumpix. [Електронний ресурс]. – URL: <https://bumpix.net/page/get-started>
8. Мюллер Дж. П. C# для чайников. – Київ: Диалектика, 2018. – 608 с.
9. Шарп Дж. Microsoft Visual C#. Подробное руководство. – Санкт-Петербург: Питер, 2017. – 848 с.
10. Ішкова Е. C# начала программирования. – Москва: Бином-Пресс, 2010. – 336 с.
11. Создание приложения Windows Forms на C#. [Електронний ресурс]. – URL: <https://docs.microsoft.com/ru-ru/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2019>
12. Metanit. [Електронний ресурс]. – URL: <https://metanit.com/sql/sqlserver/1.1.php>
13. Бази даних. [Електронний ресурс]. – URL: <http://lib.mdpu.org.ua/e-book/vstup/L5.htm>
<http://lib.mdpu.org.ua/e-book/vstup/L5.htm>
14. За способом доступу до даних БД розподіляються на БД з локальним доступом та вилученим (мережним) доступом. [Електронний ресурс]. – URL:

https://studopedia.su/1_15809_za-sposobom-dostupu-do-danih-bd-rozpodilyayutsya-na-bd-z-lokalnim-dostupom-ta-viluchenim-merezhnim-dostupom.html

15. [GitHub](https://github.com/)[Електронний ресурс]. – URL: <https://github.com/>
16. Бьюли А. Изучаем SQL. – Санкт-Петербург: Символ-Плюс, 2015. – 312 с.
17. Грофф Дж.Р. Вайнберг П.Н. Оппель Е.Дж. SQL. Полное руководство. – Київ: Вильямс, 2015. – 960 с.
18. Осипов Д.Л. Технологии проектирования баз данных. – Москва: ДМК, 2019. – 498 с.
19. Шварц Б. Зайцев П. Ткаченко В. MySQL. Оптимизация производительности. – Санкт-Петербург: Питер, 2018. – 864 с.
20. Професія адміністратор[Електронний ресурс]. – URL: <https://workking.ru/uk/proforientatsiya/professiya/administrator-2.html>
21. Ахо А.В. Хопкрофт Дж.Э. Ульман Дж.Д. Структуры данных и алгоритмы. – Київ: Диалектика. 2019. – 400 с.
22. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – Санкт-Петербург: Питер, 2017. – 288 с.
23. Блок-схемы алгоритмов. ГОСТ. Примеры [Електронний ресурс]. – URL: <https://pro-prof.com/archives/1462>
24. Вігерс К. Розробка вимог до програмного забезпечення. – Санкт-Петербург: БХВ-Петербург, 2019. – 736 с.
25. Requirement Types вимог до системи [Електронний ресурс]. – URL: <https://www.quality-assurance-group.com/requirement-types/>
26. Специфікація вимог до програмного забезпечення [Електронний ресурс]. – URL: https://www.wikiwand.com/uk/%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F_%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3_%D0%B4%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7

[%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F](#)

27. Розробка технічного завдання [Електронний ресурс]. – URL: https://web.posibnyky.vntu.edu.ua/firen/2filynyuk_tehnichna_elektronika_kursove_proektuvannya/index_2.htm
28. Системи управління базами даних [Електронний ресурс]. – URL: <http://rodak.if.ua/komptech/lection4.htm>
29. Гайдаржи В. Ізварін І. Бази даних в інформаційних системах. – Київ: Університет "Україна", 2018. – 418 с.
30. Моделі даних в БД. ER-діаграми [Електронний ресурс]. – URL: <https://www.slideshare.net/pogromskaya/er-166809727>
31. Агуров П.О. С#. Сборник рецептов / Павел Агуров. – М.: БХВ-Петербург, 2012. - 432 с.
32. Бадд Т. Объектно-ориентированное программирование в действии: Пер. с англ./Тимоти Бадд. – СПб.: Питер, 1997.– 464 с.
33. Бондарев В.М. Программирование на С#. Учебное пособие. – Харьков: Компания СМІТ, 2005. – 284 с.
34. Вагнер Б. С# Эффективное программирование / Билл Вагнер. – М.: ЛОРИ, 2013. – 320 с.
35. Вендров А.С. Практикум по проектированию программного обеспечения ЭИС: Учебное пособие.– М.: Издательство «Финансы и статистика», 2004 – 239 с.
36. Вендров А.С. Проектирование программного обеспечения ЭИС. М.: Финансы и статистика, 2003 – 432 с.
37. Голдштейн С., Зурбалева Д., Флатов И. Оптимизация приложений на платформе .Net / Саша Голдштейн , Дима Зурбалева , Идо Флатов. – М.: ДМК Пресс, 2014. – 524 с.
38. Гуриков С. Р. Введение в программирование на языке Visual C# / С.Р. Гуриков. – М.: Дрофа, 2013. – 448 с.

39. Зиборов В.В. Visual C# 2010 на примерах / Виктор Зиборов. – М.: "БХВ–Петербург", 2011. – 432 с.
40. Зиборов В.В. Visual C# 2012 на примерах / В.В. Зиборов. – М.: БХВ–Петербург, 2013. – 480 с.
41. Інформація про класи та їх створення. – [Електронний ресурс]. – URL: https://www.w3schools.com/cs/cs_classes.asp.
42. Інформація про масиви. Клас List. – [Електронний ресурс]. – URL: http://mycsharp.ru/post/10/2013_04_19_massivy_v_si-sharp_klass_list.html.
43. Керівництво з програмування в Windows Forms. – [Електронний ресурс]. – URL: <https://metanit.com/sharp/windowsforms/1.1.php>.
44. Компонент Button C#. – [Електронний ресурс]. – URL: <https://ci-sharp.ru/Teaching/komponent-Button-C.html>.
45. Культин Н.В. Microsoft Visual C# в задачах и примерах / Н. Культин. – М.: БХВ–Петербург, 2015. – 320 с.
46. Культин Н.В. Основы программирования в Microsoft Visual C# 2010 (+ CD–ROM) / Никита Культин. – М.: БХВ–Петербург, 2011. – 368 с.
47. Лабор В.В. Електронна книга по C#. Створення додатків для Windows. – [Електронний ресурс]. – URL: http://sambir.tneu.edu.ua/my_downloads/learning/pi/4kurs/ZOOP/C%23%20-%20sozdaniye_prilozhenij_Windows.pdf.
48. Прата С. Язык программирования С. Лекции и упражнения. Учебник.- СПб.: ООО ДиаСофтЮП, 2002.- 896 с.
49. Таймер. Работа з таймером в C#. – [Електронний ресурс]. – URL: http://plssite.ru/csharp/csharp_timer_article.html.
50. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. – М.: Вильямс, 2015. – 486 с.
51. Троелсен Э. Язык программирования C# 2008 и платформа .NET 3.5 / Эндрю Троелсен. – М.: Вильямс, 2010. – 370 с.

52. Фримен А., Сандерсон С. ASP.NET MVC 3 Framework с примерами на C# для профессионалов / Адам Фримен, Стивен Сандерсон. – М.: Вильямс, 2011. – 672 с.
53. Albahari J., Albahari B. C# 8.0 Pocket Reference: Instant Help for C# 8.0 Programmers. – O'Reilly, 2019. – 248 p.
54. Albahari J., C# 10 in a Nutshell: The Definitive Reference. – O'Reilly, 2022. – 1058 p.
55. Griffiths I. Programming C# 10: Build Cloud, Web, and Desktop Applications. – O'Reilly, 2022. – 1058 p.
56. Цикли C# | for, While, Foreach та операції break, continue. – [Електронний ресурс]. – URL: <https://shwanoff.ru/cycle/>.
57. Microsoft Visual Studio. Офіційний сайт. – [Електронний ресурс]. – URL: <https://visualstudio.microsoft.com>
58. Baptista G., Abbruzzese F. Software Architecture with C# 12 and .NET 8: Build enterprise applications using microservices, DevOps, EF Core, and design patterns for Azure. – O'Reilly, 2024. – 756 p.
59. Troelsen A, Japikse P. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. – Eleventh, 2022. – 1705 p.
60. Документація Microsoft. – [Електронний ресурс]. – URL: <https://learn.microsoft.com/dotnet>
61. Sharp J. Microsoft Visual C# Step by Step. – CreateSpace Independent Publishing Platform, 2017. – 140 p.
62. Deitel P., Deitel H. Visual C#: How to Program. – Deitel & Associates? Inc., 2016. – 1056 p.
63. Макарук О.С., Шикула О.М. Розробка інформаційної системи адміністратора освітніх курсів за допомогою C# та MySQL в середовищі Visual Studio // V всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУІКТ, 2025. – С. 224-226.

ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

Form 1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;
using System.Data.SqlClient;

namespace Адміністрування
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private void Form1_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.LoginTbl". При необходимости она может быть
                перемещена или удалена.
                this.loginTblTableAdapter.Fill(this.administratorDataSet.LoginTbl);
                usernameTextBox.Select();
            }

            private void button2_Click(object sender, EventArgs e)
            {
                this.Hide();
                Form2 form1 = new Form2();

                form1.Show();
            }

            private void button4_Click(object sender, EventArgs e)
            {
                this.Hide();
                Form20 form25 = new Form20();

                form25.Show();
            }

            private void button1_Click(object sender, EventArgs e)
            {
                Form4 af = new Form4();
                af.Owner = this;
                af.Show();
            }

            private void panel1_Paint(object sender, PaintEventArgs e)
            {
            }

            private void cancelButton_Click(object sender, EventArgs e)
            {
                Application.Exit();
            }

            private void openRegisterFromLinkLabel_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
            {
                RegisterForm register = new RegisterForm();
                register.ShowDialog();
            }

            private void showPasswordCheckBox_CheckedChanged(object sender, EventArgs e)
            {
            }
        }
    }
}
```

```

        if(showPasswordCheckBox.Checked==true)
        {
            passwordTextBox.UseSystemPasswordChar = false;
        }
        else
        {
            passwordTextBox.UseSystemPasswordChar = true;
        }
    }

    private void loginButton_Click(object sender, EventArgs e)
    {
        if (!string.IsNullOrEmpty(usernameTextBox.Text)&&
            !string.IsNullOrEmpty(passwordTextBox.Text))
        {
            string mySQL = string.Empty;

            mySQL += "SELECT * FROM LoginTbl ";
            mySQL += "WHERE Username = '" + usernameTextBox.Text + "' ";
            mySQL += "AND Password = '" + passwordTextBox.Text + "'";

            DataTable userData = SQLServerConnection.executeSQL(mySQL);
            if (userData.Rows.Count>0)
            {
                usernameTextBox.Clear();
                passwordTextBox.Clear();
                showPasswordCheckBox.Checked = false;

                this.Hide();

                Form2 fo2 = new Form2();
                fo2.ShowDialog();
                fo2 = null;

                //this.Show();
                this.usernameTextBox.Select();

            }
            else
            {
                {
                    MessageBox.Show("Ім'я користувача або пароль невірні. Спробуйте ще раз.",
                        "Помилка:", MessageBoxButtons.OK, MessageBoxIcon.Stop);
                    usernameTextBox.Focus();
                    usernameTextBox.SelectAll();
                }
            }
        }
        else
        {
            {
                MessageBox.Show("Введіть ім'я користувача та пароль", "Помилка:",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                usernameTextBox.Select();
            }
        }
    }

    private void loginTblBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.loginTblBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.administratorDataSet);

    }

    private void label1_Click(object sender, EventArgs e)
    {
    }
}
}
}

```

Form 2

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Адміністрування
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.Courses". При необходимости она может быть
            // перемещена или удалена.
            this.coursesTableAdapter.Fill(this.administratorDataSet.Courses);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form3 form2 = new Form3();

            form2.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form5 form3 = new Form5();

            form3.Show();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form6 form4 = new Form6();

            form4.Show();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form8 form8 = new Form8();

            form8.Show();
        }

        private void button10_Click(object sender, EventArgs e)
        {
        }

        private void button11_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form14 form14 = new Form14();

            form14.Show();
        }

        private void button12_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form15 form15 = new Form15();

            form15.Show();
        }
    }
}

```

```

private void button15_Click(object sender, EventArgs e)
{
    this.Hide();
    Form18 form18 = new Form18();

    form18.Show();
}

private void button16_Click(object sender, EventArgs e)
{
}

private void button17_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 form29 = new Form1();

    form29.Show();
}

private void button18_Click(object sender, EventArgs e)
{
    Close();
}

private void button21_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form26 = new Form2();

    form26.Show();
}

private void button22_Click(object sender, EventArgs e)
{
    this.Hide();
    Form20 form25 = new Form20();

    form25.Show();
}

private void button20_Click(object sender, EventArgs e)
{
    this.Hide();
    Form23 form27 = new Form23();

    form27.Show();
}

private void cancelButton_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void coursesBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    this.Validate();
    this.coursesBindingSource.EndEdit();
    this.tableAdapterManager.UpdateAll(this.administratorDataSet);
}

private void panell1_Paint(object sender, PaintEventArgs e)
{
}

private void button19_Click(object sender, EventArgs e)
{
    this.Hide();
    Form7 for1 = new Form7();

    for1.Show();
}
}
}

```

Form 3

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;

namespace Адміністрування
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();

            private void Form3_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.Client". При необходимости она может быть
                перемещена или удалена.
                this.clientTableAdapter.Fill(this.administratorDataSet.Client);
                loadUserData();

                // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.N11AccountingForUkraine". При необходимости
                она может быть перемещена или удалена.
                this.n11AccountingForUkraineTableAdapter.Fill(this.administratorDataSet.N11AccountingForUkraine);
                // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.N3AccountingForUkraine". При необходимости
                она может быть перемещена или удалена.
                this.n3AccountingForUkraineTableAdapter.Fill(this.administratorDataSet.N3AccountingForUkraine);
                // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.N2AccountingForUkraine". При необходимости
                она может быть перемещена или удалена.
                this.n2AccountingForUkraineTableAdapter.Fill(this.administratorDataSet.N2AccountingForUkraine);

            }

            private void loadUserData()
            {
                DataTable userData = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM
                Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Чекає навчання' and NameCourses = 'Бугалтерія'");
                dataGridView1.DataSource = userData;
                dataGridView1.Columns[0].HeaderText = "Дата";
                dataGridView1.Columns[1].HeaderText = "ФІО";
                dataGridView1.Columns[2].HeaderText = "Email";
                dataGridView1.Columns[3].HeaderText = "Телефон";
                dataGridView1.Columns[4].HeaderText = "Бажаний час навчання";
                dataGridView1.Columns[5].HeaderText = "Додаткова інформація";
                dataGridView1.Columns[6].HeaderText = "Актуальність";
                dataGridView1.Columns[7].HeaderText = "Ціна";
                dataGridView1.Columns[0].Width = 80;
                dataGridView1.Columns[1].Width = 120;
                dataGridView1.Columns[2].Width = 145;
                dataGridView1.Columns[3].Width = 80;
                dataGridView1.Columns[4].Width = 120;
                dataGridView1.Columns[5].Width = 140;
                dataGridView1.Columns[6].Width = 140;
                dataGridView1.Columns[7].Width = 85;

                DataTable userData1 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price
                FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Навчається' and NameCourses = 'Бугалтерія'");
                dataGridView2.DataSource = userData1;
                dataGridView2.Columns[0].HeaderText = "Дата";
                dataGridView2.Columns[1].HeaderText = "ФІО";
                dataGridView2.Columns[2].HeaderText = "Email";
                dataGridView2.Columns[3].HeaderText = "Телефон";
                dataGridView2.Columns[4].HeaderText = "Бажаний час навчання";
                dataGridView2.Columns[5].HeaderText = "Додаткова інформація";
```

```

dataGridView2.Columns[6].HeaderText = "Актуальність";
dataGridView2.Columns[7].HeaderText = "Ціна";
dataGridView2.Columns[0].Width = 80;
dataGridView2.Columns[1].Width = 120;
dataGridView2.Columns[2].Width = 145;
dataGridView2.Columns[3].Width = 80;
dataGridView2.Columns[4].Width = 120;
dataGridView2.Columns[5].Width = 140;
dataGridView2.Columns[6].Width = 140;
dataGridView2.Columns[7].Width = 85;

```

```

DataTable userData2 = SQLServerConnection.executeSQL("SELECT Date, Name,Email,Phone,TrainingTime,Information,Topicality,Price
FROM Client join Courses on Client.Courses=Courses.IdCourses where NameCourses = 'Бухгалтерія'");

```

```

dataGridView3.DataSource = userData2;
dataGridView3.Columns[0].HeaderText = "Дата";
dataGridView3.Columns[1].HeaderText = "ФІО";
dataGridView3.Columns[2].HeaderText = "Email";
dataGridView3.Columns[3].HeaderText = "Телефон";
dataGridView3.Columns[4].HeaderText = "Бажаний час навчання";
dataGridView3.Columns[5].HeaderText = "Додаткова інформація";
dataGridView3.Columns[6].HeaderText = "Актуальність";
dataGridView3.Columns[7].HeaderText = "Ціна";
dataGridView3.Columns[0].Width = 80;
dataGridView3.Columns[1].Width = 120;
dataGridView3.Columns[2].Width = 145;
dataGridView3.Columns[3].Width = 80;
dataGridView3.Columns[4].Width = 120;
dataGridView3.Columns[5].Width = 140;
dataGridView3.Columns[6].Width = 140;
dataGridView2.Columns[7].Width = 85;

```

```

}

```

```

private void button1_Click(object sender, EventArgs e)
{
    //n11AccountingForUkraineTableAdapter.Update(administratorDataSet.N11AccountingForUkraine);
    Form21 af = new Form21();
    af.Owner = this;
    af.Show();
}

```

```

private void button3_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button9_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

```

```

private void button4_Click(object sender, EventArgs e)
{
    Application.Exit();
}

```

```

private void button7_Click(object sender, EventArgs e)
{
    Close();
}

```

```

private void button12_Click(object sender, EventArgs e)
{
    Close();
}

```

```

    }

    private void button11_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }
}

```

Form 4

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Адміністрування
{
    public partial class Form4 : Form
    {
        public Form4()
        {
            InitializeComponent();
        }

        private void label8_Click(object sender, EventArgs e)
        {
        }

        private void AddBtn_Click(object sender, EventArgs e)
        {
            Form21 main = this.Owner as Form21;
            if (main != null)
            {
                DataRow nRow = main.administratorDataSet.Tables[0].NewRow();
                int rc = main.clientDataGridview.RowCount + 5;
                nRow[0] = rc;
                nRow[1] = tbDate.Text;
                nRow[2] = tbName.Text;
                nRow[3] = tbEmail.Text;
                nRow[4] = tbPhone.Text;
                nRow[5] = tbTrainingTime.Text;
                nRow[6] = tbInformation.Text;
                nRow[7] = tbCourses.Text;
                nRow[8] = tbTopicality.Text;
                nRow[9] = tbPrice.Text;
                main.administratorDataSet.Tables[0].Rows.Add(nRow);
                main.clientTableAdapter.Update(main.administratorDataSet.Client);
                main.administratorDataSet.Tables[0].AcceptChanges();
                main.clientDataGridview.Refresh();
                tbDate.Text = "";
                tbName.Text = "";
            }
        }
    }
}

```

```

        tbEmail.Text = "";
        tbPhone.Text = "";
        tbTrainingTime.Text = "";
        tbInformation.Text = "";
        tbCourses.Text = "";
        tbTopicality.Text = "";
        tbPrice.Text = "";
    }

}

private void CloseBtn_Click(object sender, EventArgs e)
{
    Close();
}

private void Form4_Load(object sender, EventArgs e)
{
}

private void button1_Click(object sender, EventArgs e)
{
}
}
}

```

Form 5

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;

namespace Адміністрування
{
    public partial class Form5 : Form
    {
        public Form5()
        {
            InitializeComponent();
        }

        private void Form5_Load(object sender, EventArgs e)
        {
            loadUserData();
            // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.Client". При необходимости она может быть
            перемещена или удалена.
            this.clientTableAdapter.Fill(this.administratorDataSet.Client);
        }

        private void loadUserData()
        {
            DataTable userData = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM
Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Чекає навчання' and NameCourses = 'Графічний дизайнер'");
            dataGridView1.DataSource = userData;
            dataGridView1.Columns[0].HeaderText = "Дата";
            dataGridView1.Columns[1].HeaderText = "ФІО";
            dataGridView1.Columns[2].HeaderText = "Email";
            dataGridView1.Columns[3].HeaderText = "Телефон";
            dataGridView1.Columns[4].HeaderText = "Бажаний час навчання";
            dataGridView1.Columns[5].HeaderText = "Додаткова інформація";
            dataGridView1.Columns[6].HeaderText = "Актуальність";
            dataGridView1.Columns[7].HeaderText = "Ціна";
            dataGridView1.Columns[0].Width = 80;
            dataGridView1.Columns[1].Width = 120;
            dataGridView1.Columns[2].Width = 145;
            dataGridView1.Columns[3].Width = 80;
            dataGridView1.Columns[4].Width = 120;

```

```

dataGridView1.Columns[5].Width = 140;
dataGridView1.Columns[6].Width = 140;
dataGridView1.Columns[7].Width = 85;

```

```

DataTable userData1 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price
FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Навчається' and NameCourses = 'Графічний дизайнер'");
dataGridView2.DataSource = userData1;
dataGridView2.Columns[0].HeaderText = "Дата";
dataGridView2.Columns[1].HeaderText = "ФІО";
dataGridView2.Columns[2].HeaderText = "Email";
dataGridView2.Columns[3].HeaderText = "Телефон";
dataGridView2.Columns[4].HeaderText = "Бажаний час навчання";
dataGridView2.Columns[5].HeaderText = "Додаткова інформація";
dataGridView2.Columns[6].HeaderText = "Актуальність";
dataGridView2.Columns[7].HeaderText = "Ціна";
dataGridView2.Columns[0].Width = 80;
dataGridView2.Columns[1].Width = 120;
dataGridView2.Columns[2].Width = 145;
dataGridView2.Columns[3].Width = 80;
dataGridView2.Columns[4].Width = 120;
dataGridView2.Columns[5].Width = 140;
dataGridView2.Columns[6].Width = 140;
dataGridView2.Columns[7].Width = 85;

```

```

DataTable userData2 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price
FROM Client join Courses on Client.Courses=Courses.IdCourses where NameCourses = 'Графічний дизайнер'");
dataGridView3.DataSource = userData2;
dataGridView3.Columns[0].HeaderText = "Дата";
dataGridView3.Columns[1].HeaderText = "ФІО";
dataGridView3.Columns[2].HeaderText = "Email";
dataGridView3.Columns[3].HeaderText = "Телефон";
dataGridView3.Columns[4].HeaderText = "Бажаний час навчання";
dataGridView3.Columns[5].HeaderText = "Додаткова інформація";
dataGridView3.Columns[6].HeaderText = "Актуальність";
dataGridView3.Columns[7].HeaderText = "Ціна";
dataGridView3.Columns[0].Width = 80;
dataGridView3.Columns[1].Width = 120;
dataGridView3.Columns[2].Width = 145;
dataGridView3.Columns[3].Width = 80;
dataGridView3.Columns[4].Width = 120;
dataGridView3.Columns[5].Width = 140;
dataGridView3.Columns[6].Width = 140;
dataGridView2.Columns[7].Width = 85;

```

```

}

```

```

private void button1_Click_1(object sender, EventArgs e)
{
}

```

```

private void button5_Click_1(object sender, EventArgs e)
{
}

```

```

private void button9_Click_1(object sender, EventArgs e)
{
}

```

```

private void button2_Click_1(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button6_Click_1(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button10_Click_1(object sender, EventArgs e)

```

```

{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

private void button11_Click_1(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button7_Click_1(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button3_Click_1(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button12_Click(object sender, EventArgs e)
{
    Close();
}

private void button8_Click_1(object sender, EventArgs e)
{
    Close();
}

private void button4_Click_1(object sender, EventArgs e)
{
    Close();
}

private void button15_Click(object sender, EventArgs e)
{
    Form21 af = new Form21();
    af.Owner = this;
    af.Show();
}

private void button17_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

private void button16_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button14_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form19 = new Form2();

    form19.Show();
}

private void button13_Click(object sender, EventArgs e)
{
    Close();
}

```

```

    }
}
}

```

Form 6

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;

namespace Адміністрування
{
    public partial class Form6 : Form
    {
        public Form6()
        {
            InitializeComponent();

            private void Form6_Load(object sender, EventArgs e)
            {
                loadUserData();
            }

            private void loadUserData()
            {
                DataTable userData = SQLServerConnection.executeSQL("SELECT Date, Name,Email,Phone,TrainingTime,Information,Topicality,Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Чекає навчання' and NameCourses = 'BAS Бухгалтерія'");
                dataGridView1.DataSource = userData;
                dataGridView1.Columns[0].HeaderText = "Дата";
                dataGridView1.Columns[1].HeaderText = "ФІО";
                dataGridView1.Columns[2].HeaderText = "Email";
                dataGridView1.Columns[3].HeaderText = "Телефон";
                dataGridView1.Columns[4].HeaderText = "Бажаний час навчання";
                dataGridView1.Columns[5].HeaderText = "Додаткова інформація";
                dataGridView1.Columns[6].HeaderText = "Актуальність";
                dataGridView1.Columns[7].HeaderText = "Ціна";
                dataGridView1.Columns[0].Width = 80;
                dataGridView1.Columns[1].Width = 120;
                dataGridView1.Columns[2].Width = 145;
                dataGridView1.Columns[3].Width = 80;
                dataGridView1.Columns[4].Width = 120;
                dataGridView1.Columns[5].Width = 140;
                dataGridView1.Columns[6].Width = 140;
                dataGridView1.Columns[7].Width = 85;

                DataTable userData1 = SQLServerConnection.executeSQL("SELECT Date, Name,Email,Phone,TrainingTime,Information,Topicality,Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Навчається' and NameCourses = 'BAS Бухгалтерія'");
                dataGridView2.DataSource = userData1;
                dataGridView2.Columns[0].HeaderText = "Дата";
                dataGridView2.Columns[1].HeaderText = "ФІО";
                dataGridView2.Columns[2].HeaderText = "Email";
                dataGridView2.Columns[3].HeaderText = "Телефон";
                dataGridView2.Columns[4].HeaderText = "Бажаний час навчання";
                dataGridView2.Columns[5].HeaderText = "Додаткова інформація";
                dataGridView2.Columns[6].HeaderText = "Актуальність";
                dataGridView2.Columns[7].HeaderText = "Ціна";
                dataGridView2.Columns[0].Width = 80;
                dataGridView2.Columns[1].Width = 120;
                dataGridView2.Columns[2].Width = 145;
                dataGridView2.Columns[3].Width = 80;
                dataGridView2.Columns[4].Width = 120;
                dataGridView2.Columns[5].Width = 140;
                dataGridView2.Columns[6].Width = 140;
                dataGridView2.Columns[7].Width = 85;

                DataTable userData2 = SQLServerConnection.executeSQL("SELECT Date, Name,Email,Phone,TrainingTime,Information,Topicality,Price FROM Client join Courses on Client.Courses=Courses.IdCourses where NameCourses = 'BAS Бухгалтерія'");

```

```

dataGridView3.DataSource = userData2;
dataGridView3.Columns[0].HeaderText = "Дата";
dataGridView3.Columns[1].HeaderText = "ФІО";
dataGridView3.Columns[2].HeaderText = "Email";
dataGridView3.Columns[3].HeaderText = "Телефон";
dataGridView3.Columns[4].HeaderText = "Бажаний час навчання";
dataGridView3.Columns[5].HeaderText = "Додаткова інформація";
dataGridView3.Columns[6].HeaderText = "Актуальність";
dataGridView3.Columns[7].HeaderText = "Ціна";
dataGridView3.Columns[0].Width = 80;
dataGridView3.Columns[1].Width = 120;
dataGridView3.Columns[2].Width = 145;
dataGridView3.Columns[3].Width = 80;
dataGridView3.Columns[4].Width = 120;
dataGridView3.Columns[5].Width = 140;
dataGridView3.Columns[6].Width = 140;
dataGridView2.Columns[7].Width = 85;

```

```

}

```

```

private void button9_Click(object sender, EventArgs e)
{
    clientTableAdapter.Update(administratorDataSet.Client);
}

```

```

private void button10_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button6_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button3_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

```

```

private void button7_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

```

```

private void button11_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

```

```

private void button12_Click(object sender, EventArgs e)
{
    Close();
}

```

```

private void button8_Click(object sender, EventArgs e)
{
    Close();
}

```

```

private void button4_Click(object sender, EventArgs e)

```

```

    {
        Close();
    }

    private void button13_Click(object sender, EventArgs e)
    {
        Form21 af = new Form21();
        af.Owner = this;
        af.Show();
    }

    private void button15_Click(object sender, EventArgs e)
    {
        Form4 af = new Form4();
        af.Owner = this;
        af.Show();
    }

    private void button14_Click(object sender, EventArgs e)
    {
        SearchForm sf = new SearchForm();
        sf.Owner = this;
        sf.Show();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Close();
    }
}

```

Form 7

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Адміністрування
{
    public partial class Form7 : Form
    {
        public Form7()
        {
            InitializeComponent();
        }

        private void teacherBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.teacherBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.administratorDataSet);
        }

        private void Form7_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.Teacher". При необходимости она может быть
            перемещена или удалена.
            this.teacherTableAdapter.Fill(this.administratorDataSet.Teacher);
        }

        private void button5_Click(object sender, EventArgs e)

```

```

    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
}

```

Form 8

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;

namespace Адміністрування
{
    public partial class Form8 : Form
    {
        public Form8()
        {
            InitializeComponent();
        }

        private void Form8_Load(object sender, EventArgs e)
        {
            loadUserData();
        }

        private void loadUserData()
        {
            DataTable userData = SQLServerConnection.executeSQL("SELECT Date, Name,Email,Phone,TrainingTime,Information,Topicality,Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Чекає навчання' and NameCourses = 'Кібербезпека'");
            dataGridView1.DataSource = userData;
            dataGridView1.Columns[0].HeaderText = "Дата";
            dataGridView1.Columns[1].HeaderText = "ФІО";
            dataGridView1.Columns[2].HeaderText = "Email";
            dataGridView1.Columns[3].HeaderText = "Телефон";
            dataGridView1.Columns[4].HeaderText = "Бажаний час навчання";
            dataGridView1.Columns[5].HeaderText = "Додаткова інформація";
            dataGridView1.Columns[6].HeaderText = "Актуальність";
            dataGridView1.Columns[7].HeaderText = "Ціна";
            dataGridView1.Columns[0].Width = 80;
            dataGridView1.Columns[1].Width = 120;
            dataGridView1.Columns[2].Width = 145;
            dataGridView1.Columns[3].Width = 80;
            dataGridView1.Columns[4].Width = 120;
            dataGridView1.Columns[5].Width = 140;
            dataGridView1.Columns[6].Width = 140;
            dataGridView1.Columns[7].Width = 85;

            DataTable userData1 = SQLServerConnection.executeSQL("SELECT Date, Name,Email,Phone,TrainingTime,Information,Topicality,Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Навчається' and NameCourses = 'Кібербезпека'");
            dataGridView2.DataSource = userData1;
            dataGridView2.Columns[0].HeaderText = "Дата";
            dataGridView2.Columns[1].HeaderText = "ФІО";
            dataGridView2.Columns[2].HeaderText = "Email";
            dataGridView2.Columns[3].HeaderText = "Телефон";
            dataGridView2.Columns[4].HeaderText = "Бажаний час навчання";
            dataGridView2.Columns[5].HeaderText = "Додаткова інформація";
            dataGridView2.Columns[6].HeaderText = "Актуальність";
            dataGridView2.Columns[7].HeaderText = "Ціна";
            dataGridView2.Columns[0].Width = 80;
            dataGridView2.Columns[1].Width = 120;
            dataGridView2.Columns[2].Width = 145;
            dataGridView2.Columns[3].Width = 80;
            dataGridView2.Columns[4].Width = 120;
            dataGridView2.Columns[5].Width = 140;

```

```
dataGridView2.Columns[6].Width = 140;  
dataGridView2.Columns[7].Width = 85;
```

```
DataTable userData2 = SQLServerConnection.executeSQL("SELECT Date, Name,Email,Phone,TrainingTime,Information,Topicality,Price  
FROM Client join Courses on Client.Courses=Courses.IdCourses where NameCourses = 'Кібербезпека'");
```

```
dataGridView3.DataSource = userData2;  
dataGridView3.Columns[0].HeaderText = "Дата";  
dataGridView3.Columns[1].HeaderText = "ФІО";  
dataGridView3.Columns[2].HeaderText = "Email";  
dataGridView3.Columns[3].HeaderText = "Телефон";  
dataGridView3.Columns[4].HeaderText = "Бажаний час навчання";  
dataGridView3.Columns[5].HeaderText = "Додаткова інформація";  
dataGridView3.Columns[6].HeaderText = "Актуальність";  
dataGridView3.Columns[7].HeaderText = "Ціна";  
dataGridView3.Columns[0].Width = 80;  
dataGridView3.Columns[1].Width = 120;  
dataGridView3.Columns[2].Width = 145;  
dataGridView3.Columns[3].Width = 80;  
dataGridView3.Columns[4].Width = 120;  
dataGridView3.Columns[5].Width = 140;  
dataGridView3.Columns[6].Width = 140;  
dataGridView2.Columns[7].Width = 85;
```

```
}
```

```
private void button7_Click(object sender, EventArgs e)  
{  
    clientTableAdapter.Update(administratorDataSet.Client);  
}
```

```
private void button8_Click(object sender, EventArgs e)  
{  
    Form4 af = new Form4();  
    af.Owner = this;  
    af.Show();  
}
```

```
private void button4_Click(object sender, EventArgs e)  
{  
    Form4 af = new Form4();  
    af.Owner = this;  
    af.Show();  
}
```

```
private void button1_Click(object sender, EventArgs e)  
{  
    Form4 af = new Form4();  
    af.Owner = this;  
    af.Show();  
}
```

```
private void button2_Click(object sender, EventArgs e)  
{  
    SearchForm sf = new SearchForm();  
    sf.Owner = this;  
    sf.Show();  
}
```

```
private void button5_Click(object sender, EventArgs e)  
{  
    SearchForm sf = new SearchForm();  
    sf.Owner = this;  
    sf.Show();  
}
```

```
private void button9_Click(object sender, EventArgs e)  
{  
    SearchForm sf = new SearchForm();  
    sf.Owner = this;  
    sf.Show();  
}
```

```
private void button10_Click(object sender, EventArgs e)  
{  
    Close();  
}
```

```
private void button6_Click(object sender, EventArgs e)
```

```

    {
        Close();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void button13_Click(object sender, EventArgs e)
    {
        Form21 af = new Form21();
        af.Owner = this;
        af.Show();
    }

    private void button15_Click(object sender, EventArgs e)
    {
        Form4 af = new Form4();
        af.Owner = this;
        af.Show();
    }

    private void button14_Click(object sender, EventArgs e)
    {
        SearchForm sf = new SearchForm();
        sf.Owner = this;
        sf.Show();
    }

    private void button12_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }

    private void button11_Click(object sender, EventArgs e)
    {
        Close();
    }
}

```

Form 14

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;

namespace Адміністрування
{
    public partial class Form14 : Form
    {
        public Form14()
        {
            InitializeComponent();
        }

        private void Form14_Load(object sender, EventArgs e)
        {
            loadUserData();
        }

        private void loadUserData()
        {

```

```

        DataTable userData = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Чекає навчання' and NameCourses = 'Java'");
        dataGridView1.DataSource = userData;
        dataGridView1.Columns[0].HeaderText = "Дата";
        dataGridView1.Columns[1].HeaderText = "ФІО";
        dataGridView1.Columns[2].HeaderText = "Email";
        dataGridView1.Columns[3].HeaderText = "Телефон";
        dataGridView1.Columns[4].HeaderText = "Бажаний час навчання";
        dataGridView1.Columns[5].HeaderText = "Додаткова інформація";
        dataGridView1.Columns[6].HeaderText = "Актуальність";
        dataGridView1.Columns[7].HeaderText = "Ціна";
        dataGridView1.Columns[0].Width = 80;
        dataGridView1.Columns[1].Width = 120;
        dataGridView1.Columns[2].Width = 145;
        dataGridView1.Columns[3].Width = 80;
        dataGridView1.Columns[4].Width = 120;
        dataGridView1.Columns[5].Width = 140;
        dataGridView1.Columns[6].Width = 140;
        dataGridView1.Columns[7].Width = 85;

```

```

        DataTable userData1 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Навчається' and NameCourses = 'Java'");
        dataGridView2.DataSource = userData1;
        dataGridView2.Columns[0].HeaderText = "Дата";
        dataGridView2.Columns[1].HeaderText = "ФІО";
        dataGridView2.Columns[2].HeaderText = "Email";
        dataGridView2.Columns[3].HeaderText = "Телефон";
        dataGridView2.Columns[4].HeaderText = "Бажаний час навчання";
        dataGridView2.Columns[5].HeaderText = "Додаткова інформація";
        dataGridView2.Columns[6].HeaderText = "Актуальність";
        dataGridView2.Columns[7].HeaderText = "Ціна";
        dataGridView2.Columns[0].Width = 80;
        dataGridView2.Columns[1].Width = 120;
        dataGridView2.Columns[2].Width = 145;
        dataGridView2.Columns[3].Width = 80;
        dataGridView2.Columns[4].Width = 120;
        dataGridView2.Columns[5].Width = 140;
        dataGridView2.Columns[6].Width = 140;
        dataGridView2.Columns[7].Width = 85;

```

```

        DataTable userData2 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM Client join Courses on Client.Courses=Courses.IdCourses where NameCourses = 'Java'");
        dataGridView3.DataSource = userData2;
        dataGridView3.Columns[0].HeaderText = "Дата";
        dataGridView3.Columns[1].HeaderText = "ФІО";
        dataGridView3.Columns[2].HeaderText = "Email";
        dataGridView3.Columns[3].HeaderText = "Телефон";
        dataGridView3.Columns[4].HeaderText = "Бажаний час навчання";
        dataGridView3.Columns[5].HeaderText = "Додаткова інформація";
        dataGridView3.Columns[6].HeaderText = "Актуальність";
        dataGridView3.Columns[7].HeaderText = "Ціна";
        dataGridView3.Columns[0].Width = 80;
        dataGridView3.Columns[1].Width = 120;
        dataGridView3.Columns[2].Width = 145;
        dataGridView3.Columns[3].Width = 80;
        dataGridView3.Columns[4].Width = 120;
        dataGridView3.Columns[5].Width = 140;
        dataGridView3.Columns[6].Width = 140;
        dataGridView2.Columns[7].Width = 85;

```

```

    }
    private void button2_Click(object sender, EventArgs e)
    {
        Form4 af = new Form4();
        af.Owner = this;
        af.Show();
    }

    private void button7_Click(object sender, EventArgs e)
    {
        Form4 af = new Form4();
        af.Owner = this;
        af.Show();
    }

    private void button10_Click(object sender, EventArgs e)
    {

```

```

        SearchForm sf = new SearchForm();
        sf.Owner = this;
        sf.Show();
    }

    private void button11_Click(object sender, EventArgs e)
    {
        Form4 af = new Form4();
        af.Owner = this;
        af.Show();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        SearchForm sf = new SearchForm();
        sf.Owner = this;
        sf.Show();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        SearchForm sf = new SearchForm();
        sf.Owner = this;
        sf.Show();
    }

    private void button9_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void button15_Click(object sender, EventArgs e)
    {
        Form21 af = new Form21();
        af.Owner = this;
        af.Show();
    }

    private void button17_Click(object sender, EventArgs e)
    {
        Form4 af = new Form4();
        af.Owner = this;
        af.Show();
    }

    private void button16_Click(object sender, EventArgs e)
    {
        SearchForm sf = new SearchForm();
        sf.Owner = this;
        sf.Show();
    }

    private void button14_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }

    private void button13_Click(object sender, EventArgs e)
    {
        Close();
    }
}

```

Form 15

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;

namespace Адміністрування
{
    public partial class Form15 : Form
    {
        public Form15()
        {
            InitializeComponent();
        }

        private void Form15_Load(object sender, EventArgs e)
        {
            loadUserData();
        }

        private void loadUserData()
        {
            DataTable userData = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Чекає навчання' and NameCourses = 'М Е ДОС'");
            dataGridView1.DataSource = userData;
            dataGridView1.Columns[0].HeaderText = "Дата";
            dataGridView1.Columns[1].HeaderText = "ФІО";
            dataGridView1.Columns[2].HeaderText = "Email";
            dataGridView1.Columns[3].HeaderText = "Телефон";
            dataGridView1.Columns[4].HeaderText = "Бажаний час навчання";
            dataGridView1.Columns[5].HeaderText = "Додаткова інформація";
            dataGridView1.Columns[6].HeaderText = "Актуальність";
            dataGridView1.Columns[7].HeaderText = "Ціна";
            dataGridView1.Columns[0].Width = 80;
            dataGridView1.Columns[1].Width = 120;
            dataGridView1.Columns[2].Width = 145;
            dataGridView1.Columns[3].Width = 80;
            dataGridView1.Columns[4].Width = 120;
            dataGridView1.Columns[5].Width = 140;
            dataGridView1.Columns[6].Width = 140;
            dataGridView1.Columns[7].Width = 85;

            DataTable userData1 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Навчається' and NameCourses = 'М Е ДОС'");
            dataGridView2.DataSource = userData1;
            dataGridView2.Columns[0].HeaderText = "Дата";
            dataGridView2.Columns[1].HeaderText = "ФІО";
            dataGridView2.Columns[2].HeaderText = "Email";
            dataGridView2.Columns[3].HeaderText = "Телефон";
            dataGridView2.Columns[4].HeaderText = "Бажаний час навчання";
            dataGridView2.Columns[5].HeaderText = "Додаткова інформація";
            dataGridView2.Columns[6].HeaderText = "Актуальність";
            dataGridView2.Columns[7].HeaderText = "Ціна";
            dataGridView2.Columns[0].Width = 80;
            dataGridView2.Columns[1].Width = 120;
            dataGridView2.Columns[2].Width = 145;
            dataGridView2.Columns[3].Width = 80;
            dataGridView2.Columns[4].Width = 120;
            dataGridView2.Columns[5].Width = 140;
            dataGridView2.Columns[6].Width = 140;
            dataGridView2.Columns[7].Width = 85;

            DataTable userData2 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM Client join Courses on Client.Courses=Courses.IdCourses where NameCourses = 'М Е ДОС'");
            dataGridView3.DataSource = userData2;
            dataGridView3.Columns[0].HeaderText = "Дата";
            dataGridView3.Columns[1].HeaderText = "ФІО";
            dataGridView3.Columns[2].HeaderText = "Email";
```

```

dataGridView3.Columns[3].HeaderText = "Телефон";
dataGridView3.Columns[4].HeaderText = "Бажаний час навчання";
dataGridView3.Columns[5].HeaderText = "Додаткова інформація";
dataGridView3.Columns[6].HeaderText = "Актуальність";
dataGridView3.Columns[7].HeaderText = "Ціна";
dataGridView3.Columns[0].Width = 80;
dataGridView3.Columns[1].Width = 120;
dataGridView3.Columns[2].Width = 145;
dataGridView3.Columns[3].Width = 80;
dataGridView3.Columns[4].Width = 120;
dataGridView3.Columns[5].Width = 140;
dataGridView3.Columns[6].Width = 140;
dataGridView2.Columns[7].Width = 85;

}

private void button11_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

private void button7_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

private void button2_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button6_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button10_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button9_Click(object sender, EventArgs e)
{
    Close();
}

private void button5_Click(object sender, EventArgs e)
{
    Close();
}

private void button4_Click(object sender, EventArgs e)
{
    Close();
}

private void button15_Click(object sender, EventArgs e)
{
    Form21 af = new Form21();
    af.Owner = this;
    af.Show();
}

```

```

private void button17_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

private void button16_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button14_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form19 = new Form2();

    form19.Show();
}

private void button13_Click(object sender, EventArgs e)
{
    Close();
}
}
}

```

Form 18

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;

```

```

namespace Адміністрування

```

```

{
    public partial class Form18 : Form
    {
        public Form18()
        {
            InitializeComponent();
        }
    }
}

```

```

private void Form18_Load(object sender, EventArgs e)
{
    loadUserData();
}

```

```

private void loadUserData()
{

```

```

    DataTable userData = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Чекає навчання' and NameCourses = 'Розробка web-сайтів'");
    dataGridView1.DataSource = userData;
    dataGridView1.Columns[0].HeaderText = "Дата";
    dataGridView1.Columns[1].HeaderText = "ФІО";
    dataGridView1.Columns[2].HeaderText = "Email";
    dataGridView1.Columns[3].HeaderText = "Телефон";
    dataGridView1.Columns[4].HeaderText = "Бажаний час навчання";
    dataGridView1.Columns[5].HeaderText = "Додаткова інформація";
    dataGridView1.Columns[6].HeaderText = "Актуальність";
    dataGridView1.Columns[7].HeaderText = "Ціна";
    dataGridView1.Columns[0].Width = 80;
    dataGridView1.Columns[1].Width = 120;
    dataGridView1.Columns[2].Width = 145;
}

```

```

dataGridView1.Columns[3].Width = 80;
dataGridView1.Columns[4].Width = 120;
dataGridView1.Columns[5].Width = 140;
dataGridView1.Columns[6].Width = 140;
dataGridView1.Columns[7].Width = 85;

```

```

DataTable userData1 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price
FROM Client join Courses on Client.Courses=Courses.IdCourses where Topicality='Навчається' and NameCourses = 'Розробка web-сайтів'");

```

```

dataGridView2.DataSource = userData1;
dataGridView2.Columns[0].HeaderText = "Дата";
dataGridView2.Columns[1].HeaderText = "ФІО";
dataGridView2.Columns[2].HeaderText = "Email";
dataGridView2.Columns[3].HeaderText = "Телефон";
dataGridView2.Columns[4].HeaderText = "Бажаний час навчання";
dataGridView2.Columns[5].HeaderText = "Додаткова інформація";
dataGridView2.Columns[6].HeaderText = "Актуальність";
dataGridView2.Columns[7].HeaderText = "Ціна";
dataGridView2.Columns[0].Width = 80;
dataGridView2.Columns[1].Width = 120;
dataGridView2.Columns[2].Width = 145;
dataGridView2.Columns[3].Width = 80;
dataGridView2.Columns[4].Width = 120;
dataGridView2.Columns[5].Width = 140;
dataGridView2.Columns[6].Width = 140;
dataGridView2.Columns[7].Width = 85;

```

```

DataTable userData2 = SQLServerConnection.executeSQL("SELECT Date, Name, Email, Phone, TrainingTime, Information, Topicality, Price
FROM Client join Courses on Client.Courses=Courses.IdCourses where NameCourses = 'Розробка web-сайтів'");

```

```

dataGridView3.DataSource = userData2;
dataGridView3.Columns[0].HeaderText = "Дата";
dataGridView3.Columns[1].HeaderText = "ФІО";
dataGridView3.Columns[2].HeaderText = "Email";
dataGridView3.Columns[3].HeaderText = "Телефон";
dataGridView3.Columns[4].HeaderText = "Бажаний час навчання";
dataGridView3.Columns[5].HeaderText = "Додаткова інформація";
dataGridView3.Columns[6].HeaderText = "Актуальність";
dataGridView3.Columns[7].HeaderText = "Ціна";
dataGridView3.Columns[0].Width = 80;
dataGridView3.Columns[1].Width = 120;
dataGridView3.Columns[2].Width = 145;
dataGridView3.Columns[3].Width = 80;
dataGridView3.Columns[4].Width = 120;
dataGridView3.Columns[5].Width = 140;
dataGridView3.Columns[6].Width = 140;
dataGridView2.Columns[7].Width = 85;

```

```

}

```

```

private void button11_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button7_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

```

```

private void button3_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

```

```

private void button6_Click(object sender, EventArgs e)

```

```

{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button10_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button9_Click(object sender, EventArgs e)
{
    Close();
}

private void button5_Click(object sender, EventArgs e)
{
    Close();
}

private void button4_Click(object sender, EventArgs e)
{
    Close();
}

private void button15_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form19 = new Form2();

    form19.Show();
}

private void button14_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form19 = new Form2();

    form19.Show();
}

private void button13_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 form19 = new Form2();

    form19.Show();
}

private void button18_Click(object sender, EventArgs e)
{
    Form21 af = new Form21();
    af.Owner = this;
    af.Show();
}

private void button20_Click(object sender, EventArgs e)
{
    Form4 af = new Form4();
    af.Owner = this;
    af.Show();
}

private void button19_Click(object sender, EventArgs e)
{
    SearchForm sf = new SearchForm();
    sf.Owner = this;
    sf.Show();
}

private void button17_Click(object sender, EventArgs e)
{
    this.Hide();
}

```

```

        Form2 form19 = new Form2();

        form19.Show();
    }

    private void button16_Click(object sender, EventArgs e)
    {
        Close();
    }
}

```

Form 20

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Адміністрування
{
    public partial class Form20 : Form
    {
        public Form20()
        {
            InitializeComponent();
        }

        private void noteBookBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.noteBookBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.administratorDataSet);
        }

        private void Form20_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.NoteBook". При необходимости она может быть
            // перемещена или удалена.
            this.noteBookTableAdapter.Fill(this.administratorDataSet.NoteBook);
        }

        private void textNoteBookLabel_Click(object sender, EventArgs e)
        {
        }

        private void textNoteBookTextBox_TextChanged(object sender, EventArgs e)
        {
        }

        private void dateLabel_Click(object sender, EventArgs e)
        {
        }

        private void dateDateTimePicker_ValueChanged(object sender, EventArgs e)
        {
        }

        private void noteBookDataGridView_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
        }

        private void button2_Click(object sender, EventArgs e)
        {
        }
    }
}

```

```

        Close();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form20 = new Form2();

        form20.Show();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void button17_Click(object sender, EventArgs e)
    {
    }
}
}
}

```

Form 21

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Адміністрування
{
    public partial class Form21 : Form
    {
        public Form21()
        {
            InitializeComponent();
        }

        private void clientBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.clientBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.administratorDataSet);
        }

        private void Form21_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.IdCoursesName". При необходимости она может
            // быть перемещена или удалена.
            this.idCoursesNameTableAdapter.Fill(this.administratorDataSet.IdCoursesName);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.N3AccountingInUTP". При необходимости она
            // может быть перемещена или удалена.
            this.n3AccountingInUTPTableAdapter.Fill(this.administratorDataSet.N3AccountingInUTP);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.Courses". При необходимости она может быть
            // перемещена или удалена.
            this.coursesTableAdapter.Fill(this.administratorDataSet.Courses);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "administratorDataSet.Client". При необходимости она может быть
            // перемещена или удалена.
            this.clientTableAdapter.Fill(this.administratorDataSet.Client);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            clientTableAdapter.Update(administratorDataSet.Client);
        }

        private void button17_Click(object sender, EventArgs e)
        {
            Form4 af = new Form4();
            af.Owner = this;
        }
    }
}

```

```

        af.Show();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        SearchForm sf = new SearchForm();
        sf.Owner = this;
        sf.Show();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        clientTableAdapter.Update(administratorDataSet.Client);
    }
}

```

Form 23

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Адміністрування
{
    public partial class Form23 : Form
    {
        Double resultValue = 0;
        String operationPerformed = "";
        bool isOperationPerformed = false;
        public Form23()
        {
            InitializeComponent();
        }

        private void button_click(object sender, EventArgs e)
        {
            if ((textBox_Result.Text == "0") || (isOperationPerformed))
                textBox_Result.Clear();
            isOperationPerformed = false;
            Button button = (Button)sender;
            if (button.Text == ".")
            {
                if (!textBox_Result.Text.Contains("."))
                    textBox_Result.Text = textBox_Result.Text + button.Text;
            }
            else
                textBox_Result.Text = textBox_Result.Text + button.Text;
        }

        private void operator_click(object sender, EventArgs e)
        {
            Button button = (Button)sender;
            if (resultValue != 0)

```

```

        {
            button15.PerformClick();
            operationPerformed = button.Text;
            labelCurrentOperation.Text = resultValue + " " + operationPerformed;
            isOperationPerformed = true;
        }
        else
        {
            operationPerformed = button.Text;
            resultValue = Double.Parse(textBox_Result.Text);
            labelCurrentOperation.Text = resultValue + " " + operationPerformed;
            isOperationPerformed = true;
        }
    }

    private void button4_Click(object sender, EventArgs e)
    {
        textBox_Result.Text = "0";
    }

    private void button5_Click(object sender, EventArgs e)
    {
        textBox_Result.Text = "0";
        resultValue = 0;
    }

    private void button15_Click(object sender, EventArgs e)
    {
        switch (operationPerformed)
        {
            case "+":
                textBox_Result.Text = (resultValue + Double.Parse(textBox_Result.Text)).ToString();
                break;
            case "-":
                textBox_Result.Text = (resultValue - Double.Parse(textBox_Result.Text)).ToString();
                break;
            case "*":
                textBox_Result.Text = (resultValue * Double.Parse(textBox_Result.Text)).ToString();
                break;
            case "/":
                textBox_Result.Text = (resultValue / Double.Parse(textBox_Result.Text)).ToString();
                break;
            default:
                break;
        }
        resultValue = Double.Parse(textBox_Result.Text);
        labelCurrentOperation.Text = "";
    }

    private void cancelButton_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form2 form19 = new Form2();

        form19.Show();
    }
}

```

RegisrerForm

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Адміністрування.Connection;

namespace Адміністрування
{
    public partial class RegisterForm : Form

```

```

{
    public RegisterForm()
    {
        InitializeComponent();
    }

    private void closeButton_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void clearButton_Click(object sender, EventArgs e)
    {
        clearControls();
        firstNameTextBox.Select();
    }

    private void clearControls()
    {
        foreach (TextBox tb in this.Controls.OfType<TextBox>())
        {
            tb.Text = string.Empty;
        }
    }

    private void RegisterForm_Load(object sender, EventArgs e)
    {
        loadUserData();
        firstNameTextBox.Select();
    }

    private void loadUserData()
    {
        DataTable userData = SQLServerConnection.executeSQL("SELECT (First_Name + ' ' + Last_Name) AS Fullname, Username FROM LoginTbl");
        dataGridView1.DataSource = userData;
        dataGridView1.Columns[0].HeaderText = "Full Name";
        dataGridView1.Columns[1].HeaderText = "Username";
        dataGridView1.Columns[0].Width = 234;
        dataGridView1.Columns[1].Width = 234;
    }

    private void deleteButton_Click(object sender, EventArgs e)
    {
        try
        {
            if (MessageBox.Show("Ви хочете назавжди видалити вибраний запис?",
                "Видалити дані:", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
                MessageBoxDefaultButton.Button1) == System.Windows.Forms.DialogResult.Yes)
            {
                SQLServerConnection.executeSQL("DELETE FROM LoginTbl WHERE Username = '" + dataGridView1.CurrentRow.Cells[1].Value + "'");

                loadUserData();

                MessageBox.Show("Запис видалено.",
                    "Видалити дані:",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                return;
            }
        }
        catch (Exception)
        {
            // An error occurred!
        }
    }

    private void saveButton_Click(object sender, EventArgs e)
    {
        MessageBoxButtons btn = MessageBoxButtons.OK;
        MessageBoxIcon ico = MessageBoxIcon.Information;
        string caption = "Збереження даних:";

        if (string.IsNullOrEmpty(firstNameTextBox.Text))

```

```

    {
        MessageBox.Show("Будь ласка, введіть ім'я", caption, btn, ico);
        firstNameTextBox.Select();
        return;
    }

    if (string.IsNullOrEmpty(lastNameTextBox.Text))
    {
        MessageBox.Show("Введіть прізвище.", caption, btn, ico);
        lastNameTextBox.Select();
        return;
    }

    if (string.IsNullOrEmpty(usernameTextBox.Text))
    {
        MessageBox.Show("Введіть ім'я користувача.", caption, btn, ico);
        usernameTextBox.Select();
        return;
    }

    if (string.IsNullOrEmpty(passwordTextBox.Text))
    {
        MessageBox.Show("Введіть пароль.", caption, btn, ico);
        passwordTextBox.Select();
        return;
    }

    if (string.IsNullOrEmpty(confirmPasswordTextBox.Text))
    {
        MessageBox.Show("Введіть пароль підтвердження.", caption, btn, ico);
        confirmPasswordTextBox.Select();
        return;
    }

    if (passwordTextBox.Text != confirmPasswordTextBox.Text)
    {
        MessageBox.Show("Ваш пароль та пароль підтвердження не збігаються.", caption, btn, ico);
        confirmPasswordTextBox.SelectAll();
        return;
    }

    string yourSQL = "SELECT Username FROM LoginTbl WHERE Username = '" + usernameTextBox.Text + "'";
    DataTable checkDuplicates = Адміністрування.Connection.SQLServerConnection.executeSQL(yourSQL);

    if (checkDuplicates.Rows.Count > 0)
    {
        MessageBox.Show("Ім'я користувача вже існує. Спробуйте інше ім'я користувача.",
            "Збереження даних:", MessageBoxButtons.OK, MessageBoxIcon.Information);
        usernameTextBox.SelectAll();
        return;
    }

    DialogResult result;
    result = MessageBox.Show("Ви хочете зберегти запис?", "Збереження даних:", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    if (result == DialogResult.Yes)
    {
        string mySQL = string.Empty;

        mySQL += "INSERT INTO LoginTbl (First_Name, Last_Name, Email_Address, Username, Password) ";
        mySQL += "VALUES ('" + firstNameTextBox.Text + "','" + lastNameTextBox.Text + "','" + emailTextBox.Text + "',";
        mySQL += "'" + usernameTextBox.Text + "','" + passwordTextBox.Text + "')";

        Адміністрування.Connection.SQLServerConnection.executeSQL(mySQL);

        MessageBox.Show("Запис успішно збережено.",
            "Збереження даних:", MessageBoxButtons.OK,
            MessageBoxIcon.Information);

        loadUserData();
        clearControls();
    }
}
}
}

```

SearchForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Адміністрування
{
    public partial class SearchForm : Form
    {
        public SearchForm()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form3 main = this.Owner as Form3;
            Form5 main1 = this.Owner as Form5;
            Form6 main2 = this.Owner as Form6;
            Form8 main4 = this.Owner as Form8;
            Form14 main10 = this.Owner as Form14;
            Form15 main11 = this.Owner as Form15;
            Form18 main14 = this.Owner as Form18;
            Form21 main16 = this.Owner as Form21;

            if (main16 != null)
            {
                for (int i = 0; i < main16.clientDataGridView.RowCount; i++)
                {
                    main16.clientDataGridView.Rows[i].Selected = false;
                    for (int j = 0; j < main16.clientDataGridView.ColumnCount; j++)
                        if (main16.clientDataGridView.Rows[i].Cells[j].Value != null)
                            if (main16.clientDataGridView.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                            {
                                main16.clientDataGridView.Rows[i].Selected = true;
                                break;
                            }
                }
            }

            if (main != null)
            {
                for (int i = 0; i < main.dataGridView2.RowCount; i++)
                {
                    main.dataGridView2.Rows[i].Selected = false;
                    for (int j = 0; j < main.dataGridView2.ColumnCount; j++)
                        if (main.dataGridView2.Rows[i].Cells[j].Value != null)
                            if (main.dataGridView2.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                            {
                                main.dataGridView2.Rows[i].Selected = true;
                                break;
                            }
                }
            }

            if (main != null)
            {
                for (int i = 0; i < main.dataGridView1.RowCount; i++)
                {
                    main.dataGridView1.Rows[i].Selected = false;
                    for (int j = 0; j < main.dataGridView1.ColumnCount; j++)
                        if (main.dataGridView1.Rows[i].Cells[j].Value != null)
                            if (main.dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
```

```

        {
            main.dataGridView1.Rows[i].Selected = true;
            break;
        }
    }
}

```

```

if (main != null)
{
    for (int i = 0; i < main.dataGridView3.RowCount; i++)
    {
        main.dataGridView3.Rows[1].Selected = false;
        for (int j = 0; j < main.dataGridView3.ColumnCount; j++)
            if (main.dataGridView3.Rows[i].Cells[j].Value != null)
                if (main.dataGridView3.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main.dataGridView3.Rows[i].Selected = true;
                    break;
                }
    }
}

```

```

if (main1 != null)
{
    for (int i = 0; i < main1.dataGridView1.RowCount; i++)
    {
        main1.dataGridView1.Rows[1].Selected = false;
        for (int j = 0; j < main1.dataGridView1.ColumnCount; j++)
            if (main1.dataGridView1.Rows[i].Cells[j].Value != null)
                if (main1.dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main1.dataGridView1.Rows[i].Selected = true;
                    break;
                }
    }
}

```

```

if (main1 != null)
{
    for (int i = 0; i < main1.dataGridView2.RowCount; i++)
    {
        main1.dataGridView2.Rows[1].Selected = false;
        for (int j = 0; j < main1.dataGridView2.ColumnCount; j++)
            if (main1.dataGridView2.Rows[i].Cells[j].Value != null)
                if (main1.dataGridView2.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main1.dataGridView2.Rows[i].Selected = true;
                    break;
                }
    }
}

```

```

if (main1 != null)
{
    for (int i = 0; i < main1.dataGridView3.RowCount; i++)
    {
        main1.dataGridView3.Rows[1].Selected = false;
        for (int j = 0; j < main1.dataGridView3.ColumnCount; j++)
            if (main1.dataGridView3.Rows[i].Cells[j].Value != null)
                if (main1.dataGridView3.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main1.dataGridView3.Rows[i].Selected = true;
                    break;
                }
    }
}

```

```

if (main2 != null)
{
    for (int i = 0; i < main2.dataGridView1.RowCount; i++)
    {
        main2.dataGridView1.Rows[1].Selected = false;
        for (int j = 0; j < main2.dataGridView1.ColumnCount; j++)

```

```

        if (main2.dataGridView1.Rows[i].Cells[j].Value != null)
            if (main2.dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
            {
                main2.dataGridView1.Rows[i].Selected = true;
                break;
            }
        }
    }
}

if (main2 != null)
{
    for (int i = 0; i < main2.dataGridView2.RowCount; i++)
    {
        main2.dataGridView2.Rows[1].Selected = false;
        for (int j = 0; j < main2.dataGridView2.ColumnCount; j++)
            if (main2.dataGridView2.Rows[i].Cells[j].Value != null)
                if (main2.dataGridView2.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main2.dataGridView2.Rows[i].Selected = true;
                    break;
                }
    }
}

if (main2 != null)
{
    for (int i = 0; i < main2.dataGridView3.RowCount; i++)
    {
        main2.dataGridView3.Rows[1].Selected = false;
        for (int j = 0; j < main2.dataGridView3.ColumnCount; j++)
            if (main2.dataGridView3.Rows[i].Cells[j].Value != null)
                if (main2.dataGridView3.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main2.dataGridView3.Rows[i].Selected = true;
                    break;
                }
    }
}
}

```

```

if (main4 != null)
{
    for (int i = 0; i < main4.dataGridView1.RowCount; i++)
    {
        main4.dataGridView1.Rows[1].Selected = false;
        for (int j = 0; j < main4.dataGridView1.ColumnCount; j++)
            if (main4.dataGridView1.Rows[i].Cells[j].Value != null)
                if (main4.dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main4.dataGridView1.Rows[i].Selected = true;
                    break;
                }
    }
}

```

```

if (main4 != null)
{
    for (int i = 0; i < main4.dataGridView2.RowCount; i++)
    {
        main4.dataGridView2.Rows[1].Selected = false;
        for (int j = 0; j < main4.dataGridView2.ColumnCount; j++)
            if (main4.dataGridView2.Rows[i].Cells[j].Value != null)
                if (main4.dataGridView2.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main4.dataGridView2.Rows[i].Selected = true;
                    break;
                }
    }
}

```

```

if (main4 != null)
{
    for (int i = 0; i < main4.dataGridView3.RowCount; i++)
    {
        main4.dataGridView3.Rows[1].Selected = false;
        for (int j = 0; j < main4.dataGridView3.ColumnCount; j++)
            if (main4.dataGridView3.Rows[i].Cells[j].Value != null)

```

```

        if (main4.dataGridView3.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
        {
            main4.dataGridView3.Rows[i].Selected = true;
            break;
        }
    }
}

```

```

if (main10 != null)
{
    for (int i = 0; i < main10.dataGridView1.RowCount; i++)
    {
        main10.dataGridView1.Rows[i].Selected = false;
        for (int j = 0; j < main10.dataGridView1.ColumnCount; j++)
            if (main10.dataGridView1.Rows[i].Cells[j].Value != null)
                if (main10.dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main10.dataGridView1.Rows[i].Selected = true;
                    break;
                }
    }
}
if (main10 != null)
{
    for (int i = 0; i < main10.dataGridView2.RowCount; i++)
    {
        main10.dataGridView2.Rows[i].Selected = false;
        for (int j = 0; j < main10.dataGridView2.ColumnCount; j++)
            if (main10.dataGridView2.Rows[i].Cells[j].Value != null)
                if (main10.dataGridView2.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main10.dataGridView2.Rows[i].Selected = true;
                    break;
                }
    }
}
if (main10 != null)
{
    for (int i = 0; i < main10.dataGridView3.RowCount; i++)
    {
        main10.dataGridView3.Rows[i].Selected = false;
        for (int j = 0; j < main10.dataGridView3.ColumnCount; j++)
            if (main10.dataGridView3.Rows[i].Cells[j].Value != null)
                if (main10.dataGridView3.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main10.dataGridView3.Rows[i].Selected = true;
                    break;
                }
    }
}
}

```

```

if (main11 != null)
{
    for (int i = 0; i < main11.dataGridView1.RowCount; i++)
    {
        main11.dataGridView1.Rows[i].Selected = false;
        for (int j = 0; j < main11.dataGridView1.ColumnCount; j++)
            if (main11.dataGridView1.Rows[i].Cells[j].Value != null)
                if (main11.dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main11.dataGridView1.Rows[i].Selected = true;
                    break;
                }
    }
}
if (main11 != null)
{
    for (int i = 0; i < main11.dataGridView2.RowCount; i++)
    {
        main11.dataGridView2.Rows[i].Selected = false;
        for (int j = 0; j < main11.dataGridView2.ColumnCount; j++)
            if (main11.dataGridView2.Rows[i].Cells[j].Value != null)
                if (main11.dataGridView2.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main11.dataGridView2.Rows[i].Selected = true;

```

```

        break;
    }
}
}
if (main11 != null)
{
    for (int i = 0; i < main11.dataGridView3.RowCount; i++)
    {
        main11.dataGridView3.Rows[1].Selected = false;
        for (int j = 0; j < main11.dataGridView3.ColumnCount; j++)
            if (main11.dataGridView3.Rows[i].Cells[j].Value != null)
                if (main11.dataGridView3.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main11.dataGridView3.Rows[i].Selected = true;
                    break;
                }
    }
}
}
}

```

```

if (main14 != null)
{
    for (int i = 0; i < main14.dataGridView1.RowCount; i++)
    {
        main14.dataGridView1.Rows[1].Selected = false;
        for (int j = 0; j < main14.dataGridView1.ColumnCount; j++)
            if (main14.dataGridView1.Rows[i].Cells[j].Value != null)
                if (main14.dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main14.dataGridView1.Rows[i].Selected = true;
                    break;
                }
    }
}
if (main14 != null)
{
    for (int i = 0; i < main14.dataGridView2.RowCount; i++)
    {
        main14.dataGridView2.Rows[1].Selected = false;
        for (int j = 0; j < main14.dataGridView2.ColumnCount; j++)
            if (main14.dataGridView2.Rows[i].Cells[j].Value != null)
                if (main14.dataGridView2.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main14.dataGridView2.Rows[i].Selected = true;
                    break;
                }
    }
}
if (main14 != null)
{
    for (int i = 0; i < main14.dataGridView3.RowCount; i++)
    {
        main14.dataGridView3.Rows[1].Selected = false;
        for (int j = 0; j < main14.dataGridView3.ColumnCount; j++)
            if (main14.dataGridView3.Rows[i].Cells[j].Value != null)
                if (main14.dataGridView3.Rows[i].Cells[j].Value.ToString().Contains(tbStr.Text))
                {
                    main14.dataGridView3.Rows[i].Selected = true;
                    break;
                }
    }
}
}
}
}

```

ДОДАТОК Б. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ
Кафедра Комп'ютерних наук

ДИПЛОМНА РОБОТА

на на ступінь вищої освіти магістр
із спеціальності F3 Комп'ютерні науки

• Інформаційна система адміністратора освітніх курсів на мові
програмування C# з використанням СКБД MySQL

Виконав: студент 6 курсу, групи КНДМ-62

Макарук Олександр Сергійович

Керівник: професор кафедри Комп'ютерних наук
д.ф.-м.н., професор Шикіула О.М.

Київ - 2025

1

Слайд 1

Актуальність дослідження

Адміністратор – це особа, яка здійснює роботу з якісного і ефективного обслуговування відвідувачів, консультує їх із питань, які стосуються товарів і послуг, що надає підприємство, організація. В наш динамічний час стрімко зростає попит на роботу адміністратора. Зараз відсутнє достатньо просте та зрозуміле програмне забезпечення для адміністратора освітніх курсів. Існуючі аналоги розраховані здебільшого на школи та вищі навчальні заклади, робота яких кардинально відрізняється від роботи освітніх курсів. Тому розробка програмного продукту, що полегшує роботу адміністратора освітніх курсів, є актуальною.

2

Слайд 2

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Об'єкт дослідження – інформаційна система адміністратора освітніх курсів.

Предмет дослідження – теоретичні та практичні аспекти розроблення та впровадження інформаційної системи адміністратора освітніх курсів.

Мета дослідження – створенні інформаційної системи адміністратора освітніх курсів, яка забезпечуватиме ефективне виконання його професійних обов'язків.

Наукове завдання :

- Дослідити роль і завдання освітніх курсів в навчальному процесі, Провести огляд та аналіз наявних програмних продуктів, орієнтованих на адміністрування, виявити їх функціонал, переваги та недоліки.
- Розробити базу даних, призначену для адміністрування освітніх курсів.
- Реалізувати інформаційну систему, яка сприятиме полегшенню роботи адміністратора освітніх курсів.
- Протестувати роботу інформаційної системи.

Методи дослідження – мова програмування C# з використанням СКБД MySQL в середовищі Microsoft Visual Studio

3

Слайд 3

1. ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ

Курсове навчання суттєво відрізняється як від освітнього процесу у закладах вищої освіти, так і від інших форм підготовки. Основними його ознаками є:

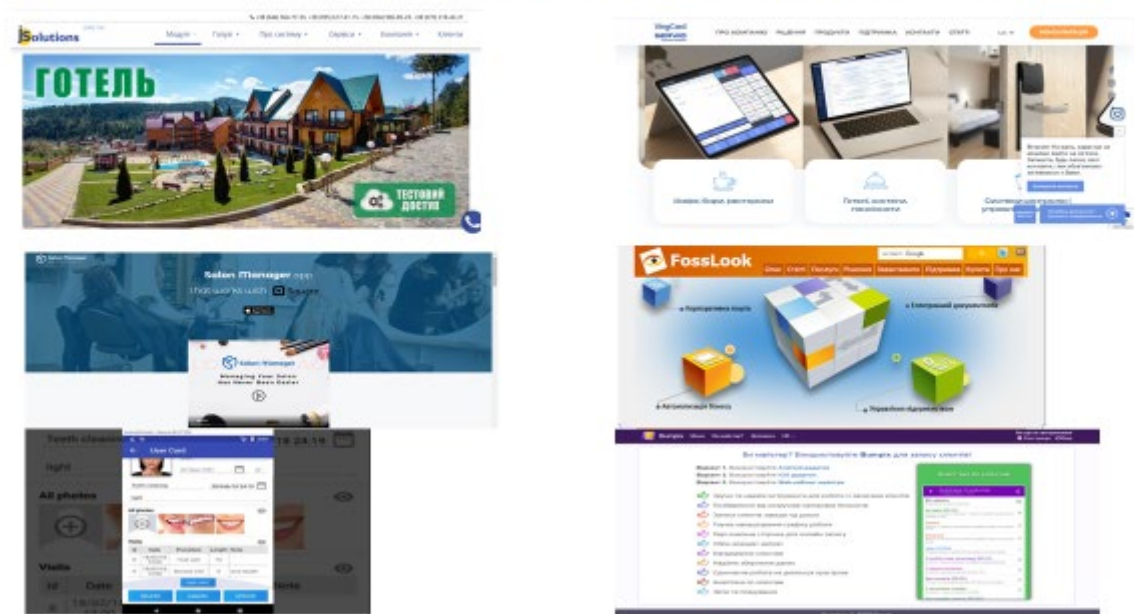
- інтенсивні та динамічні темпи засвоєння матеріалу;
- орієнтація на формування практичних навичок.

До основних обов'язків адміністратора навчального центру належать:

- приймання телефонних дзвінків та консультування потенційних слухачів із організаційних питань;
- реєстрація учасників курсів;
- організація старту навчальних груп у разі набору необхідної кількості слухачів (від 4 осіб) за одним курсом і на одну зміну.

Слайд 4

1. ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ



Слайд 5

1. ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ

Автоматизація процесів при адмініструванні готельного, ресторанного та сервісного бізнесу є важливим інструментом для підвищення ефективності роботи, мінімізації помилок персоналу та покращення якості обслуговування клієнтів. Сучасні програмні рішення (jSolutions, SERVIO, Salon Manager, FossLook, «Ресепшен», Bumprix) забезпечують інтегрований контроль бізнес-процесів, управління клієнтською базою, фінансовим та складським обліком, комунікацією з гостями, а також створення аналітики й звітності.

Впровадження сучасного програмного забезпечення в різних сферах послуг забезпечує:

- зниження ймовірності помилок персоналу;
- оптимізацію бізнес-процесів і підвищення продуктивності;
- зручність управління фінансами, клієнтською базою та складським обліком;
- формування лояльності клієнтів і покращення якості обслуговування;
- зростання прибутковості підприємств.

Таким чином, автоматизація є не просто додатковим інструментом, а стратегічним чинником успіху підприємств готельно-ресторанної та сервісної сфери, який дозволяє їм бути конкурентоспроможними та відповідати сучасним вимогам ринку.

Слайда 6

2. ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ДЛЯ СТВОРЕННЯ СИСТЕМИ

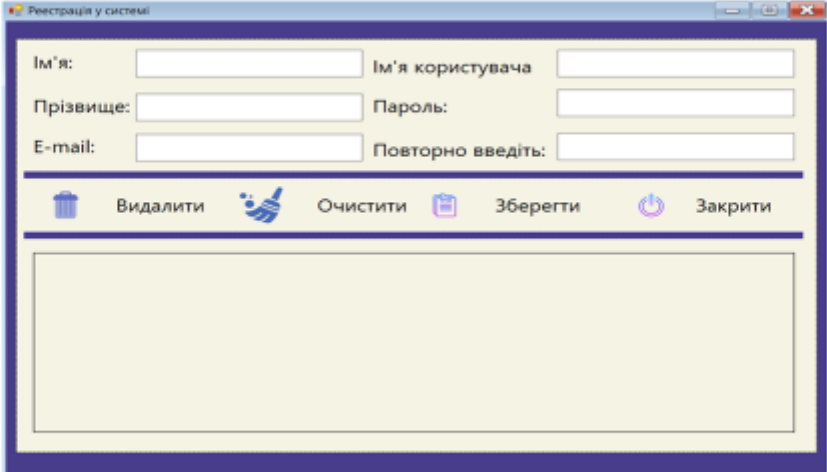
Для реалізації проекту було обрано наступні технології:

- Для написання додатку використана об'єктно-орієнтована мова програмування C# з використанням розширення Windows Forms платформи .NET Framework.
- MySQL є системою управління базами даних, яка забезпечує ефективну організацію, збереження та обробку інформації.
- Розробка проводилась в середовищі Microsoft Visual Studio.

7

Слайд 7

3. ОПИС РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ



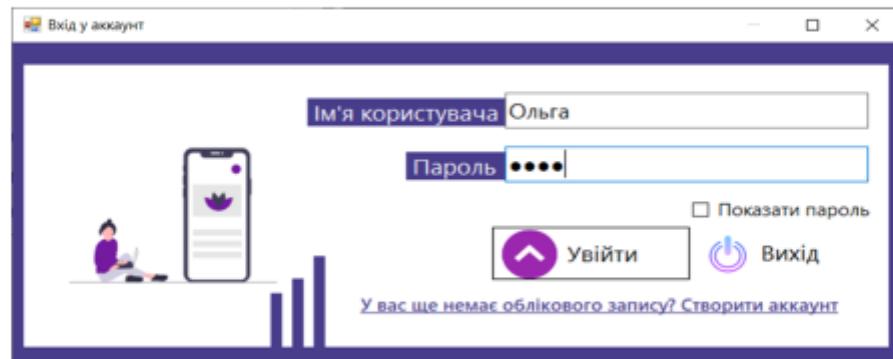
The screenshot shows a Windows-style application window titled "Реєстрація у системі" (System Registration). The form contains several input fields for user registration: "Ім'я:" (Name), "Ім'я користувача:" (Username), "Прізвище:" (Surname), "Пароль:" (Password), "E-mail:", and "Повторно введіть:" (Repeat). Below the input fields is a horizontal bar with five buttons: "Видалити" (Delete) with a trash icon, "Очистити" (Clear) with a hand icon, "Зберегти" (Save) with a floppy disk icon, and "Закрити" (Close) with a power icon. The bottom of the window features a large, empty rectangular area.

Форма реєстрації

8

Слайд 8

3. ОПИС РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

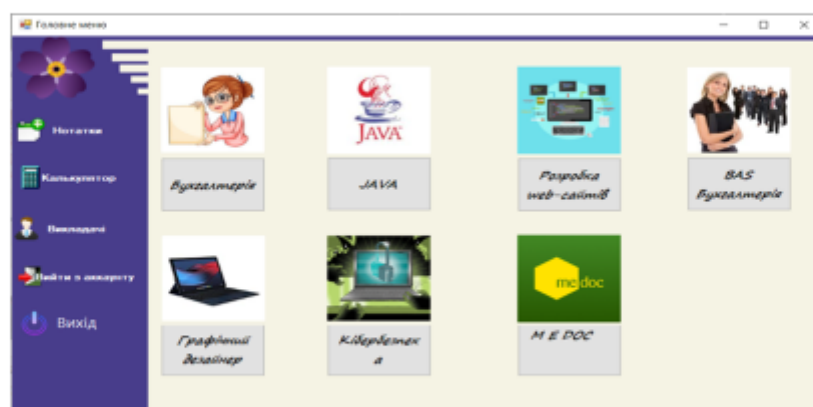


Форма входу в систему

9

Слайд 9

3. ОПИС РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ



Форма «Головне меню»

10

Слайд 10

3. ОПИС РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Форма курсів Java

11

Слайд 11

3. ОПИС РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

The figure consists of six screenshots of the 'Epidemiology' software interface, arranged in a 3x2 grid. Each screenshot shows a different view of the software, including data tables, charts, and text descriptions.

- Top Left:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.
- Top Middle:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.
- Top Right:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.
- Middle Left:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.
- Middle Middle:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.
- Middle Right:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.
- Bottom Left:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.
- Bottom Middle:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.
- Bottom Right:** Shows a table with columns: Date, Age, Sex, Name, Vaccination status, and various counts. The table is titled 'Epidemiology'.

Форми різних курсів

12

Слайд 12

3. ОПИС РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

IDCard	Date	Name	Email	Phone	WaitingTime	Information	Counters	Specialty
1	05.05.2025	Анна Фролова	annafr@mail.com	50545105	До 10 хвилин	Бюджет у корист.	1	Чоловік-нарколог
3	22.05.2025	Степан-Павло	stepan@mail.com	12545222	3-14 хвилин	Закрити курс	2	Чоловік-нарколог
5	02.07.2025	Карпенко Карина	karin@mail.com	57545455	в 12	апр	7	Жінка-психолог
7	15.07.2025	Валентина Тонка	valen@mail.com	50545455	3-18 до 17 год.	Діагностика	2	Жінка-психолог
9	02.07.2025	Ліза Ушакова	liza@mail.com	50222222	у вечір	Бюджет у 30	3	Чоловік-нарколог
10	08.07.2025	Тарас Шевченко	taras@mail.com	00275752	15-20	інформаційно-нарколог	4	Чоловік-нарколог
11	10.05.2025	Роздубинський	roz@mail.com	00022222	10-14	Жінка-психолог	5	Чоловік-нарколог
12	07.09.2025	Вікторія Марія	viktor@mail.com	9875875	09-15		6	Чоловік-нарколог
13	01.02.2025	Мельник Степан	mel@mail.com	96255555	15-20		7	Чоловік-нарколог
14	05.04.2025	Кристіна Кат.	kris@mail.com	50457512	9-16	Група медичного	8	Чоловік-нарколог

Код: Дата: Активність:
 ФІО: Email: Ім'я:
 Телефон: Карт:
 Бюджет на навігатор:
 Додати інформацію:

Форма «Редагування/Додавання»

13

Слайд 13

3. ОПИС РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

SearchForm

Пошук

Введіть дані для пошуку

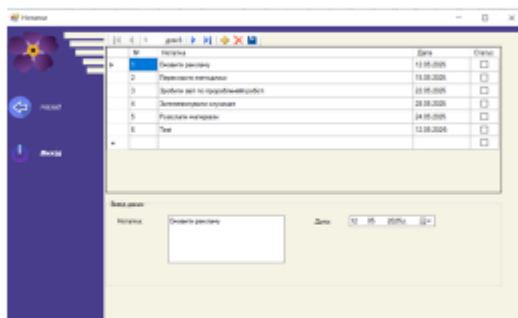
Знайти Закрити

Вікно «Знайти слухача»

14

Слайд 14

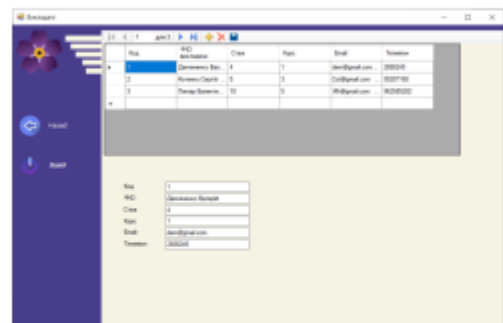
3. ОПИС РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ



The 'Notes' form displays a table with columns: ID, Notes, Date, and Status. The table contains five rows of data. Below the table, there is a section for 'New record' with input fields for 'Notes' and 'Date'.

ID	Notes	Date	Status
1	Одноразовий розрахунок	12.05.2025	☐
2	Повторюваний розрахунок	12.05.2025	☐
3	Розрахунок за періодичними даними	12.05.2025	☐
4	Додатковий розрахунок	12.05.2025	☐
5	Розрахунок за періодом	12.05.2025	☐
6	Інше	12.05.2025	☐

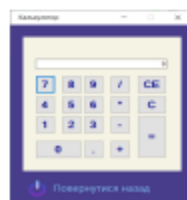
Форма «Нотатки»



The 'Instructor' form displays a table with columns: ID, Name, Surname, Date, Name, Email, and Telephone. The table contains three rows of data. Below the table, there is a section for 'New record' with input fields for 'Name', 'Surname', 'Date', 'Name', 'Email', and 'Telephone'.

ID	Name	Surname	Date	Name	Email	Telephone
1	Олександр	Василь	12	1	vasil@ukr.net	9876543
2	Олександр	Василь	12	2	vasil@ukr.net	9876543
3	Олександр	Василь	12	3	vasil@ukr.net	9876543

Форма «Викладачі»



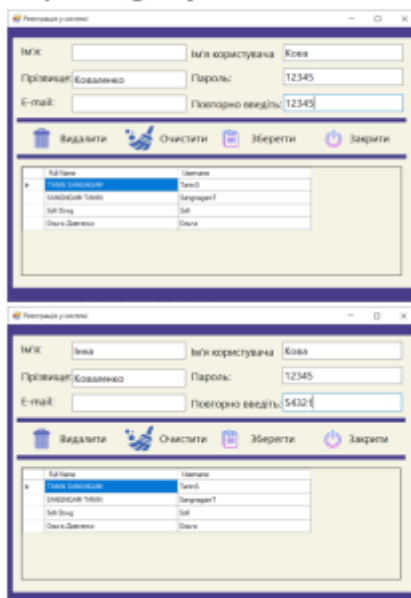
The 'Calculator' form displays a standard calculator interface with a numeric keypad and a display area. A button labeled 'Повернутися назад' (Back) is located at the bottom.

Форма «Калькулятор»

15

Слайд 15

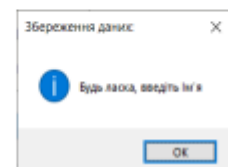
3. ТЕСТУВАННЯ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ



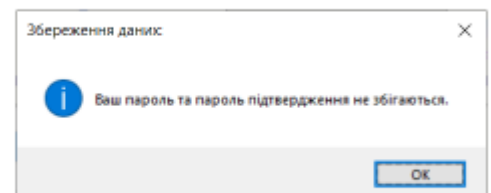
The 'Registration' form displays input fields for 'ІМ' (Name), 'ІМ' (Surname), 'Прізвище' (Surname), 'Пароль' (Password), and 'E-mail'. Below the form, there is a table with columns: ID, Name, Surname, and Date. The table contains three rows of data.

ID	Name	Surname	Date
1	Олександр	Василь	12.05.2025
2	Олександр	Василь	12.05.2025
3	Олександр	Василь	12.05.2025

Введення даних без імені адміністратора



The 'Data Saved' dialog box displays a message: 'Будь ласка, введіть ім'я' (Please, enter the name). The 'OK' button is highlighted.



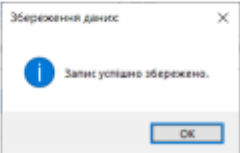
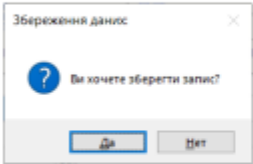
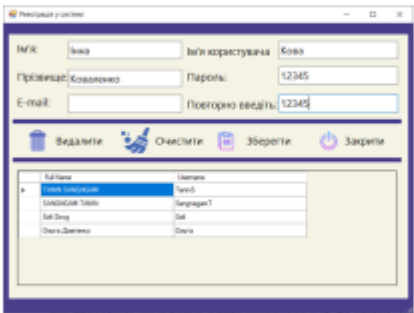
The 'Data Saved' dialog box displays a message: 'Ваш пароль та пароль підтвердження не збігаються.' (Your password and confirmation password do not match). The 'OK' button is highlighted.

Введення паролів, що не співпадають

16

Слайд 16

3. ТЕСТУВАННЯ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

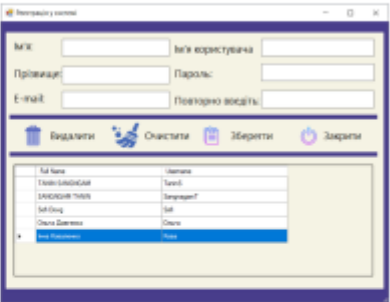


Введення вірно всіх даних

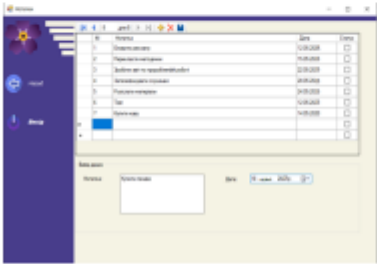
17

Слайд 17

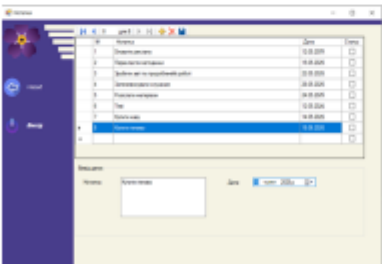
3. ТЕСТУВАННЯ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ



Проведення реєстрації в системі



Нова нотатка



Збережені дані в нотатці

18

Слайд 18

ВИСНОВКИ

Мета атестаційної роботи магістра полягала в розробці інформаційної системи адміністратора освітніх курсів для полегшення його роботи. Інформаційна система повинна має зручний та інтуїтивно зрозумілий інтерфейс і надає можливість адміністратору переглядати та редагувати інформацію.

Для її реалізації було зроблено:

- Досліджено роль і завдання освітніх курсів в навчальному процесі. Проведено огляд та аналіз наявних програмних продуктів, орієнтованих на адміністрування, виявлено їх функціонал, переваги та недоліки.

19

Слайд 19

ВИСНОВКИ

- За допомогою об'єктно-орієнтованої мови програмування C# з використанням СКБД MySQL в середовищі Microsoft Visual Studio було створено інформаційну систему адміністратора освітніх курсів із інтуїтивно зрозумілим і дружнім інтерфейсом, що робить її зручною для будь-якої категорії користувачів, які займаються адмініструванням освітніх курсів. Ця система дає адміністратору можливість додавати нові дані про слухачів, редагувати та видаляти їх, а також здійснювати перегляд інформації. Завдяки цьому адміністратор навчального центру може ефективно працювати з базою слухачів, оперативно зв'язуватися з ними для уточнення організаційних питань. У процесі розробки були реалізовані модулі автентифікації, внесення та збереження даних, перегляду інформації та пошуку по слухачах. Кожен із цих модулів було адаптовано під специфіку роботи адміністратора та інтегровано в систему.
- Протестована робота інформаційної системи. Результати тестування підтвердили працездатність продукту та його готовність до подальшого застосування.

20

Слайд 20

ВИСНОВКИ

Практична цінність атестаційної роботи бакалавра полягає в створенні інформаційної системи адміністратора освітніх курсів з метою полегшенні його роботи. Розроблена інформаційна система виключить рутину роботи, а також буде запобігати людським помилкам.

У перспективі вдосконалення програмного забезпечення можливим напрямком є розробка модулів управління доступом, а також покращення інтерфейсу сайту для підвищення його інформативності, привабливості та зручності використання.