Зміст

[Вступ 3](#_Toc25598658)

[1 Загальна частина 5](#_Toc25598659)

[1.1 Аналіз прикладної області 5](#_Toc25598660)

[1.2 Вимоги до майбутньої бази даних 6](#_Toc25598661)

[1.3 Проектування бази даних 7](#_Toc25598662)

[1.3.1 Визначення сутностей бази даних, та зв’язки між ними 7](#_Toc25598663)

[1.3.2 Визначення об’єктів бази даних та їх властивостей 8](#_Toc25598664)

[1.3.3 Розробка концептуальної схеми бази даних, згідно стандарту IDF1X 11](#_Toc25598665)

[1.4 Обґрунтування вибору середовища для реалізації завдання 14](#_Toc25598666)

[1.4.1 Обґрунтування вибору середовища для реалізації бази даних 14](#_Toc25598667)

[1.4.2 Обґрунтування вибору середовища програмування 15](#_Toc25598668)

[2 Спеціальна частина 17](#_Toc25598669)

[2.1 Проектування в обраному середовищі 17](#_Toc25598670)

[2.2 Проектування інтерфейсу програми 18](#_Toc25598671)

[2.3 Програмна реалізація 22](#_Toc25598672)

[2.4 Інструкція користувача 25](#_Toc25598673)

[Загальні висновки 26](#_Toc25598674)

[Список використаної літератури 28](#_Toc25598675)

[Додаток А 29](#_Toc25598676)

[Додаток Б 31](#_Toc25598677)

[Додаток В 50](#_Toc25598678)

Вступ

Основи сучасної інформаційної технології складають бази даних і системи управління базами даних, роль яких як єдиного засобу зберігання, обробки і доступу до великих обсягів інформації постійно зростає. При цьому істотним є постійне підвищення обсягів інформації, що зберігається в базі даних, що тягне за собою вимогу збільшення продуктивності таких систем.

Комп’ютеризація навчальних закладів торкнулася всіх сторін їх життя, в тому числі і роботи співробітників, які може використовувати інформаційні технології (ІТ) для полегшення своєї роботи.

В останні роки світ змінюється дуже швидкими темпами, і викладачам потрібно використовувати більш сучасні рішення для автоматизації своєї роботи, адже в університеті навчається велика кількість студентів, то все гострішою стає проблема автоматизованого опрацювання інформації щодо обліку студентів.

Все це вимагає кропіткої роботи з різноманітними паперами. При цьому багато даних повинно співставлятися з даними із особових карток студентів, що знижує продуктивність праці методистів деканатів. Ось чому все більшою постає потреба в автоматизованому опрацюванні цієї інформації.

Для розв'язання даної задачі доцільно використовувати базу даних, а весь процес реєстрації та обробки даних повинен бути автоматизованим, оскільки система управління базами даних (програмне забезпечення бази даних) швидко здійснює пошук необхідної інформації в базі даних, формує вихідні документи, проводить зміни в записах бази даних.

Тому, темою курсового проекту є «Проектування і розробка бази даних обліку студентів заочного відділення».

Бази даних - сукупність даних, організована за певними правилами, що передбачає загальні принципи опису, зберігання, маніпулювання даними, незалежними від прикладних програм.

Система управління базами даних - сукупність програм, призначених для управління баз даних і можливості отримання користувачами необхідної інформації з бази.

Метою даної курсової роботи є розробка і реалізація бази даних для обліку студентів заочного відділення та розробка програмного додатку для демонстрації роботи з базою даних.

Розроблена база даних в подальшому може використовуватись будь-яким вищим навчальним закладом для полегшеної роботи з інформацією про студентів даного відділення та зручного сприйняття її користувачем.

1 Загальна частина

1.1 Аналіз прикладної області

В вищих навчальних закладах навчається величезна кількість студентів. Усі документи та інформація збережена на папері, тому кожного разу не зручно гортати велику кількість паперів для перевірки потрібної інформації. Тому для підвищення зручності зберігання і використовування про них інформацію, ми створимо базу даних на прикладі Криворізького національного університету. База даних потрібна для того, щоб у будь-який момент часу можна було б дізнатися будь-яку інформацію стосовно студента: форму навчання, місце проживання і т.д.

Для розробки такої бази нам необхідно знати таку інформацію про студентів: ПІБ, контактний телефон, домашня адреса (включаючи область, місто (селище і т.д.), рік народження. Якщо, припустимо, неможливо зв'язатися зі студентом (довго не було в ВУЗі, змінилися контактні дані і т. д.) або про що-то необхідно повідомити їх батькам, то нам буде потрібно їх інформація, а саме: ПІБ мами і її контактні дані, а також ПІБ і контактна інформація тата. Якщо немає батьків, то вказуємо контактні дані родичів (близьких).

Також у нашій базі необхідно вказувати групу, де враховуємо рік набору тієї або іншої групи. Так як група належить до спеціальності, то ми повинні занести таку інформацію: повну назву спеціальності та шифр. Будь-яка спеціальність відноситься до певного факультету, про який нам потрібно знати повну назву, ПІБ декана, номер телефону деканату.

Відповідно до предметної області система будується з урахуванням таких особливостей:

* кожна факультет та кафедра мають свій унікальний номер та назву;
* кожен викладач має свій унікальний номер та інформацію про нього;
* кожен студент має свій унікальний номер та інформацію про нього.

Використовуючи базу даних у програмному додатку ми зможемо:

1. Швидко надавати та отримувати потрібну для нас інформацію за короткий час.
2. Компактно зберігати великого обсягу інформацію.
3. Отримувати інформацію в будь-який час.

Отже, ми проаналізували предметну область для майбутнього створення бази даних обліку студентів заочного відділення.

1.2 Вимоги до майбутньої бази даних

Аналіз прикладної області показав, що база даних для обліку студентів заочного відділення повинна відповідати наступним вимогам:

База даних повинна містити не менше 8 таблиць (головних, підлеглих, довідкових).

Створити систему управління базою даних (СУБД) для демонстрації роботи з базою даних. Інтерфейс програмного додатку повинен містити по одному прикладу форм для додавання, перегляду та редагування даних основних об’єктів, підлеглих та довідкових, форми для пошуку та обробки інформації, друку вихідних документів.

У розроблену базу даних потрібно буде закласти наступну інформацію:

дані про викладачів, а саме: прізвище, ім’я, по батькові, а також являються вони деканами або секретарями;

данні про студентів, а саме: прізвище, ім’я, по батькові, група;

інформація про дисципліни, а саме: назва предмету;

дані про факультети та кафедри, а саме: їх назва.

Інформаційна система повинна підтримувати такі функції:

виконувати пошук та фільтрацію за прізвищем, ім’ям, по батькові;

сортування за прізвищем, ім’ям, по батькові людей, за назвою факультети та кафедри, за датою період сесії;

Реалізувати запити на вибірку даних за різними типами полів (2 простих, 3 складних та 2 вкладених), запити на створення таблиці, вилучення та оновлення даних (по одному).

Інформаційною системою мають формуватися наступні звіти: «Групи», «Студенти певної групи».

Отже, нами були розглянуті вимоги до майбутньої бази даних.

1.3 Проектування бази даних

1.3.1 Визначення сутностей бази даних, та зв’язки між ними

При аналізі предметної області було встановлено, що БД повинна містити такі сутності: «Викладачі», «Студенти», «Куратори», «Декани», «Завідувачі», «Кафедри», «Факультети», «Групи», «Групи і куратори», «Групи і студенти» та «Дисципліни».

Сутність «Викладачі» повинна містити інформацію про викладачів, а саме: прізвище, ім’я, по батькові, дата народження.

Сутність «Студенти» повинна містити інформацію про студентів, а саме: прізвище, ім’я, по батькові, дата народження.

Сутність «Декани» та «Секретарі» – це сутності, які будуть зв’язана з сутністю «Викладачі».

Сутність «Кафедри» повинна містити назву кафедри. Пов’язана з сутністю «Факультети» та «Секретарі».

Сутність «Факультети» повинна містити назву факультету. Пов’язана з сутністю «Декани».

Сутність «Групи» повинна містити назву групи та номер курсу. Пов’язана з сутністю «Кафедри».

Сутність «Групи і секретарі» – це сутність, яка буде містити у собі дані про зв’язок з сутностями «Групи» та «Секретарі».

Сутність «Групи і студенти» – це сутність, яка буде містити у собі дані про зв’язок з сутностями «Групи» та «Студенти».

Сутність «Дисципліни» повинна містити назву дисципліни. Пов’язана з сутністю «Викладачі».

Отже, нами було розглянуто всі сутності бази даних та зв’язки між цими сутностями.

1.3.2 Визначення об’єктів бази даних та їх властивостей

Опираючись на сукупність всіх вище перерахованих, у пункті 1.3.1, сутностей можна визначити об’єкти бази даних та їх властивості. Далі буде наведено табличне представлення сутностей та повний опис властивостей їх атрибутів.

Сутність – модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. Модель сутність-зв’язок є результатом систематичного процесу, який описує та визначає деяку предметну область.

Опис сутності «Викладачі» наведено в таблиці 1.1.

Таблиця 1.1 – опис сутності «Викладачі»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_Teachers | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Surname | Прізвище | nvarchar | Не нульове, за замовчуванням прізвище викладача |
| Name | Ім’я | nvarchar | Не нульове, за замовчуванням ім’я викладача |
| Patronimyc | По батькові | nvarchar | Не нульове, за замовчуванням по батькові викладача |

Опис сутності «Дисципліни» наведено в таблиці 1.2.

Таблиця 1.2 – опис сутності «Дисципліни»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_Subjects | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Name | Назва | nvarchar | Не нульове, за замовчуванням назва предмету |
| Id\_Teachers | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |

Опис сутності «Декани» наведено в таблиці 1.3.

Таблиця 1.3 – опис сутності «Декани»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_Deans | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Id\_Teachers | Ід. типу | int | Не нульове, зовнішній ключ |

Опис сутності «Факультети» наведено в таблиці 1.4.

Таблиця 1.4 – опис сутності «Факультети»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_Faculties | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Name | Назва | nvarchar | Не нульове, за замовчуванням назва факультету |
| Id\_Deans | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |

Опис сутності «Кафедри» наведено в таблиці 1.5.

Таблиця 1.5 – опис сутності «Кафедри»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_Departments | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Name | Назва | nvarchar | Не нульове, за замовчуванням назва кафедри |
| Id\_Faculties | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Id\_Heads | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |

Опис сутності «Секретар» наведено в таблиці 1.6.

Таблиця 1.6– опис сутності «Секретар»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_Secretaries | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Id\_Teachers | Ід. типу | int | Не нульове, зовнішній ключ |
| Id\_Departments | Ід. типу | int | Ключове поле, ідентифікуюче, не нульове, унікальне |

Опис сутності «Групи і секретар» наведено в таблиці 1.7.

Таблиця 1.7 – опис сутності «Групи і секретар»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_GroupsSecretaries | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Id\_Groups | Ід. | int | Ключове поле, ідентифікуюче, не нульове |
| Id\_Secretaries | Ід. | int | Ключове поле, ідентифікуюче, не нульове |
| Session\_begin | Початок сесії | date | Не нульове, за замовчуванням дата початку сесії |
| Session\_end | Кінець сесії | date | Не нульове, за замовчуванням дата кінця сесії |

Опис сутності «Групи» наведено в таблиці 1.8.

Таблиця 1.8– опис сутності «Групи»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_Groups | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Name | Назва | nvarchar | Не нульове |
| Year | Курс | int | Не нульове |
| Id\_Departments | Ід. типу | int | Ключове поле, ідентифікуюче, не нульове, унікальне |

Опис сутності «Групи і студенти» наведено в таблиці 1.9.

Таблиця 1.9 – опис сутності «Групи і студенти»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_GroupsStudents | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Id\_Groups | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Id\_Students | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |

Опис сутності «Студенти» наведено в таблиці 1.10.

Таблиця 1.10 – опис сутності «Студенти»:

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | | Тип | Примітка |
| Name | Ім’я |
| Id\_Students | Ід. | int | Ключове поле, ідентифікуюче, не нульове, унікальне |
| Surname | Прізвище | nvarchar | Не нульове, за замовчуванням приймає прізвище |
| Name | Ім’я | nvarchar | Не нульове, за замовчуванням приймає ім’я |
| Patronimyc | По батькові | nvarchar | Не нульове, за замовчуванням приймає по батькові |

Отже, нами було визначено об’єкти бази даних та їх властивості.

1.3.3 Розробка концептуальної схеми бази даних, згідно стандарту IDF1X

Аналіз прикладної області показав, що база даних «Облік студентів заочного відділення» повинна складатися з одинадцяти таблиць (головних, підлеглих), тому необхідно спроектувати концептуальну схему для майбутньої БД, згідно стандарту IDF1X. Графічне зображення концептуальної схеми наведено на рисунку 1.1.

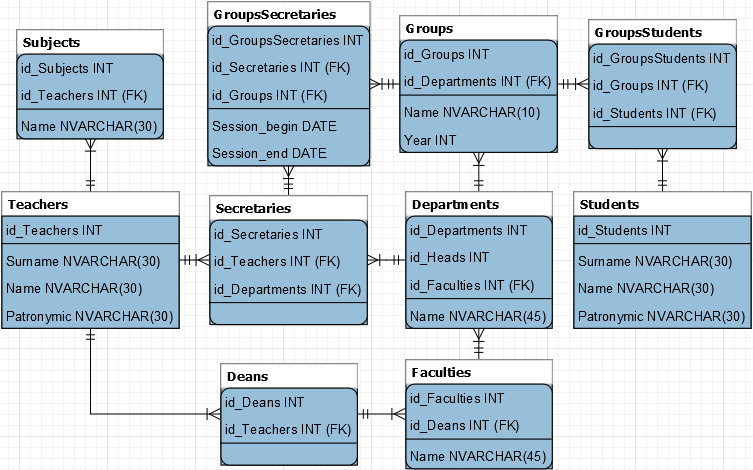


Рисунок 1.1 – Концептуальна схема бази даних

Маючи концептуальну схему бази даних можна повністю описати обмеження посилальної цілісності.

Відношення Teachers – Subjects має таке обмеження посилальної цілісності: ON DELETE: RESTRICT; ON UPDATE: CASCADE. Тому що при видалені основної інформації не повинна видалятися додаткова інформація, а при оновлені каскадно змінювати свої значення.

Відношення Teachers – Curators має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, таке саме і при оновлені даних.

Відношення Teachers – Deans має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, таке саме і при оновлені даних.

Відношення Teachers – Heads має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, таке саме і при оновлені даних.

Відношення Departments – Heads має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, таке саме і при оновлені даних.

Відношення Faculties – Deans має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, таке саме і при оновлені даних.

Відношення Faculties – Departments має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, таке саме і при оновлені даних.

Відношення Departments – Groups має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, таке саме і при оновлені даних.

Відношення Groups – GroupsCurators має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, таке саме і при оновлені даних.

Відношення Curators – GroupsCurators має таке обмеження посилальної цілісності: ON DELETE: RESTRICT; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, а при оновлені каскадно змінювати свої значення.

Відношення Groups – GroupsStudents має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE. Тому що при видалені основної інформації не повинна видалятися додаткова інформація, таке саме і при оновлені даних.

Відношення Students – GroupsStudents має таке обмеження посилальної цілісності: ON DELETE: RESTRICT; ON UPDATE: CASCADE. Тому що при видалені основної інформації повинна видалятися і вся додаткова інформація, а при оновлені каскадно змінювати свої значення.

Отже, нами було розроблено концептуальну схему бази даних, а також описано обмеження посилальної цілісності.

1.4 Обґрунтування вибору середовища для реалізації завдання

1.4.1 Обґрунтування вибору середовища для реалізації бази даних

Для розробки бази даних нам знадобляться такі інструменти: для доступу до всіх компонентів БД, керування ними, їх налаштування, розробки та адміністрування.

Весь перелічений функціонал можна знайти в таких середовищах: SQL Server Management Studio та Denver phpMyAdmin.

phpMyAdmin – це безкоштовний програмний інструмент, написаний на PHP, призначений для керування адмініструванням MySQL через Інтернет. phpMyAdmin підтримує широкий спектр операцій на MySQL та MariaDB. Часто використовувані операції (керування базами даних, таблицями, стовпцями, зв’язками, індексами, користувачами, дозволами тощо), можуть виконуватися через користувальницький інтерфейс, хоча також є можливість безпосередньо виконувати будь-який SQL-запит.

SQL Server Management Studio (SSMS) – це інтегроване середовище для управління будь-якою інфраструктурою SQL. Використовується для доступу, налаштування, адміністрування та розробки всіх компонентів SQL Server, а також управління ними. Середа SSMS надає єдину повнофункціональну службову програму, яка поєднує в собі велику групу графічних інструментів з рядом відмінних редакторів сценаріїв для доступу до служби SQL Server для розробників і адміністраторів баз даних.

Компоненти середовища SQL Server Management Studio:

* оглядач об'єктів містить відомості про всі серверах, які підключені до нього;
* вікно запитів відображає результати запитів;
* властивості відображає час запуску запиту, кількість повернутих рядків і відомості про підключення;
* оглядач шаблонів містить різні готові шаблони T-SQL. Ці шаблони можна використовувати для виконання різних дій, таких як створення або резервне копіювання баз даних;
* докладні відомості оглядача об'єктів використовується для одночасного управління декількома об'єктами. Наприклад, в цьому вікні можна вибрати кілька баз даних і одночасно видалити їх або створити сценарії для них.

Отже, для створення бази даних нами було обрано середовище SQL Server Management Studio, тому що, відповідно до обраного середовища програмування, яке ми розглянемо пізніше, нам необхідно буде створити базу даних типу SQL Server.

1.4.2 Обґрунтування вибору середовища програмування

Для створення демонстраційного програмного додатку, нам знадобляться такі інструменти: для створення зручного і зрозумілого графічного інтерфейсу, для зрозумілої компіляції коду програми, для можливості роботи з базами даних, для створення запитів до БД.

Всі інструменти, які було перелічено попередньо, можна знайти у таких середовищах програмування: «Sublime Text» (мови PHP, HTML, CSS) та «Visual Studio» (мова C#).

«Sublime Text» — швидкий кросплатформенний текстовий редактор. Sublime Text підтримує велику кількість мов програмування і має можливість підсвічування синтаксису для C ++, C #, CSS, HTML, Java, JavaScript, PHP, Python, SQL і XML.

PHP (Personal Home Page) - це в кінцевому рахунку текст, одержуваний веб-сервером і перетворюваний в набір команд і інформації для веб-браузера. За допомогою цієї мови програмування можна працювати з базою даних.

HTML (HyperText Markup Language) - мова розмітки гіпертексту - призначений для створення Web-сторінок.

CSS (Cascading Style Sheets) - це потужний стандарт на основі текстового формату, який визначає представлення даних в браузері.

Visual Studio C# це середовище програмування, призначене для розробки найрізноманітніших додатків, призначених для виконання в середовищі .NET Framework. Мова C# проста, безпечна і об’єктно-орієнтована. Завдяки безлічі нововведень C# забезпечує можливість швидкої розробки додатків, але при цьому зберігає виразність і елегантність.

У Visual Studio C# є багато візуальних компонентів які поділяються на:

Стандартні елементи керування - містить найбільш часто вживані компоненти, що фігурують у всіх програмах.

Контейнери – містить компоненти які використовуються як контейнери для вміщення в себе інших різноманітних компонентів.

Меню та панелі інструментів – містить компоненти меню, контекстного меню та панелі інструментів.

Дані – містить ту частину інтерфейсу користувача, що пов’язана з даними. Тут є компоненти, що дозволяють вам представляти дані користувачу будь-яким способом.

Компоненти – містить компоненти для роботи з операційною системою.

Друк – містить компоненти для швидкого будування звітів по базам даних.

Діалогові вікна – містить компоненти, за допомогою яких можна встановити діалогові вікна для виконання задач загального характеру (відкриття файлу, надання стилів тексту, надання кольорів, тощо).

Отже, нами було обрано середовище програмування «Visual Studio» (мова C#), тому що воно має всі необхідні інструменти для роботи з базами даних, а також засоби для зручного створення форм, на відміну від «Sublime ++».

2 Спеціальна частина

2.1 Проектування в обраному середовищі

Згідно з розробленою концептуальною схемою бази даних «Облік студентів заочного відділення», яка показана на рисунку 1.1, можна побачити, що наша БД повинна складатися з десяти таблиць (головних, підлеглих), тому необхідно спроектувати її в обраному нами середовищі – Microsoft SQL Server Management Studio. Графічне зображення схеми розробленої БД наведено на рисунку 2.1.

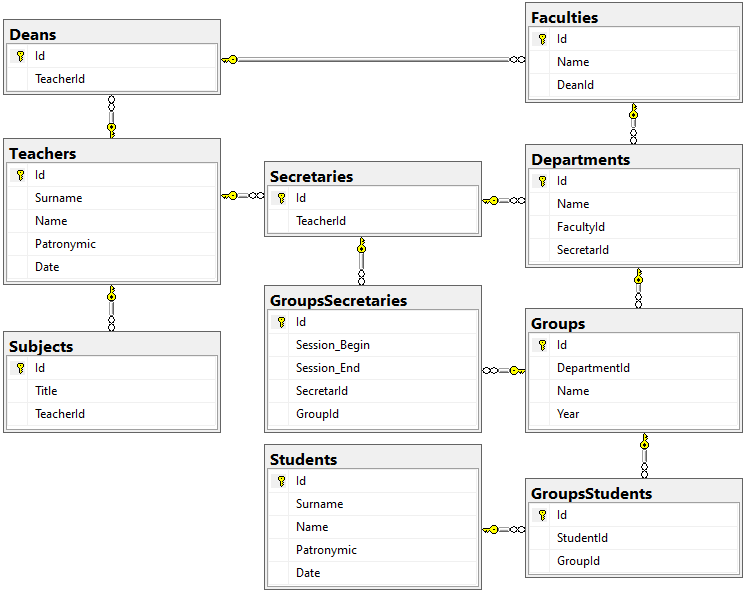


Рисунок 2.1 – Схема розробленої бази даних

Маючи розроблену схему бази даних, ми можемо повністю описати обмеження посилальної цілісності.

Відношення Teachers – Subjects має таке обмеження посилальної цілісності: ON DELETE: RESTRICT; ON UPDATE: CASCADE.

Відношення Teachers – Curators має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Teachers – Deans має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Teachers – Heads має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Departments – Heads має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Faculties – Deans має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Faculties – Departments має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Departments – Groups має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Groups – GroupsCurators має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Curators – GroupsCurators має таке обмеження посилальної цілісності: ON DELETE: RESTRICT; ON UPDATE: CASCADE.

Відношення Groups – GroupsStudents має таке обмеження посилальної цілісності: ON DELETE: CASCADE; ON UPDATE: CASCADE.

Відношення Students – GroupsStudents має таке обмеження посилальної цілісності: ON DELETE: RESTRICT; ON UPDATE: CASCADE.

Отже, нами було розроблено базу даних, а також зв’язки між таблицями та відповідні обмеження посилальної цілісності.

2.2 Проектування інтерфейсу програми

Графічний інтерфейс користувача — тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

Основні вимоги до розробленого інтерфейсу:

* наявність контекстного меню;
* продумана зміна фокусу.

Юзабіліті – це властивість продукту бути придатним до використання. Та визначає загальну степінь зручності предмета при використанні.

Згідно з класифікацією фірми IBM розрізняють такі типи користувачів:

* системний програміст;
* програміст;
* кінцевий користувач.

Кінцеві користувачі – це люди, які хочуть більш ефективно вирішувати свої завдання, використовуючи комп’ютерні засоби. Частіше всього вони не є знатоками в області інформатики.

В програмі «Academy.exe» розроблено 4 форми:

* для головного вікна;
* для основного вікна;
* для вікна пошуку даних;
* для вікна додаткових запитів.

Розглянемо спроектовану форму для головного вікна, яка зображена на рисунку 2.2.

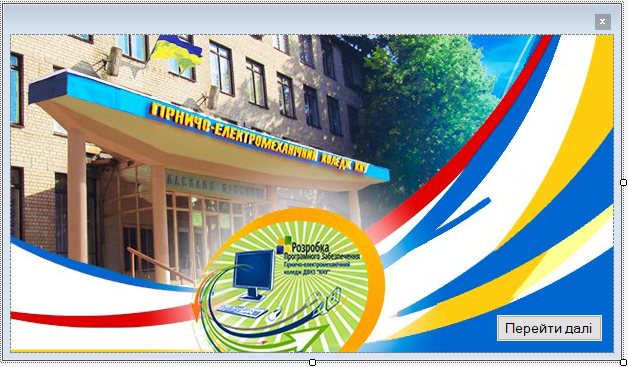


Рисунок 2.2 – Форма головного вікна

Розглянуте вище вікно призначене для подальшого переходу до програми. Кнопка «Перейти далі» призначена для переходу до вікна перегляду таблиць з даними.

Розглянемо спроектовану форму для основного вікна, яке зображене на рисунку 2.3.

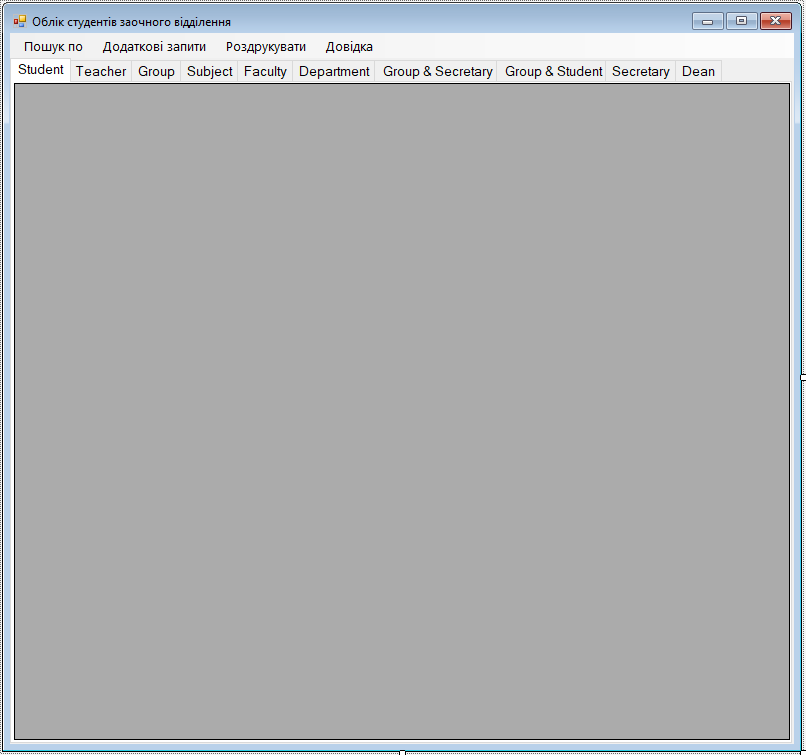


Рисунок 2.3 – Форма основного вікна програми

Це вікно призначене для виведення всієї інформації про навчальний заклад. Виведення інформації відбувається шляхом виконання запитів до БД, які ми розглянемо у пункті 2.3.

У меню можна виконати запити, які пов’язані з інформацією – вивести дані, запустити вікно пошуку даних, запустити вікно додаткових запитів, роздрукувати список студентів, викладачів та предметів, запустити довідку або остаточно вийти з програми.

Розглянемо спроектовану форму для вікна пошуку даних, яка зображена на рисунку 2.4.

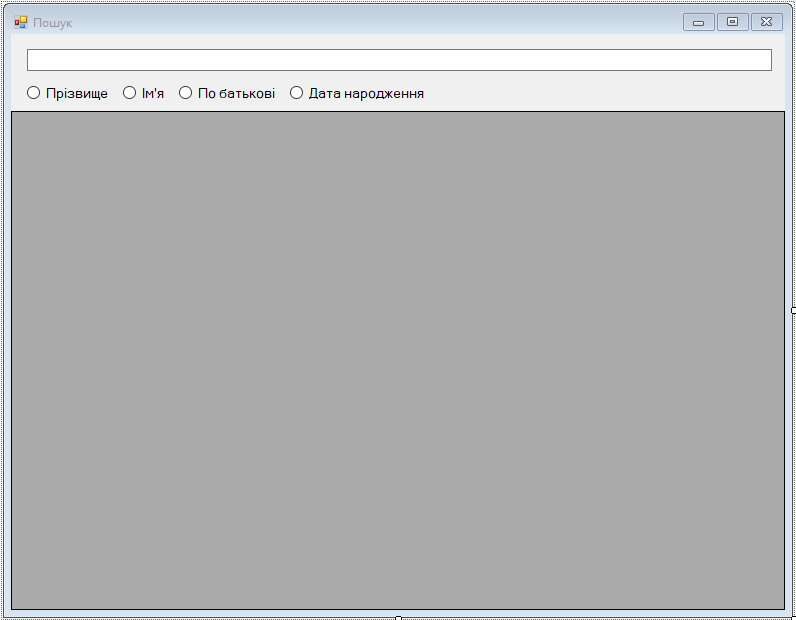


Рисунок 2.4 – Форма вікна для пошуку даних

Це вікно призначене для пошуку даних, шляхом виконання запиту до БД, який буде розглянуто у пункті 2.3.

У текстове поле можна ввести частину інформації, після чого буде виконано пошук за вибраним критерієм.

Розглянемо спроектовану форму для додаткових запитів, яка зображена на рисунку 2.4.

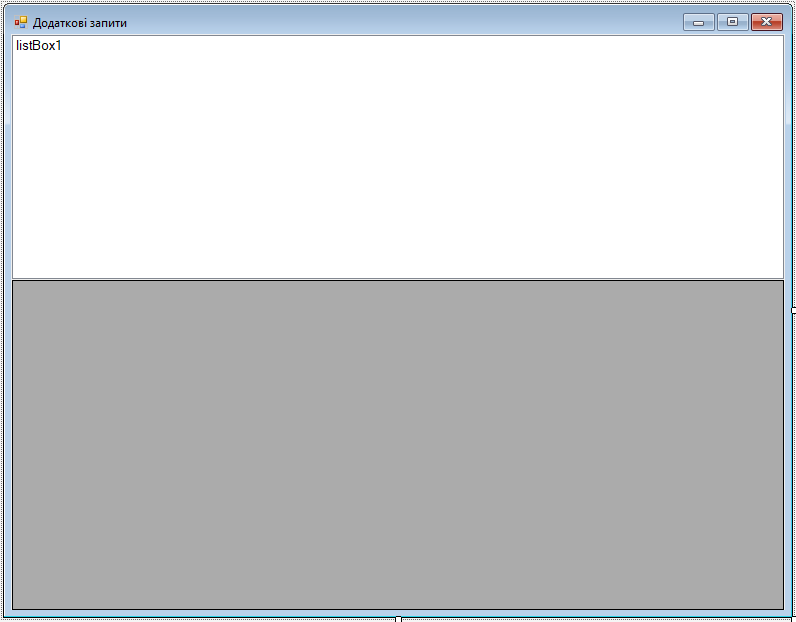


Рисунок 2.4 – Форма вікна для додаткових запитів

Це вікно призначене для виведення даних за допомогою додаткових запитів.

При натисканні запиту, які відображаються в ListBox, здійснюється вибірка даних.

Розглянемо спроектовану форму для довідки, яка зображена на рисунку 2.5.

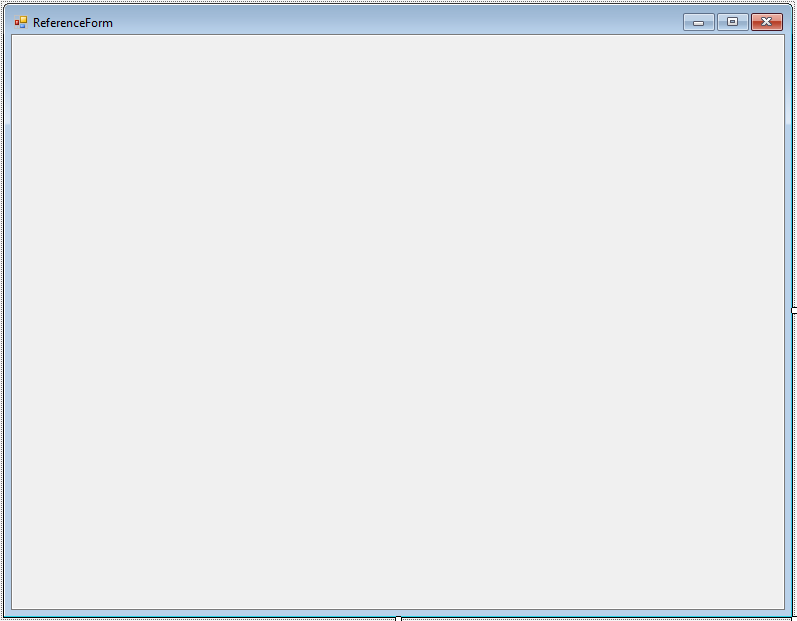


Рисунок 2.5 – Форма вікна для вставки даних

Це вікно призначене для виведення інструкції користувача, для подальшого користування програмою.

Отже, нами було розглянуто спроектований інтерфейс демонстраційної програми та її функціонал.

2.3 Програмна реалізація

Спроектувавши інтерфейс демонстраційної програми нами було вирішено, що необхідно створити запити на вибірку (2 простих, 3 складних, 2 вкладених), оновлення, видалення, вставку даних, а також для створення нової таблиці, тому, далі ми розглянемо кожен з запитів, відповідно до тих, які було розглянуто у предметній області.

Розглянемо простий запит на вибірку даних з таблиці «about\_me»:

string[] tables = { "Students", "Teachers", "Groups", "Subjects", "Faculties", "Departments", "GroupsSecretaries", "GroupsStudents", "Secretaries", "Deans" };

da = new SqlDataAdapter($"select \* from {tables[i]}", conn);

Запит призначений для отримання даних з різних таблиць. Після їх отримання, кожне значення додається до таблиці, розташованій на формі. Приклад виведення даних на форму наведено у наступному блоці команд:

ds = new DataSet(); for (int i = 0; i < tabControl1.TabCount; i++) {

da = new SqlDataAdapter($"select \* from {tables[i]}", conn);

table = new DataTable(); da.Fill(table); switch (i) {

case 0: dataGridView1.DataSource = table; dataGridView1. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 1: dataGridView2.DataSource = table; dataGridView2. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 2: dataGridView3.DataSource = table; dataGridView3. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 3: dataGridView4.DataSource = table; dataGridView4. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 4: dataGridView5.DataSource = table; dataGridView5. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 5: dataGridView6.DataSource = table; dataGridView6. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 6: dataGridView7.DataSource = table; dataGridView7. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 7: dataGridView8.DataSource = table; dataGridView8. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 8: dataGridView9.DataSource = table; dataGridView9. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 9: dataGridView10.DataSource = table; dataGridView10. AutoSizeColumns Mode = DataGridViewAutoSizeColumnsMode.AllCells; break;

default: break; } da.Fill(ds, tables[i]); }

Розглянемо складний запит на вибірку даних з таблиці «Departments» та «Groups»:

select Departments.[Name] as 'Кафедра', Groups.[Name] as 'Група' from Departments join Groups on Departments.Id = Groups.DepartmentId

Цей запит служить для відображення назв кафедр та груп, які їм належать..

Розглянемо складний запит на вибірку даних з таблиці «Subjects» та «Teachers»:

select Title as 'Назва предмету' from Subjects join Teachers on Subjects.TeacherId = Teachers.Id where Teachers.[Name]='Инга' and Teachers.Surname='Шалаганова'

Цей запит призначений для отримання дисциплін, які викладає вчитель «Шалаганова Инга»

Розглянемо запит на вибірку даних з різних таблиць:

da = new SqlDataAdapter($"select \* from [{strValue}] where [{str}] like '%{textBox1.Text}%'", conn);

Цей запит використовується для пошуку даних, по фрагменту рядка. Вибірка відбувається по вибраному критерію.

Розглянемо запити на вставку даних до таблиці «Students»:

Students students = new Students {

Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].

Value.ToString(), Name = dataGridView1[2, dataGridView1.CurrentRow.Index]. Value.ToString(), Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index]. Value.ToString(), Date = DateTime.Parse(dataGridView1[4, dataGridView1. CurrentRow.Index].Value.ToString()) }; db.Students.AddOrUpdate(students);

Розглянемо запит на створення нової таблиці «stud\_group»:

IF NOT EXISTS (SELECT \* FROM [sysobjects] WHERE [name] = 'stud\_rez' and [xtype]='U') CREATE TABLE [dbo].[Students]( [Id] [int] IDENTITY(1,1) NOT NULL, [Surname] [nvarchar](max) NOT NULL, [Name] [nvarchar](max) NOT NULL, [Patronymic] [nvarchar](max) NOT NULL, [Date] [date] NOT NULL, PRIMARY KEY CLUSTERED ([Id] ASC) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

Запит призначено для демонстрації створення таблиці. Створена, після цього запиту, таблиця повністю дублюватиме таблицю «stud\_group», але дані імпортовано у новостворену таблицю не буде.

Розглянемо запит на оновлення даних у таблиці «students»:

Students students = db.Students.Find(int.Parse(dataGridView1[0, dataGridView1. CurrentRow.Index].Value.ToString()));

if (students == null) return;

students.Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value. ToString(); students.Name = dataGridView1[2, dataGridView1.CurrentRow.Index] .Value.ToString(); students.Patronymic = dataGridView1[3, dataGridView1.CurrentRow. Index].Value.ToString(); students.Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString());

db.Students.AddOrUpdate(students);

Розглянемо запит на видалення даних з таблиці «students»:

Students students = db.Students.Find(int.Parse(dataGridView1[0, dataGridView1. CurrentRow.Index].Value.ToString()));

if (students == null) return;

students.Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value. ToString(); students.Name = dataGridView1[2, dataGridView1.CurrentRow.Index]. Value.ToString(); students.Patronymic = dataGridView1[3, dataGridView1.CurrentRow. Index].Value.ToString(); students.Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString());

db.Students.Remove(students);

Запит призначений для видалення даних по обраному у таблиці ідентифікатору події.

Отже, нами було розглянуто основні запити до бази даних та певні блоки команд з демонстраційної програми.

2.4 Інструкція користувача

Засобами демонстраційної програми надається можливість розглянути основні об’єкти бази даних, та способи маніпулювання з ними.

Для повноцінної роботи програми нам знадобиться SQL Server Management Studio, для підключення бази даних до серверу.

Перед запуском демонстраційної програми необхідно розпакувати архівний файл “Academy.rar” у будь яке місце на локальному диску вашого компьютера.

Архів містить папку з однойменною назвою. Дана папка містить наступні файли:

Academy.exe – виконуваний файл демонстраційного програмного додатку;

Connection.txt – текстовий файл з назвою вашого сервера;

Academy.mdf та Academy \_log.ldf – файли розробленої бази даних;

Academy.exe.config та Academy.pdb – конфігураційні файли програмного додатку.

Після розпакування архіву, необхідно відкрити текстовий файл Connection.txt та ввести у нього назву вашого екземпляру SQL Server. Після внесення змін до файлу необхідно підключити базу даних Academy до SQL Server’у.

Отже, нами було розроблено інструкцію користувача, для налаштування та роботи з базою даних і демонстраційним програмним додатком.

Загальні висновки

Завданням курсового проекту є розробка і реалізація бази даних для обліку студентів заочного відділення.

Користувачами розробленої бази даних можуть стати співробітники навчального закладу. Вони повинні підключити дану БД до свого локального серверу щоб використовувати та маніпулювати її даними.

Для створення бази даних та демонстраційного програмного додатку, нами було використано середовище для створення баз даних SQL Server Management Studio та середовище програмування Visual Studio C#, які мають весь необхідний набір інструментів та весь необхідний функціонал.

Нами було розроблено концептуальну схему бази даних, згідно стандарту IDEF1X, та алгоритми вирішення прикладної задачі.

При проектуванні інтерфейсу демонстраційного програмного додатку враховувались наступні вимоги:

* сумісність з потребами та можливостями користувача;
* забезпечення простоти переходу між різними формами;
* забезпечення користувача на високому рівні вказівками стосовно його можливих дій;
* особливості користувачів різних рівнів.

При розробці інтерфейсу програми, нами було створено 4 форми.

Розроблена база даних містить наступні сутності та відповідні атрибути:

1. Інформація про студентів (ідентифікатор, прізвище, ім’я, по батькові, дата народження).
2. Інформація про викладачів (ідентифікатор, прізвище, ім’я, по батькові, дата народження).
3. Групи (ідентифікатор, ідентифікатор кафедри, назва, курс).
4. Предмети (ідентифікатор, назва, ідентифікатор викладача).
5. Факультети (ідентифікатор, назва, ідентифікатор декана).
6. Кафедри (ідентифікатор, назва, ідентифікатор факультету, ідентифікатор секретаря).
7. Групи і Секретаря (ідентифікатор, початок сесії, кінець сесії, ідентифікатор секретаря, ідентифікатор групи).
8. Групи і Студента(ідентифікатор, ідентифікатор секретаря, ідентифікатор групи).
9. Секретарів (ідентифікатор, ідентифікатор викладача).
10. Деканів (ідентифікатор, ідентифікатор викладача).

База даних була нормалізована відповідно до трьох нормальних форм.

Для демонстрації бази даних ділового записника викладача, було розроблено програмний додаток «Academy», який надає можливість роботи з даними БД, має зручний графічний інтерфейс для додавання, редагування та перегляду даних.

Отже, розроблена база даних для обліку студентів заочного відділення, може бути використана співробітниками навчального закладу для автоматизації роботи.

Список використаної літератури

1. Алоева В.А., Бейсова В.Е. Настольная книга классного руководителя. – Ростов: Феникс, 2004. – 320 с.

2. Анісімов А.В., Кулябко П.П. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. – Київ, 2017. – 110 с.

3. http://www.kgemt.org.ua

4. Маклафин Б. PHP и MySQL. Исчерпывающее руководство. — СПб.: Питер, 2013. — 512 с.

5. Хеник Б. HTML и CSS: путь к совершенству. - СПб.: Питер, 2011. – 336 с.

6. Герберт Шилдт C# 4.0. Полное руководство. – М.: Издательський дом «Вильямс», 2015. – 1056 с.

7. Бідюк П. І., Гожий О. П., Коршевнюк Л. О. Комп’ютерні системи підтримки прийняття рішень. – Київ, 2010. – 336 с.

8. Петух А.М., Майданюк В.П. Навчальний посібник "Інтерфейс користувач-комп’ютер". – Вінниця: ВДТУ, 1999. – 65 с.

9. Довгялло А. М., Брановицький В. И., Вершинін К. П. Диалоговые системы. Современное состояние и перспективы развития. – Київ: Наук. думка, 1987. – 248 с.

Додаток А

Додаток Б

Код програми

Файл MainForm.cs:

using System; using System.Data; using System.Data.Entity.Migrations;

using System.Data.SqlClient; using System.IO; using System.Reflection;

using System.Text; using System.Windows.Forms;

using Word = Microsoft.Office.Interop.Word;

namespace Academy {

public partial class MainForm : Form {

string[] tables = { "Students", "Teachers", "Groups", "Subjects", "Faculties", "Departments", "GroupsSecretaries", "GroupsStudents", "Secretaries", "Deans" };

AcademyEntities db = new AcademyEntities();

string[] ConnectionString = null; //рядок підключення

SqlConnection conn = null; //об'єк підключення

DataSet ds = null;

SqlDataAdapter da = null;

DataTable table = null;

public MainForm() { InitializeComponent(); //ініціалізація компонентів

try { //отримання назви серверу з файлу

ConnectionString = File.ReadAllLines("Connection.txt", Encoding.Default);

conn = new SqlConnection($@"Data Source={ConnectionString[0]}; Initial Catalog=Academy; Integrated Security=SSPI;"); OutputBase();

for (int i = 0; i < tabControl1.TabCount; i++) { switch (i) {

case 0: dataGridView1.ContextMenuStrip = contextMenuStrip1; break; //студент

case 1: dataGridView2.ContextMenuStrip = contextMenuStrip1; break; //викладач

case 2: dataGridView3.ContextMenuStrip = contextMenuStrip1; break; //група

case 3: dataGridView4.ContextMenuStrip = contextMenuStrip1; break; //предмет

case 4: dataGridView5.ContextMenuStrip = contextMenuStrip1; break; //факультет

case 5: dataGridView6.ContextMenuStrip = contextMenuStrip1; break; //кафедра

case 6: dataGridView7.ContextMenuStrip = contextMenuStrip1; break; //група і секретар

case 7: dataGridView8.ContextMenuStrip = contextMenuStrip1; break; //група і студент

case 8: dataGridView9.ContextMenuStrip = contextMenuStrip1; break; //секретар

case 9: dataGridView10.ContextMenuStrip = contextMenuStrip1; break; //декан

default: break; } } }

catch (Exception ex) {MessageBox.Show(ex.Message); //при винекненні вийтятку } }

private void OutputBase() {

try { ds = new DataSet(); for (int i = 0; i < tabControl1.TabCount; i++) {

da = new SqlDataAdapter($"select \* from {tables[i]}", conn); //підготовка команди

table = new DataTable(); da.Fill(table); //об'єкт для вміщення результатів вибірки

switch (i) {

case 0: dataGridView1.DataSource = table; dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 1: dataGridView2.DataSource = table; dataGridView2.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 2: dataGridView3.DataSource = table; dataGridView3.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 3: dataGridView4.DataSource = table; dataGridView4.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 4: dataGridView5.DataSource = table; dataGridView5.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 5: dataGridView6.DataSource = table; dataGridView6.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 6: dataGridView7.DataSource = table; dataGridView7.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 7: dataGridView8.DataSource = table; dataGridView8.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 8: dataGridView9.DataSource = table; dataGridView9.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

case 9: dataGridView10.DataSource = table; dataGridView10.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells; break;

default: break; } da.Fill(ds, tables[i]); } }//виконати запит на вибірку

catch (Exception ex) { MessageBox.Show(ex.Message); } }//при винекненні вийтятку

//перехід до вікна пошуку по студентам

private void студентамToolStripMenuItem\_Click(object sender, EventArgs e) {

SearchForm form = new SearchForm("Students"); form.ShowDialog(); }

//перехід до вікна пошуку по викладачам

private void викладачамToolStripMenuItem\_Click(object sender, EventArgs e) {

SearchForm form = new SearchForm("Teachers"); form.ShowDialog(); }

//перехід до вікна пошуку по групам

private void групамToolStripMenuItem\_Click(object sender, EventArgs e) {

SearchForm form = new SearchForm("Groups"); form.ShowDialog(); }

//перехід до вікна пошуку по предметам

private void предметамToolStripMenuItem\_Click(object sender, EventArgs e) {

SearchForm form = new SearchForm("Subjects"); form.ShowDialog();}

//перехід до вікна пошуку по факультетам

private void факультетамToolStripMenuItem\_Click(object sender, EventArgs e) {

SearchForm form = new SearchForm("Faculties"); form.ShowDialog(); }

//перехід до вікна пошуку по кафедрам

private void кафедрамToolStripMenuItem\_Click(object sender, EventArgs e) {

SearchForm form = new SearchForm("Departments"); form.ShowDialog(); }

private void Refreshed(int ad) { switch (ad) { //обновлення таблиць

case 0: dataGridView1.Refresh(); break;

case 1: dataGridView2.Refresh(); break;

case 2: dataGridView3.Refresh(); break;

case 3: dataGridView4.Refresh(); break;

case 4: dataGridView5.Refresh(); break;

case 5: dataGridView6.Refresh(); break;

case 6: dataGridView7.Refresh(); break;

case 7: dataGridView8.Refresh(); break;

case 8: dataGridView9.Refresh(); break;

case 9: dataGridView10.Refresh(); break;

default: break; } }

private void додатиЗаписToolStripMenuItem\_Click(object sender, EventArgs e) {

try { int ad = tabControl1.SelectedIndex;

switch (ad) {

case 0: Students students = new Students {

Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString(),

Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString(),

Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString(),

Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString()) };

db.Students.AddOrUpdate(students); break;

case 1: Teachers teachers = new Teachers {

Surname = dataGridView2[1, dataGridView2.CurrentRow.Index].Value.ToString(),

Name = dataGridView2[2, dataGridView2.CurrentRow.Index].Value.ToString(),

Patronymic = dataGridView2[3, dataGridView2.CurrentRow.Index].Value.ToString(),

Date = DateTime.Parse(dataGridView2[4, dataGridView2.CurrentRow.Index].Value.ToString()) };

db.Teachers.AddOrUpdate(teachers); break;

case 2: Groups groups = new Groups {

DepartmentId = int.Parse(dataGridView3[1, dataGridView3.CurrentRow.Index].Value

.ToString()),

Name = dataGridView3[2, dataGridView3.CurrentRow.Index].Value.ToString(),

Year = int.Parse(dataGridView3[3, dataGridView3.CurrentRow.Index].Value.ToString()) };

db.Groups.AddOrUpdate(groups); break;

case 3: Subjects subjects = new Subjects {

Title = dataGridView4[1, dataGridView4.CurrentRow.Index].Value.ToString(),

TeacherId = int.Parse(dataGridView4[2,dataGridView4.CurrentRow.Index].Value.ToString())};

db.Subjects.AddOrUpdate(subjects); break;

case 4: Faculties faculties = new Faculties {

Name = dataGridView5[1, dataGridView5.CurrentRow.Index].Value.ToString(),

DeanId = int.Parse(dataGridView5[2, dataGridView5.CurrentRow.Index].Value.ToString())};

db.Faculties.AddOrUpdate(faculties); break;

case 5:Departments departments = new Departments {

Name = dataGridView6[1, dataGridView6.CurrentRow.Index].Value.ToString(),

FacultyId = int.Parse(dataGridView6[2, dataGridView6.CurrentRow.Index].Value.ToString()),

SecretarId =int.Parse(dataGridView6[3,dataGridView6.CurrentRow.Index].Value.ToString())};

db.Departments.AddOrUpdate(departments); break;

case 6:GroupsSecretaries groupsSecretaries = new GroupsSecretaries {

Session\_Begin = DateTime.Parse(dataGridView7[1, dataGridView7.CurrentRow.Index].Value. ToString()), Session\_End = DateTime.Parse(dataGridView7[2, dataGridView7.CurrentRow. Index].Value.ToString()), SecretarId = int.Parse(dataGridView7[3,dataGridView7.CurrentRow. Index].Value.ToString()), GroupId = int.Parse(dataGridView7[4, dataGridView7. CurrentRow.Index].Value.ToString()) };

db.GroupsSecretaries.AddOrUpdate(groupsSecretaries); break;

case 7:GroupsStudents groupsStudents = new GroupsStudents {

StudentId = int.Parse(dataGridView8[1, dataGridView8.CurrentRow.Index].Value.ToString()),

GroupId = int.Parse(dataGridView8[2, dataGridView8.CurrentRow.Index].Value.ToString())};

db.GroupsStudents.AddOrUpdate(groupsStudents); break;

case 8:Secretaries secretaries = new Secretaries {

TeacherId = int.Parse(dataGridView9[1,dataGridView9.CurrentRow.Index].Value.ToString())};

db.Secretaries.AddOrUpdate(secretaries); break;

case 9:Deans deans = new Deans { TeacherId = int.Parse(dataGridView10[1, dataGridView10.CurrentRow.Index].Value.ToString())};

db.Deans.AddOrUpdate(deans); break;

default: break; }db.SaveChanges();Refreshed(ad);}

catch (Exception ex) {MessageBox.Show(ex.Message);}}

private void редагуватиЗаписToolStripMenuItem\_Click(object sender, EventArgs e) {

try {int ad = tabControl1.SelectedIndex;switch (ad) {

case 0:Students students = db.Students.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

if (students == null) return;

students.Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

students.Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString();

students.Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString();

students.Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString());

db.Students.AddOrUpdate(students); break;

case 1:Teachers teachers = db.Teachers.Find(int.Parse(dataGridView2[0, dataGridView2.CurrentRow.Index].Value.ToString()));

teachers.Surname = dataGridView2[1, dataGridView2.CurrentRow.Index].Value.ToString();

teachers.Name = dataGridView2[2, dataGridView2.CurrentRow.Index].Value.ToString();

teachers.Patronymic = dataGridView2[3, dataGridView2.CurrentRow.Index].Value.ToString();

teachers.Date = DateTime.Parse(dataGridView2[4, dataGridView2.CurrentRow.Index].Value.ToString());

db.Teachers.AddOrUpdate(teachers); break;

case 2:Groups groups = db.Groups.Find(int.Parse(dataGridView3[0, dataGridView3.CurrentRow.Index].Value.ToString()));

groups.DepartmentId = int.Parse(dataGridView3[1, dataGridView3.CurrentRow.Index].Value.ToString());

groups.Name = dataGridView3[2, dataGridView3.CurrentRow.Index].Value.ToString();

groups.Year = int.Parse(dataGridView3[3, dataGridView3.CurrentRow.Index].Value.ToString());

db.Groups.AddOrUpdate(groups); break;

case 3:Subjects subjects = db.Subjects.Find(int.Parse(dataGridView4[0, dataGridView4.CurrentRow.Index].Value.ToString()));

subjects.Title = dataGridView4[1, dataGridView4.CurrentRow.Index].Value.ToString();

subjects.TeacherId = int.Parse(dataGridView4[2, dataGridView4.CurrentRow.Index].Value.ToString());

db.Subjects.AddOrUpdate(subjects); break;

case 4:Faculties faculties = db.Faculties.Find(int.Parse(dataGridView5[0, dataGridView5.CurrentRow.Index].Value.ToString()));

faculties.Name = dataGridView5[1, dataGridView5.CurrentRow.Index].Value.ToString();

faculties.DeanId = int.Parse(dataGridView5[2, dataGridView5.CurrentRow.Index].Value.ToString());

db.Faculties.AddOrUpdate(faculties); break;

case 5:Departments departments = db.Departments.Find(int.Parse(dataGridView6[0,

dataGridView6.CurrentRow.Index].Value.ToString()));

departments.Name = dataGridView6[1, dataGridView6.CurrentRow.Index].Value.ToString();

departments.FacultyId = int.Parse(dataGridView6[2, dataGridView6.CurrentRow.Index].Value.ToString());

departments.SecretarId = int.Parse(dataGridView6[3, dataGridView6.CurrentRow.Index].Value.ToString());

db.Departments.AddOrUpdate(departments); break;

case 6:GroupsSecretaries groupsSecretaries = db.GroupsSecretaries. Find(int.Parse(dataGridView7[0, dataGridView7.CurrentRow.Index].Value.ToString()));

groupsSecretaries.Session\_Begin = DateTime.Parse(dataGridView7[1, dataGridView7.CurrentRow.Index].Value.ToString());

groupsSecretaries.Session\_End = DateTime.Parse(dataGridView7[2, dataGridView7.CurrentRow.Index].Value.ToString());

groupsSecretaries.SecretarId = int.Parse(dataGridView7[3, dataGridView7.CurrentRow.Index].Value.ToString());

groupsSecretaries.GroupId = int.Parse(dataGridView7[4, dataGridView7.CurrentRow.Index].Value.ToString());

db.GroupsSecretaries.AddOrUpdate(groupsSecretaries); break;

case 7:GroupsStudents groupsStudents = db.GroupsStudents.Find(int.Parse(dataGridView8[0, dataGridView8.CurrentRow.Index].Value.ToString()));

groupsStudents.StudentId = int.Parse(dataGridView8[1, dataGridView8.CurrentRow.Index].Value.ToString());

groupsStudents.GroupId = int.Parse(dataGridView8[2, dataGridView8.CurrentRow.Index].Value.ToString());

db.GroupsStudents.AddOrUpdate(groupsStudents); break;

case 8:Secretaries secretaries = db.Secretaries.Find(int.Parse(dataGridView9[0, dataGridView9.CurrentRow.Index].Value.ToString()));

secretaries.TeacherId = int.Parse(dataGridView9[1, dataGridView9.CurrentRow.Index].Value.ToString());

db.Secretaries.AddOrUpdate(secretaries); break;

case 9:Deans deans = db.Deans.Find(int.Parse(dataGridView10[0, dataGridView10.CurrentRow.Index].Value.ToString()));

deans.TeacherId = int.Parse(dataGridView10[1, dataGridView10.CurrentRow.Index].Value.ToString());

db.Deans.AddOrUpdate(deans); break;

default: break;}

db.SaveChanges();Refreshed(ad);}

catch (Exception ex) {MessageBox.Show(ex.Message);}}

private void видалитиЗаписToolStripMenuItem\_Click(object sender, EventArgs e) {

try {int ad = tabControl1.SelectedIndex;switch (ad) {

case 0:Students students = db.Students.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

if (students == null) return;

students.Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

students.Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString();

students.Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString();

students.Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString());

db.Students.Remove(students); break;

case 1:Teachers teachers = db.Teachers.Find(int.Parse(dataGridView2[0, dataGridView2.CurrentRow.Index].Value.ToString()));

teachers.Surname = dataGridView2[1, dataGridView2.CurrentRow.Index].Value.ToString();

teachers.Name = dataGridView2[2, dataGridView2.CurrentRow.Index].Value.ToString();

teachers.Patronymic = dataGridView2[3, dataGridView2.CurrentRow.Index].Value.ToString();

teachers.Date = DateTime.Parse(dataGridView2[4, dataGridView2.CurrentRow.Index].Value.ToString());

db.Teachers.Remove(teachers); break;

case 2:Groups groups = db.Groups.Find(int.Parse(dataGridView3[0, dataGridView3.CurrentRow.Index].Value.ToString()));

groups.DepartmentId = int.Parse(dataGridView3[1, dataGridView3.CurrentRow.Index].Value.ToString());

groups.Name = dataGridView3[2, dataGridView3.CurrentRow.Index].Value.ToString();

groups.Year = int.Parse(dataGridView3[3, dataGridView3.CurrentRow.Index].Value.ToString());

db.Groups.Remove(groups); break;

case 3:Subjects subjects = db.Subjects.Find(int.Parse(dataGridView4[0, dataGridView4.CurrentRow.Index].Value.ToString()));

subjects.Title = dataGridView4[1, dataGridView4.CurrentRow.Index].Value.ToString();

subjects.TeacherId = int.Parse(dataGridView4[2, dataGridView4.CurrentRow.Index].Value.ToString());

db.Subjects.Remove(subjects); break;

case 4:Faculties faculties = db.Faculties.Find(int.Parse(dataGridView5[0, dataGridView5.CurrentRow.Index].Value.ToString()));

faculties.Name = dataGridView5[1, dataGridView5.CurrentRow.Index].Value.ToString();

faculties.DeanId = int.Parse(dataGridView5[2, dataGridView5.CurrentRow.Index].Value.ToString());

db.Faculties.Remove(faculties); break;

case 5:Departments departments = db.Departments.Find(int.Parse(dataGridView6[0, dataGridView6.CurrentRow.Index].Value.ToString()));

departments.Name = dataGridView6[1, dataGridView6.CurrentRow.Index].Value.ToString();

departments.FacultyId = int.Parse(dataGridView6[2, dataGridView6.CurrentRow.Index].Value.ToString());

departments.SecretarId = int.Parse(dataGridView6[3, dataGridView6.CurrentRow.Index].Value.ToString());

db.Departments.Remove(departments); break;

case 6:GroupsSecretaries groupsSecretaries = db.GroupsSecretaries.Find(int.Parse(dataGridView7[0, dataGridView7.CurrentRow.Index].Value.ToString()));

groupsSecretaries.Session\_Begin = DateTime.Parse(dataGridView7[1, dataGridView7.CurrentRow.Index].Value.ToString());

groupsSecretaries.Session\_End = DateTime.Parse(dataGridView7[2, dataGridView7.CurrentRow.Index].Value.ToString());

groupsSecretaries.SecretarId = int.Parse(dataGridView7[3, dataGridView7.CurrentRow.Index].Value.ToString());

groupsSecretaries.GroupId = int.Parse(dataGridView7[4, dataGridView7.CurrentRow.Index].Value.ToString());

db.GroupsSecretaries.Remove(groupsSecretaries); break;

case 7:GroupsStudents groupsStudents = db.GroupsStudents.Find(int.Parse(dataGridView8[0, dataGridView8.CurrentRow.Index].Value.ToString()));

groupsStudents.StudentId = int.Parse(dataGridView8[1, dataGridView8.CurrentRow.Index].Value.ToString());

groupsStudents.GroupId = int.Parse(dataGridView8[2, dataGridView8.CurrentRow.Index].Value.ToString());

db.GroupsStudents.Remove(groupsStudents); break;

case 8:Secretaries secretaries = db.Secretaries.Find(int.Parse(dataGridView9[0, dataGridView9.CurrentRow.Index].Value.ToString()));

secretaries.TeacherId = int.Parse(dataGridView9[1, dataGridView9.CurrentRow.Index].Value.ToString());

db.Secretaries.Remove(secretaries); break;

case 9:Deans deans = db.Deans.Find(int.Parse(dataGridView10[0, dataGridView10.CurrentRow.Index].Value.ToString()));

deans.TeacherId = int.Parse(dataGridView10[1, dataGridView10.CurrentRow.Index].Value.ToString());

db.Deans.Remove(deans); break;

default: break;}

db.SaveChanges();Refreshed(ad);}

catch (Exception ex) {MessageBox.Show(ex.Message);}}

private void MainForm\_Activated(object sender, EventArgs e) {OutputBase();}

private void Export\_Data\_To\_Word(DataGridView DGV, string fileName) {

if (DGV.Rows.Count != 0) {int RowCount = DGV.Rows.Count;

int ColumnCount = DGV.Columns.Count;

Object[,] DataArray = new object[RowCount + 1, ColumnCount + 1];

//add rows

int r = 0;for (int c = 0; c <= ColumnCount - 1; c++) {

for (r = 0; r <= RowCount - 1; r++) {DataArray[r, c] = DGV.Rows[r].Cells[c].Value;}}

Word.Document oDoc = new Word.Document();

oDoc.Application.Visible = true;

//page orintation

oDoc.PageSetup.Orientation = Word.WdOrientation.wdOrientLandscape;

dynamic oRange = oDoc.Content.Application.Selection.Range;

string oTemp = "";

for (r = 0; r <= RowCount - 1; r++) {

for (int c = 0; c <= ColumnCount - 1; c++) {

oTemp = oTemp + DataArray[r, c] + "\t";}}

//table format

oRange.Text = oTemp;

object oMissing = Missing.Value;

object Separator = Word.WdTableFieldSeparator.wdSeparateByTabs;

object ApplyBorders = true;object AutoFit = true;

object AutoFitBehavior = Word.WdAutoFitBehavior.wdAutoFitContent;

oRange.ConvertToTable(ref Separator, ref RowCount, ref ColumnCount,

Type.Missing, Type.Missing, ref ApplyBorders,

Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing,

Type.Missing, ref AutoFit, ref AutoFitBehavior, Type.Missing);

oRange.Select();

oDoc.Application.Selection.Tables[1].Select();

oDoc.Application.Selection.Tables[1].Rows.AllowBreakAcrossPages = 0;

oDoc.Application.Selection.Tables[1].Rows.Alignment = 0;

oDoc.Application.Selection.Tables[1].Rows[1].Select();

oDoc.Application.Selection.InsertRowsAbove(1);

oDoc.Application.Selection.Tables[1].Rows[1].Select();

//header row style

oDoc.Application.Selection.Tables[1].Rows[1].Range.Bold = 1;

oDoc.Application.Selection.Tables[1].Rows[1].Range.Font.Name = "Tahoma";

oDoc.Application.Selection.Tables[1].Rows[1].Range.Font.Size = 14;

//add header row manually

for (int c = 0; c <= ColumnCount - 1; c++) {

oDoc.Application.Selection.Tables[1].Cell(1, c + 1).Range.Text = DGV.Columns[c].HeaderText;}

//table style

oDoc.Application.Selection.Tables[1].Rows[1].Select();

oDoc.Application.Selection.Cells.VerticalAlignment = Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

//header text

foreach (Word.Section section in oDoc.Application.ActiveDocument.Sections) {

Word.Range headerRange = section.Headers[Word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range;

headerRange.Fields.Add(headerRange, Word.WdFieldType.wdFieldPage);

headerRange.Text = "your header text";

headerRange.Font.Size = 16;

headerRange.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;}}}

private void студентівПоГрупамToolStripMenuItem\_Click(object sender, EventArgs e) {

Export\_Data\_To\_Word(dataGridView1, "List\_Students");}

private void списокВикладачівToolStripMenuItem\_Click(object sender, EventArgs e) {

Export\_Data\_To\_Word(dataGridView2, "List\_Teachers");}

private void довідкаToolStripMenuItem\_Click(object sender, EventArgs e)

private void списокПредметівToolStripMenuItem\_Click(object sender, EventArgs e) {

Export\_Data\_To\_Word(dataGridView4, "List\_Teachers");}

private void додатковіЗапитиToolStripMenuItem\_Click(object sender, EventArgs e) {

AdditionalRequestForm form = new AdditionalRequestForm();form.ShowDialog();}}}

Файл SearchForm.cs

using System; using System.Data; using System.Data.Entity.Migrations;

using System.Data.SqlClient; using System.IO; using System.Text;

using System.Windows.Forms;

namespace Academy {

public partial class SearchForm : Form {

string[] tables = { "Students", "Teachers", "Groups", "Subjects", "Faculties", "Departments" };

AcademyEntities db = new AcademyEntities();

public string[] ConnectionString;

public SqlConnection conn;

public SqlDataAdapter da;

public SqlCommandBuilder cmd;

public DataTable table;

public string strValue;

public SearchForm(string str) {

InitializeComponent();

try { ConnectionString = File.ReadAllLines("Connection.txt", Encoding.Default);

conn = new SqlConnection($@"Data Source={ConnectionString[0]}; Initial Catalog=Academy; Integrated Security=SSPI;");

strValue = str; OutputBase(); GenerateRadioButton();

dataGridView1.ContextMenuStrip = contextMenuStrip1; }

catch (Exception ex) { MessageBox.Show(ex.Message); } }

private void GenerateRadioButton() {swit ch (strValue) {

case "Students": case "Teachers": radioButton1.Text = "Прізвище"; radioButton2.Text = "Ім'я"; radioButton3.Text = "По батькові"; radioButton4.Text = "Дата народження"; break;

case "Groups": radioButton2.Location = new System.Drawing.Point(92, 51);

radioButton1.Text = "Назва"; radioButton2.Text = "Курс"; radioButton3.Visible = false; radioButton4.Visible = false; break;

case "Subjects":case "Faculties":case "Departments":radioButton1.Visible = false; radioButton2.Visible = false; radioButton3.Visible = false; radioButton4.Visible = false;

dataGridView1.Location = new System.Drawing.Point(0, 51);

dataGridView1.Size = new System.Drawing.Size(774, 525);break;default: break;}}

private void OutputBase() {

da = new SqlDataAdapter($"select \* from {strValue}", conn);

cmd = new SqlCommandBuilder(da);table = new DataTable();da.Fill(table);

dataGridView1.DataSource = table;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;}

private void textBox1\_TextChanged(object sender, EventArgs e) {

try {string str = "";switch (strValue) {

case "Students":case "Teachers":

if (radioButton1.Checked) str = "Surname";

if (radioButton2.Checked) str = "Name";

if (radioButton3.Checked) str = "Patronymic";

if (radioButton4.Checked) str = "Date";break;

case "Groups":

if (radioButton1.Checked) str = "Name";

if (radioButton2.Checked) str = "Year";break;

case "Subjects": case "Faculties": case "Departments": str = "Name"; break;default: break;}

da = new SqlDataAdapter($"select \* from [{strValue}] where [{str}] like '%{textBox1.Text}%'", conn);

cmd = new SqlCommandBuilder(da);table = new DataTable();da.Fill(table);

dataGridView1.DataSource = table;}

catch (Exception ex) {if (textBox1.Text != "") MessageBox.Show("Сначала выберите критерий поиска");}}

private void додатиЗаписToolStripMenuItem\_Click(object sender, EventArgs e) {

try {switch (strValue) {

case "Students":Students students = new Students {

Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString(),

Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString(),

Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString(),

Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString())};

db.Students.AddOrUpdate(students); break;

case "Teachers":Teachers teachers = new Teachers {

Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString(),

Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString(),

Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString(),

Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString())};

db.Teachers.AddOrUpdate(teachers); break;

case "Groups":Groups groups = new Groups {

DepartmentId = int.Parse(dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString()),

Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString(),

Year = int.Parse(dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString())};

db.Groups.AddOrUpdate(groups); break;

case "Subjects":Subjects subjects = new Subjects {

Title = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString(),

TeacherId = int.Parse(dataGridView1[2,dataGridView1.CurrentRow.Index].Value.ToString())};

db.Subjects.AddOrUpdate(subjects); break;

case "Faculties":Faculties faculties = new Faculties {

Name = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString(),

DeanId = int.Parse(dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString())};

db.Faculties.AddOrUpdate(faculties); break;

case "Departments":Departments departments = new Departments {

Name = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString(),

FacultyId = int.Parse(dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString()),

SecretarId =int.Parse(dataGridView1[3,dataGridView1.CurrentRow.Index].Value.ToString())};

db.Departments.AddOrUpdate(departments); break;

default: break;}

db.SaveChanges();dataGridView1.Refresh();}

catch (Exception ex) {MessageBox.Show(ex.Message);}}

private void редагуватиЗаписToolStripMenuItem\_Click(object sender, EventArgs e) {

try {switch (strValue) {

case "Students":Students students = db.Students.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

if (students == null) return;

students.Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

students.Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString();

students.Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString();

students.Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.

ToString());db.Students.AddOrUpdate(students); break;

case "Teachers":Teachers teachers = db.Teachers.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

teachers.Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

teachers.Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString();

teachers.Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString();

teachers.Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString());

db.Teachers.AddOrUpdate(teachers); break;

case "Groups":Groups groups = db.Groups.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

groups.DepartmentId = int.Parse(dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString());

groups.Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString();

groups.Year = int.Parse(dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString());

db.Groups.AddOrUpdate(groups); break;

case "Subjects":Subjects subjects = db.Subjects.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

subjects.Title = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

subjects.TeacherId = int.Parse(dataGridView1[2,

dataGridView1.CurrentRow.Index].Value.ToString());

db.Subjects.AddOrUpdate(subjects); break;

case "Faculties":Faculties faculties = db.Faculties.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

faculties.Name = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

faculties.DeanId = int.Parse(dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString());

db.Faculties.AddOrUpdate(faculties); break;

case "Departments":Departments departments=db.Departments.Find(int.Parse(dataGridView1

[0,dataGridView1.CurrentRow.Index].Value.ToString()));

departments.Name = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

departments.FacultyId = int.Parse(dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString());

departments.SecretarId = int.Parse(dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString());

db.Departments.AddOrUpdate(departments); break;

default: break;}

db.SaveChanges();dataGridView1.Refresh();}

catch (Exception ex) {MessageBox.Show(ex.Message);}}

private void видалитиЗаписToolStripMenuItem\_Click(object sender, EventArgs e) {

try {switch (strValue) {

case "Students":Students students = db.Students.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

if (students == null) return;

students.Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

students.Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString();

students.Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString();

students.Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString());

db.Students.Remove(students); break;

case "Teachers":Teachers teachers = db.Teachers.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

teachers.Surname = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

teachers.Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString();

teachers.Patronymic = dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString();

teachers.Date = DateTime.Parse(dataGridView1[4, dataGridView1.CurrentRow.Index].Value.ToString());

db.Teachers.Remove(teachers); break;

case "Groups":

Groups groups = db.Groups.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

groups.DepartmentId = int.Parse(dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString());

groups.Name = dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString();

groups.Year = int.Parse(dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString());

db.Groups.Remove(groups); break;

case "Subjects":Subjects subjects = db.Subjects.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

subjects.Title = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

subjects.TeacherId = int.Parse(dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString());

db.Subjects.Remove(subjects); break;

case "Faculties":Faculties faculties = db.Faculties.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

faculties.Name = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

faculties.DeanId = int.Parse(dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString());

db.Faculties.Remove(faculties); break;

case "Departments":Departments departments = db.Departments.Find(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()));

departments.Name = dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString();

departments.FacultyId = int.Parse(dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString());

departments.SecretarId = int.Parse(dataGridView1[3, dataGridView1.CurrentRow.Index].Value.ToString());

db.Departments.Remove(departments); break;

default: break;}db.SaveChanges();dataGridView1.Refresh();}

catch (Exception ex) {MessageBox.Show(ex.Message);}}}}

Файл AdditionalRequestForm.cs

using System; using System.Data;using System.Data.SqlClient;

using System.IO;using System.Text;using System.Windows.Forms;

namespace Academy {

public partial class AdditionalRequestForm : Form {

public string[] ConnectionString;

public SqlConnection conn;

public SqlDataAdapter da;

public SqlCommandBuilder cmd;

public DataTable table;

public string strQuery = "";

public AdditionalRequestForm() {

InitializeComponent();

try {listBox1.Text = "";

listBox1.Items.AddRange(File.ReadAllLines("Query.txt", Encoding.Default));

ConnectionString = File.ReadAllLines("Connection.txt", Encoding.Default);

conn = new SqlConnection($@"Data Source={ConnectionString[0]}; Initial Catalog=Academy; Integrated Security=SSPI;");}

catch (Exception ex) {MessageBox.Show(ex.Message);}}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e) {

try {int i = listBox1.SelectedIndex;OutputBase(i);

da = new SqlDataAdapter(strQuery, conn);

cmd = new SqlCommandBuilder(da);

table = new DataTable();da.Fill(table);

dataGridView1.DataSource = table;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;}

catch (Exception ex) {MessageBox.Show(ex.Message);}}

private void OutputBase(int i) {switch (i) {

case 0:strQuery = "select Groups.[Name] as 'Назва' from Groups where Groups.[Year] = 4"; break;

case 1:strQuery = "select [Surname] + ' ' + [Name] + ' ' + [Patronymic] as 'Викладач' from Teachers where [Date]<'01.01.1970'"; break;

case 2:strQuery = "select Departments.[Name] as 'Кафедра', Groups.[Name] as 'Група' from Departments join Groups on Departments.Id = Groups.DepartmentId"; break;

case 3:strQuery = "select Title as 'Назва предмету' from Subjects join Teachers on Subjects.TeacherId = Teachers.Id where Teachers.[Name]='Инга' and Teachers.Surname='Шалаганова'"; break;

case 4:strQuery = "select [Surname] + ' ' + [Name] + ' ' + [Patronymic] as 'Викладач', count(Subjects.Id) as 'Кількість предметів' from Teachers " +

"join Subjects on Subjects.TeacherId = Teachers.Id group by[Surname] +' ' + [Name] + ' ' + [Patronymic]"; break;

default: break;}}}}

Додаток В

Диск з програмою