**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп’ютерних наук та кібернетики

Кафедра математичної інформатики

«До захисту допущено»

Завідувач кафедри

Терещенко В.М

(підпис)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_р.

**Дипломна робота**

**На здобуття ступеня бакалавра**

за спеціальністю 122 Комп’ютерні науки

на тему:

**Перетворення Барроуза-Уіллера та його застосування**

Виконав студент 4-го курсу  
Волохович Ігор Ігорович

(підпис)

Науковий керівник:

професор, доктор фіз.-мат. наук

Анісімов Анатолій Васильович

(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Київ – 2020

**Реферат**

Обсяг роботи 32 сторінки, 20 ілюстрацій, 5 таблиць, 24 джерел посилань.

СТВОРЕННЯ WEB-ЗАСТОСУНКІВ, РОЗРОБКА БАЗ ДАНИХ, СТВОРЕННЯ ОБ’ЄКТНО-ОРІЄНТОВАНИХ БАЗ ДАНИХ ЗА ДОПОМОГОЮ МЕТОДУ CODE-FIRST APPROACH.

Об’єктом роботи є дослідження розробки баз даних для WEB-застосунків, їх зв’язок та виконання методу реєстрації нового користувача. Предметом роботи є дослідження різних моделей та підходів до формування баз даних, зв’язків між таблицями, полями.

Метою роботи є створення ефективної бази даних для доступного та швидкого доступу до будь-якої бажаної комірки, налаштування правильних зв’язків між таблицями та передавання між ними даних. Дослідження бази даних для збереження даних про студентів, вчителів, лабораторні роботи.

Методи дослідження: дослідження існуючих на даний момент ефективних баз даних та сайтів, які їх використовують, дослідження різних моделей та підходів до формування баз даних. Розробка бази даних методом Code First. Інструменти розробки: безкоштовне та вільно поширюване інтегроване середовище Visual Studio 2019 Community Edition[[1]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), Internet Information Services(IIS)[[4]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), bootstrap[[2]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), JavaScript[[3]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), мова програмування C#/ASP.NET[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)

Результат роботи: створено базу даних об’єктно-орієнтованого типу на основі описаних моделей; детально визначено конкретні зв’язки між моделями та наслідування даних; стабільність та гнучкість роботи застосунку, який спирається на задану базу даних; протестована ефективність саме цієї моделі бази даних, порівняно з аналогами; реалізовано метод сортування даних використовуючи контролери, спираючи на базу даних;

**Зміст**

[ВСТУП 4](#_Toc41829016)

[РОЗДІЛ 1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО БАЗИ ДАНИХ 6](#_Toc41829017)

[1.1 Перше застосування терміну «База Даних» 6](#_Toc41829018)

[1.2 Аналіз існуючих підходів 8](#_Toc41829019)

[1.3 Ранній період розвитку технології Баз Даних 9](#_Toc41829020)

[1.4 Архітектура баз даних 10](#_Toc41829021)

[1.5 Сьогоденний період Баз Даних 11](#_Toc41829022)

[РОЗДІЛ 2. МЕТОДИ СТВОРЕННЯ БАЗИ ДАНИХ 12](#_Toc41829023)

[2.1 Відомості про Entity Framework[7] 12](#_Toc41829024)

[2.2 Модель Code First 14](#_Toc41829025)

[2.3 Робочий процес методу Code First 15](#_Toc41829026)

[2.4 Концептуальна модель 15](#_Toc41829027)

[2.5 Переваги фреймворку 16](#_Toc41829028)

[РОЗДІЛ 3. РОЗРОБКА БАЗИ ДАНИХ 19](#_Toc41829029)

[3.1 Логічна структура бази даних 20](#_Toc41829030)

[3.2 Опис таблиць 21](#_Toc41829031)

[3.3 Аналіз створеної USE- CASE діаграми 22](#_Toc41829032)

[3.4 Реалізація методу «Code First» 23](#_Toc41829033)

[3.5 Реалізація алгоритму пошуку 27](#_Toc41829034)

[ВИСНОВКИ 30](#_Toc41829035)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 31](#_Toc41829036)

[ДОДАТКИ 33](#_Toc41829037)

# ВСТУП

**Оцінка сучасного стану об’єкта дослідження.** Створення моделі бази даних – це процес створення сукупності даних, які описують характеристику цих даних і взаємозв’язки між їх елементами. В загальному та у цьому випадку, стандартна база даних містить таблиці, схеми, подання, запити, збережені процедури та інші об’єкти. Із цього випливає, що база даних може містити, крім саме даних, опис цих даних та, можливо, засоби їх обробки.

На сьогоднішній момент існує багато різновидів та підходів до створення баз даних, які широко застосовуються з метою ефективного та швидкого реагування, раціональності зв’язків даних та налаштування під спеціальний вузький профіль. Крім того, існує різні класифікації БД за моделлю даних, технологією фізичного зберігання, вмістом та ступенем розподіленості. Тож варто відразу замислитися над цими критеріями перед створенням власної гнучкої та правильної бази даних.

Реалізацію баз даних у веб застосунках, як правило, простий користувач не бачить, але вправно користується, адже більшість функцій, які є на сайтах вправно записують дані користувача та беруть необхідні дані з бази даних. Бази даних для сайтів використовуються з ціллю зберігання різної інформації та представляє собою деякий набір взаємопов’язаних таблиць. Також завдяки роботі з базами даних можливе формування сторінок «на льоту» або ж редагування дій користувачів сайтів.

**Актуальність роботи та підстави для її виконання.** Звичайно зрозуміло, що створення будь-якого сайту не обходиться без створення доступної, простої та гнучкої бази даних. Для створення будь-якого сайту необхідна база даних. В даний час Бази даних містяться або створюються у всіх підприємствах малого, середнього та великого бізнесу : в банках; інвестиційних компаніях; пенсійних, благодійних фондах; у промислових, торгових підприємствах широкого спектру галузях народного господарства.

Тому я вважаю, що дослідження створення та роботи баз даних є достатньо актуальним. Створений проект має на **меті** полегшити буття вчителів та простих студентів, які переймаються за оригінальність своєї роботи. База даних допомагає впоратися з величезним потоком інформації.

**Мета й завдання роботи:** Отже, метою моєї курсової роботи є створення бази даних для веб-застосунку, створення зв’язків між таблицями, зручний пошук людей. Дослідження та створення баз даних на основі ефективно працюючих систем, зробити максимально простою та ефективною без обтяження. Створити методи для сортування даних за вказаними параметрами. Для досягнення цієї мети поставлено такі завдання:

* Теоретично дослідити існуючі та популярні бази даних сайтів
* Теоретично дослідити та створити тестову модель бази даних для подальшого тестування на стабільність, гнучкість та ефективність.
* Розробити модель бази даних для додатку ASP.NET[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), яка реалізує та створює базу даних на основі написаних моделей та контролерів.
* Створити методи для сортування даних за параметрами, визначеними у моделі
* Забезпечити адекватну роботу алгоритму пошуку та додавання даних у базу даних
* Проаналізувати створену базу даних та визначити, чи є вона оптимальною.

# РОЗДІЛ 1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО БАЗИ ДАНИХ

## Перше застосування терміну «База Даних»

В широкому розумінні, поняття «історії» баз даних зводиться до історії будь-яких методів, за допомогою яких люди вперше зберігали та обробляли дані.

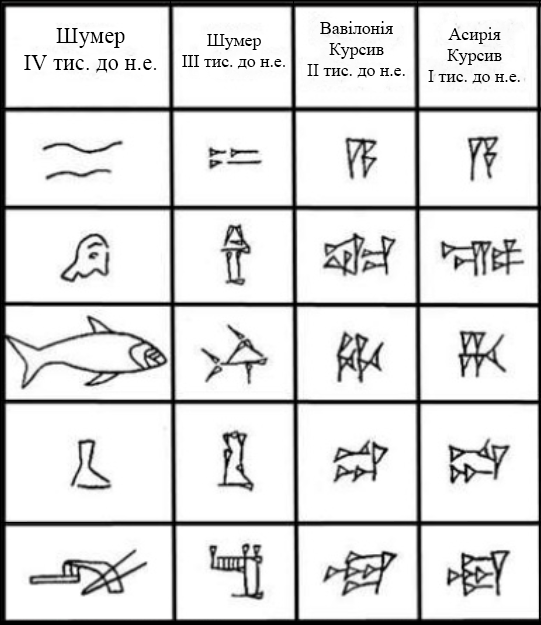


Рисунок 1.1 – Перші «Бази даних»

Наприклад: перші алфавіти(рис.1.1), засоби спілкування та зберігання інформації, засоби обліку царської казни та податків в стародавньому Шумері. Однак, в цьому випадку розмивається поняття «база даних» та фактичне поєднання з поняттями «архів» та навіть «письменність».

Розглядаючи традиційне (сучасне) розуміння, база даних – це множина взаємопов’язаних елементарних груп даних (інформації), які можуть оброблятися одною чи декількома прикладними системами.

Сьогодні, майже усі системи інформаційних технологій пов’язані з функціями довгострокового збереження й обробки інформації. Фактично, інформація є тим фактором, що дійсно визначає ефективність будь-якої сфери діяльності. З розвитком комп’ютерних технологій значно збільшилися інформаційні потоки та підвищилися вимоги до швидкої обробки даних. Тобто стало неможливо виконувати більшість операцій вручну, доводиться використовувати найкращі комп’ютерні технології.

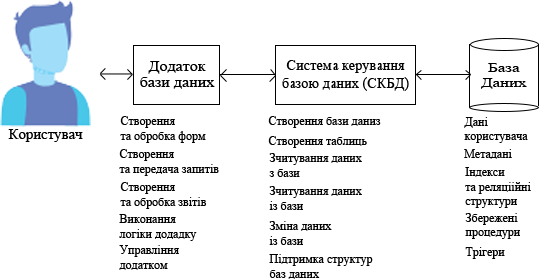


Рисунок 1.2 – Ієрархія взаємодії

На ранніх стадіях розвитку інформаційних технологій використовувалися списки – набиті на перфокарті та написані на магнітній стрічці. Однак, тільки деякі проблеми можна вирішити за допомогою таких списків. Ціль бази даних – допомогти людям та організаціям вести облік деяких речей(рис.1.2). Це одна з найбільш важливих сучасних комп’ютерних технологій. Сьогодні вони в більшості випадків асоціюються з банківськими транзакціями, хоча так було не завжди.

Головна перевага баз даних це швидкість внесення та використання потрібної інформації. Завдяки спеціальним алгоритмам, які використовуються для баз даних, із легкістю можна знайти необхідні дані всього за декілька секунд. Також в базі даних існує певний взаємозв’язок інформації: зміна в одному рядку може спричинити зміни в інших рядках – це допомагає працювати з інформацією швидке, зручніше та простіше.

## Аналіз існуючих підходів

Сьогодні при розробці інформаційних систем в цілому, і баз даних у зокрема, використовуються різні відомі методи, такі як : каскадна модель, поетапна модель з проміжним контролем, спіральна модель, модель еволюційної розробки (метод збільшень), кожен з яких має свої області застосування, переваги і недоліки.

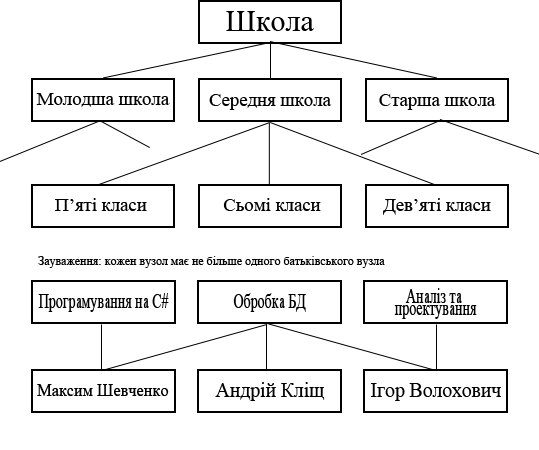
Однак, який би метод проектування не був обраний, доведеться хоча б один раз пройти всі його стадії, незалежно від того робити це послідовно або ітеративно. Складається завдання в розробці системи в цілому або її окремої частини, чи є технологія формальною або неформальною. І в цьому випадку ефективним рішенням могла б бути визначена технологія, яка:

а) мала б комплексне і загальне рішення, що дозволяє розробляти основні складові елементи ІС: бази даних і програмні додатки з нею працюють, інтероперабельними. Тобто сприяла б використанню вже існуючих компонентів, з метою відмови від екстенсивного ручного програміста праці в користь інтенсивних методів складання, а не розробки їх кожен раз унікальним способом під конкретне завдання;

б) дозволяла б прозоро та в стислі терміни розробляти будь-якого рівня складності нові БД і програмні додатки з нею працюють; в) дозволяла б проводити реінжиніринг існуючих баз даних, побудованих на різних принципах, платформах і містять значний обсяг різноманітних даних; г) спрощувала і здешевляла б процес подальшої експлуатації баз даних та ІС в цілому.

## Ранній період розвитку технології Баз Даних

Із самого започаткування сучасного поняття «База даних» були нереляційні бази даних (період 1968-1980 роки). Нереляційні бази даних (NoSQL[[16]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)) спеціально створені для особливих моделей даних та володіють гнучкими схемами, які дозволяють розробляти сучасні додатки. В 1971 році відбулася конференція по мовах обробки даних(CODASYL[[22]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)) в задачу якої входила розробка стандартів баз даних.

Новий стандарт був розширений на ієрархічну модель даних, яка застосовується в IMS[[21]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), яка була розроблена корпорацією IBM[[21]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), яка заклала основу концепції СКБД. Саме у цю еру були розроблені ієрархічні моделі даних(рис.1.3) та мережева моделі даних та незабаром з’явилися мережеві моделі даних. Рисунок 1.3 – Ієрархічна(зверху) та мережева(знизу) моделі даних

Особливість мережевої моделі була в тому, що будь-який запис може брати участь в деяких відношенням батько/дитина. Це дозволяло обходити цілий ряд обмежень ієрархічної моделі.

## Архітектура баз даних

Звісно очевидно, що ефективність будь-якої інформаційної системи залежить від її моделі або структури. В системах сьогоднішнього тисячоліття широко розповсюдилася архітектура «Клієнт-сервер». Комп’ютер, який керує цим процесом називається сервером. Клієнтом виступає програма або комп’ютер, база даних або файлові системи. В інформаційних системах із структурою «файл-сервер» (рис 1.4), по запитах клієнта файли або запити передаються на персональні комп’ютери клієнта де, власне, і виконується обробка даних. Великим недоліком такої системи є надлишкова кількість даних, які можуть передаватися між сервером та клієнтом.

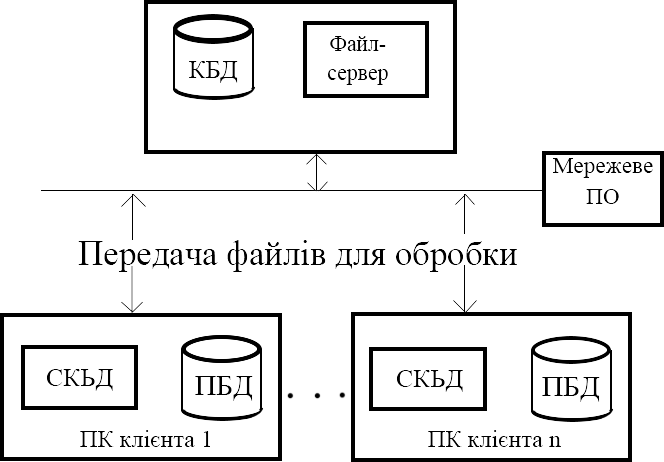


Рисунок 1.4 – База даних з архітектурою «файл-сервер»

Власне у груповому проекті застосовувалася структура розподіленої системи, яка побудована за архітектурою «клієнт-сервер» з використанням сервера баз даних (рис1.5). Така архітектура з сервером баз даних забезпечує виконання основного обсягу обробки даних, сильно не навантажуючи клієнтів. Користувачем формуються запити у формі LINQ[[23]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), які транслітеруються спеціальним фреймворком EF[[18]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) в SQL[[16]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ). Ці запити перекидаються до сервера баз даних. В свою чергу сервер бази даних вже виконує обробку даних, які потім надсилаються на екран комп’ютера юзера. Завдяки саме такому підходу, у співвідношенні з наведеним нижче, помітний значно менший обсяг даних, що передаються.

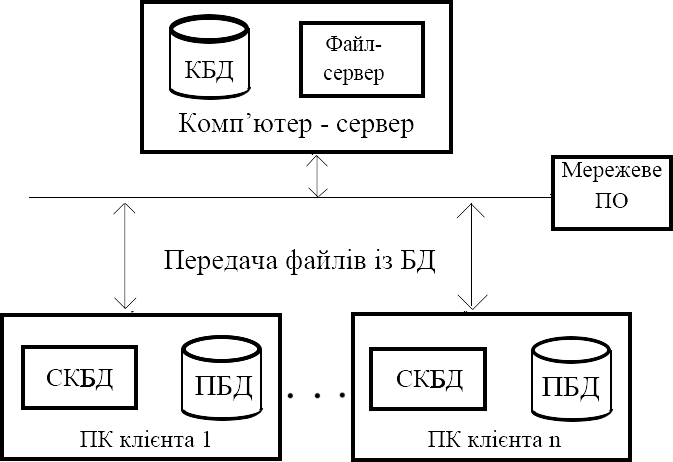


Рисунок 1.5 – База даних з архітектурою розподіленої системи

## Сьогоденний період Баз Даних

Для цього періоду характерне створення та використання реляційної моделі даних. Перевага такої моделі закладається в мінімальному дублюванні даних та виключенні деяких типів помилок, які притаманні іншим моделям. Згідно цієї моделі, дані зберігаються у виді таблиці з стовпчиками та рядками. Саме цей вид баз даних застосований у проекті курсової роботи.

# РОЗДІЛ 2. МЕТОДИ СТВОРЕННЯ БАЗИ ДАНИХ

## Відомості про Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)

Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) був представлений корпорацією Microsoft в 2008 році як основний засіб взаємодії між додатками .NET та реляційними базами даних. Це інструмент, який значно спрощує зіставлення об’єктів в програмному забезпеченні з таблицями та стовпцями реляційної бази даних. Варто зазначити, що Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) являється об’єктно-реляційним відображенням, яке націлене на підвищення швидкодії за рахунок скорочення задач по збереженню даних, які використовуються в додатках.

Entity Framework є сукупністю технологій, які підтримують розробку програмно-орієнтованих програм. Розробники, орієнтовані на дані, боролися з необхідністю досягнення двох різних між собою цілей. Вони хотіли мати можливість моделювати сутності, взаємини та бізнес-логіку, якими вони займаються, а також працювати з методами обробки даних, які використовуються для зберігання та отримання даних.

Дані можуть охоплювати декілька систем зберігання даних, кожна з яких має власні протоколи; навіть додатки, які працюють із єдиною системою зберігання, повинні співвідносити вимоги системи зберігання даних з вимогами написання ефективного та бездоганного коду програми. Фреймворк дозволяє в прямому сенсі «оживити» моделі.

Наступна схема ілюструє архітектуру Entity Framework для доступу до даних:

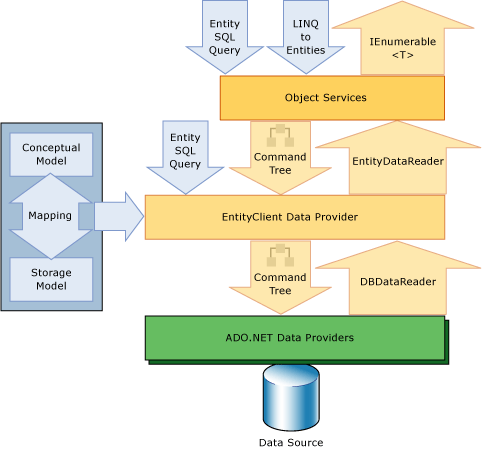


Рисунок 1.5 – Архітектура для доступу до даних

Інструменти моделей даних особи можуть генерувати клас, похідний від System.Data.Objects.ObjectContext або System.Data.Entity.DbContext, який представляє контейнер сутності в концептуальній моделі.

Цей об'єктний контекст надає засоби для відстеження змін та управління особистістю, одночасністю та відносинами. Цей клас також відкриває метод SaveChanges, який записує вставки, оновлення та видалення до джерела даних. Як і запити, ці зміни здійснюються або командами, автоматично генерованими системою, або збереженими процедурами, які вказані розробником.

В курсовій роботі я свідомо вибрав саме Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) через наступні переваги:

* Фреймворк вміє генерувати команди баз даних, необхідних для читання та запису даних а також їх виконувати.
* При необхідності можна виражати об’єкти через об’єкти домена використовуючи LINQ[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ).
* Вміє виконувати задані запити в базі даних, а потім виводить результати в екземплярах об’єктів домена, щоб було зручно працювати з ними у додатку.

Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) має більш високий рівень відображення, тому дозволяє налаштовувати співвідношення (рис2.1).

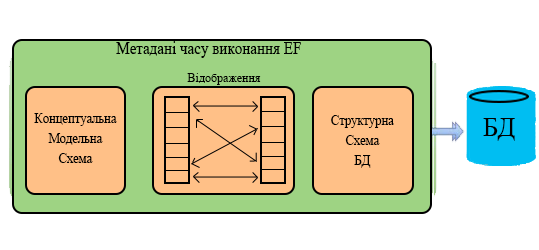


Рисунок 2.1 – Модель фреймворку Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)

## Модель Code First

Такий підхід було започатковано починаючи з Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) 4.1. Він корисний в основному у дизайні, керованому доменом. В такому підході ми зосереджуємося на домені додатка та починаємо створювати класи для домена, а не розробляти спочатку базу даних, а потім створювати класи, які відповідають базі даних.

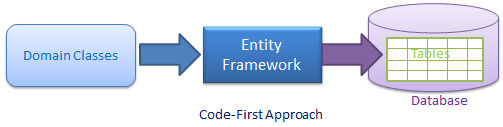


Рисунок 2.2 – Підхід «Спершу код»

Як видно із малюнка(рис 2.2), API фреймворку створить базу даних спираючись на доменні класи та конфігуратори. Це означає, що спочатку треба починати писати код, щоб потім EF створив на його основі базу даних.

## Робочий процес методу Code First

Наступне зображення іллюструє весь робочий процес розгортання бази даних цим методом:

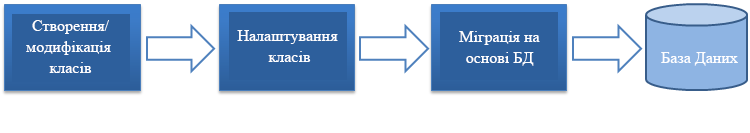


Рисунок 1.1 – Робочий процес фреймворку Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ)

Тобто весь робочий процес такого підходу зводиться до: Створити спеціальні класи – моделі -> налаштування класів для більш гнучкої та зручної розробки за допомогою Fluent-API[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) або атрибутів анотацій даних -> створення схеми бази даних використовуючи міграцію на основі бази даних.

## Концептуальна модель

Для розробників, які звикли до фокусування на детальній розробці бази даних, найбільший зсув таких думок із фреймворком спричиняє зміну фокуса на бізнес домен. Це дозволяє розширити можливості та стерти обмеження операції з базами даних. Отже, які переваги фреймворку?

* З Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), координатор називається концептуальною моделлю. Це модель об'єктів у програмі, а не модель бази даних, яка використовується для збереження даних застосунку.
* Концептуальна модель може узгоджуватися зі схемою у базі даних, або може бути зовсім іншою.
* Ми можемо використовувати візуальний дизайнер для визначення концептуальної моделі, яка потім може генерувати класи, які в кінцевому рахунку будуть використовуватися у програмі.
* Ми можемо просто визначити класи та використовувати функцію Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) під назвою Code First. І тоді Entity Framework осмислить концептуальну модель та створить на її основі таблицю у базі даних.

Так чи інакше, Entity Framework знаходить шлях, як перейти від концептуальної моделі(створених класів) до бази даних. Отже, можна запитувати об’єкти своєї концептуальної моделі та безпосередньо працювати з ними.

## Переваги фреймворку

Нижче наведені основні особливості Entity Framework. Цей список створений на основі найпомітніших функцій, а також із часто заданих питань щодо Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ).

* Являється розробкою Microsoft[[1]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ).
* Розробляється як продукт із відкритим кодом.
* Більше не прив'язаний до циклу випуску .NET[[10]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ).
* Працює з будь-якою реляційною базою даних з дійсним постачальником Entity Framework.
* Генерація команд SQL[[24]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) від LINQ[[23]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) до Entities.
* Вміє створювати параметризовані запити.
* Відстежує зміни в об'єктах пам'яті.
* Дозволяє вставляти, оновлювати та видаляти генерацію команд.
* Працює з візуальною моделлю або з власними заняттями.
* Зберігає підтримку процедур.

Використовуючи у поєднанні з ASP.NET[[10]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) маємо гнучку систему, в якій легко будувати достатньо великі додатки. ASP.NET[[10]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) надає розробникам свободу розробляти великі програми із різким скороченням кількості коду. Функція ADO.NET[[10]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) відключає доступ до бази даних, тому з'єднання не підтримуються тривалий час, тим самим покращуючи продуктивність та масштабованість веб-додатків. Також забезпечується краща швидкодія. ASP.NET[[10]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) забезпечує ранню прив'язку та своєчасну компіляцію, що забезпечує кращу ефективність роботи з нативною оптимізацією та кешуванням. Тут використовується модель інтелектуального клієнтського додатка, яка може працювати без будь-якої взаємодії з мережею та сервером, або отримувати дані з сервера лише за потреби. Це призводить до більш динамічного користувальницького досвіду та більш ефективного використання інфраструктури сервера клієнтів.

За допомогою вбудованої інформації про конфігурацію немає необхідності реєструвати компоненти, оскільки це спрощує розгортання програм ASP.NET[[10]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ). Більшість завдань розгортання автоматизовано за допомогою Інтернету інформаційних служб Microsoft (IIS), що є веб-сервером розширень, створеним Microsoft.

Розробникам надається свобода вибору мови, яка найкраще застосовує логіку програми. Додаток можна розділити на кілька мов. До класів та об’єктів в ASP.NET[[10]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) можна отримати доступ, не знаючи мови, якою написано веб-додаток.

ASP.NET[[10]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) забезпечує ранню прив'язку та своєчасну компіляцію, що забезпечує кращу ефективність роботи з нативною оптимізацією та кешуванням. Тут використовується модель інтелектуального клієнтського додатка, яка може працювати без будь-якої взаємодії з мережею та сервером, або отримувати дані з сервера лише за потреби. Це призводить до більш динамічного користувальницького досвіду та більш ефективного використання інфраструктури сервера клієнтів. Забезпечує спрощене виконання таких завдань, як подання форми, автентифікація клієнта тощо. Він надає набір елементів управління та інфраструктури, що спрощує створення веб-додатків. ASP.NET використовує власні мобільні додатки, а також мобільні веб-сайти за допомогою адаптивних дизайнерських рамок. Це також дозволяє легко виконувати розгортання та налаштування веб-сайтів.

Інтегроване середовище розробки Microsoft Visual Studio (IDE) надає багату панель інструментів, яка дозволяє розробникам та дизайнерам редагувати, контролювати серверне управління, багату бібліотеку класів тощо. Інструменти IDE легко працюють разом для створення веб-сайтів та веб-служб на базі ASP.NET. Visual Studio IDE можна персоналізувати різними способами, щоб підтримати стиль розробки та вимоги.

Саме через ці переваги я зупинився на розробці бази даних використовуючи саме цей фреймворк.

# РОЗДІЛ 3. РОЗРОБКА БАЗИ ДАНИХ

Для створення зручної бази даних я почав досліджувати існуючі сайти, вивчати їх моделі баз даних. Також аналітично розробляв різні варіанти бази даних, їх недоліки та переваги. Крім того дослідив комбінацію різних баз даних, поєднуючи переваги однієї та іншої. Отже, була створена база даних. Має буди передбачена можливість застосування методів сортування, додавання, видалення, редагування та перегляду інформації. У відношенні виду, моя праця являє собою створені моделі даних, бази даних та методу пошуку даних у Веб-застосунку використовуючи LINQ запити.

**Деталізація предметної області:**

Є студенти, які характеризуються прізвищем, ім’ям, номером групи, в якій навчаються, датою народження. Також за кожним студентом закріплена лабораторна робота, таблиця якої має поля назви роботи та предмету, для якого була виконана. Один студент може мати безліч лабораторних робіт.

У вчителів є студенти. Вчителі мають своє ім’я та прізвище, дату народження та назву кафедри, на якій вони знаходяться. Звісно, неможливо перевірити лабораторну роботу через алгоритм плагіату, якщо не завантажити її використовуючи таблицю бази даних під назвою «Здача». Вона характеризується полем з шляхом до файлу, назви лабораторної роботи та повне ім’я студента. Інші моделі мають технічний характер та реалізують ключові можливості веб-застосунку.

## 3.1 Логічна структура бази даних

Нижче представлена логічна структура бази даних, яка ілюструє зв’язки між таблицями.

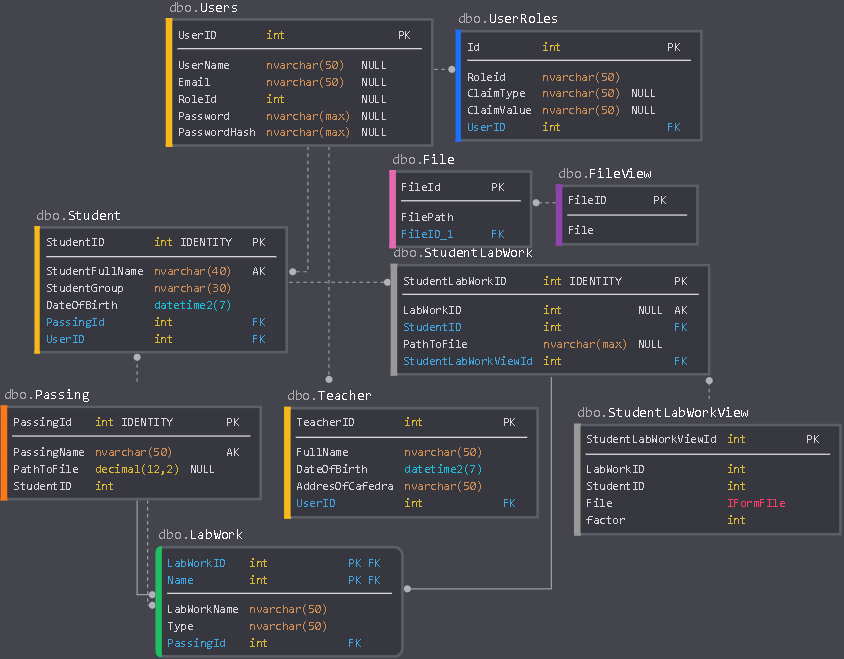


Рисунок 3.1 – Візуалізація Бази Даних

***STUDENT*** *– список усіх студентів, які є*

***PASSING****- список лабораторних робіт, які вже зареєстровані*

***LABWORK****- список лабораторних робіт, які мають бути весь курс*

***TEACHER****- список вчителів*

***STUDENTLABWORK****- список, який показує які лабораторні роботи завантажені*

***FILE****- метод зберігання файлів*

***FILEVIEW****- метод зберігання шляху до файлів та запровадження алгоритму*

***STUDENTLABWORKVIEW****- метод завантаження файлів*

Основна вимога до створення бази даних – забезпечення комфортної роботи команди розробників над подальшою модернізацією застосунку

## 3.2 Опис таблиць

У таблицях 1-5 наведено опис кожного поля таблиці.

Таблиця 1. Student

|  |  |  |
| --- | --- | --- |
| StudentID | int | Id студента |
| StudenFullName | int | Id лабораторної роботи |
| StudentGroup | text | Група студента |
| DateOfBirth | text | Дата народження |
| PassingID | date | Id здачі |

Таблиця 2. Passing

|  |  |  |
| --- | --- | --- |
| PassingID | int | Id перевірки |
| PassingName | text | Назва перевірки |
| PathToFile | text | Шлях до файлу |
| StudentID | int | Id студента |

Таблиця 3. LabWork

|  |  |  |
| --- | --- | --- |
| LabWorkID | int | Id лабораторної роботи |
| LabWorkName | int | Назва роботи |
| Type | text | Тип Роботи |
| PassingID | date | Id перевірки |

Таблиця 4. Teacher

|  |  |  |
| --- | --- | --- |
| TeacherID | int | Id вчителя |
| FullName | text | Повне ім’я |
| DateOfBirth | text | Дата народження |
| Address | date | Назва кафедри |

Таблиця 5. StudentLabWork

|  |  |  |
| --- | --- | --- |
| StudentLabWorkID | int | Id таблиці |
| LabWorkID | text | Назва лабораторної роботи |
| StudentID | data | Ім’я студента |
| PathToFile | text | Шлях до файду |

## 3.3 Аналіз створеної USE- CASE діаграми

Створення UML-діаграми дозволяє подивитися на задачу з різних точок зору та полегшує іншим людям зрозуміти суть задачі та спосіб її реалізації. Така діаграма просто та легко показує можливості проекту та допомагає зрозуміти суть кожного.

Use-Case[[20]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) – це перелік дій, сценаріїв по якому користувач взаємодіє з додатком для виконання якоїсь певної дії для досягнення конкретної цілі. Тестування по юз кейсам проводиться для того, щоб знайти додаткові логічні дірки та баги у додатках, які важко знайти у тестуванні індивідуальних модулів, частин додатку один від одного.

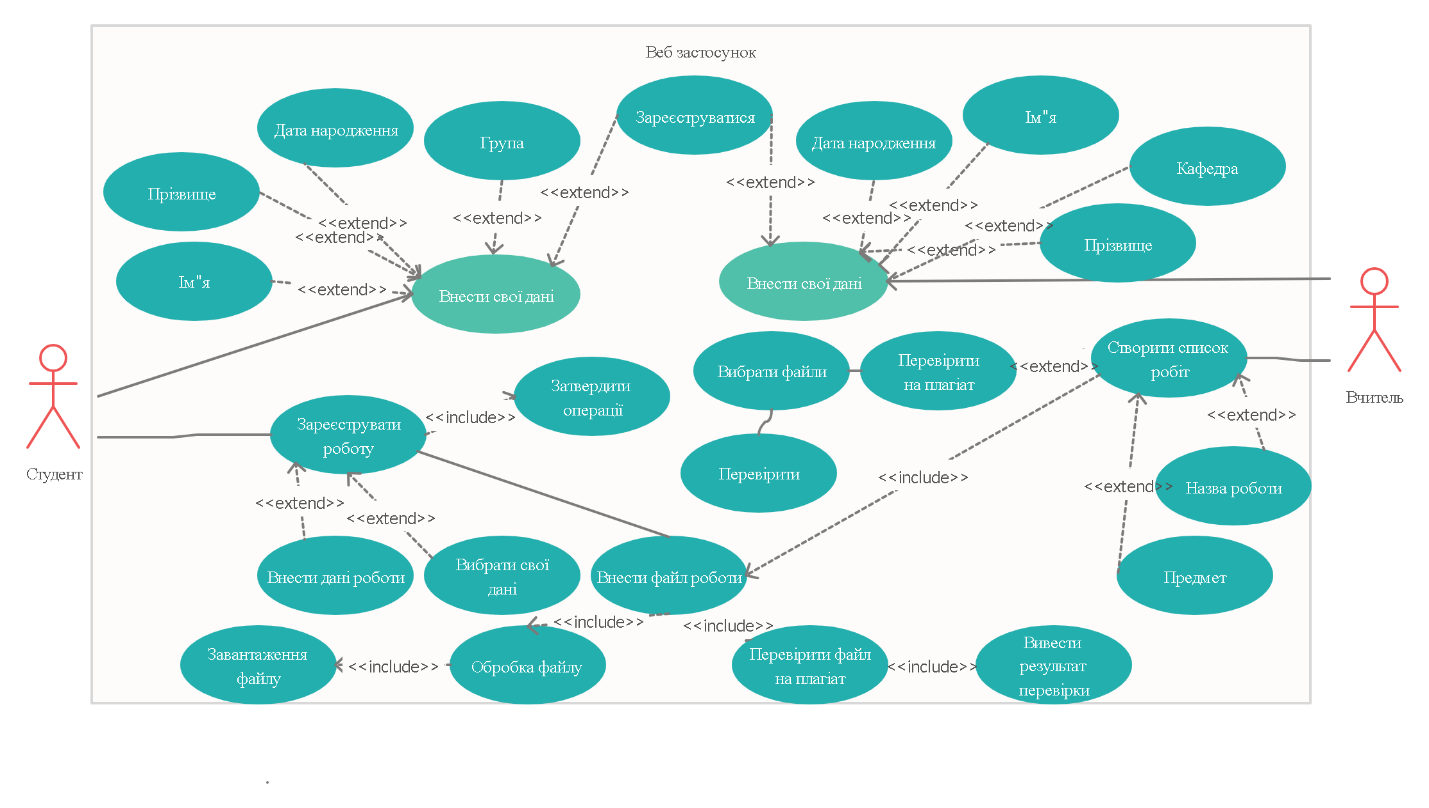


Рисунок 3.3 – Use Case діаграма можливостей

Спираючись на малюнок (рис 3.3), я описав сценарії взаємодії користувачів. Використовувати саме такі діаграми дуже зручно, адже все чітко та зрозуміло: хто, що та коли викликає та що отримується в результаті.

Я побудував діаграма для товаришів, щоб вона мати якісну та повну специфікацію дій та сценаріїв. Також ця система допомагає вишукувати проблеми та розбиратися що та на якому кроці пішло не так. Діаграма допомагає описати якусь частину функціональності, роботи користувача з інтерфейсом та інше. Завдяки цій діаграмі інші розробники мають змогу реалізувати новий, або модифікувати старі функції, прослідкувати звідки вони починаються.

## 3.4 Реалізація методу «Code First»

Для створення табличних даних використаємо метод «Спершу код». Спочатку створюємо моделі для всіх наших моделей, які були все оголошені зверху(див. Додаток A-З). Не забуваємо час від часу будувати проект, щоб нові таблиці та дані додавалися у базу даних. Налагоджуємо зв’язки між таблицями, використовуючи конструктори тих моделей, які хочемо зв’язати. Отже, створені атрибути мають створити нову базу даних, в яку користувач матиме можливість додавати, видаляти та редагувати дані.

Після компіляції проекту маємо створені за нашою моделлю таблиці, які пов’язані між собою (рис 3.4.1).

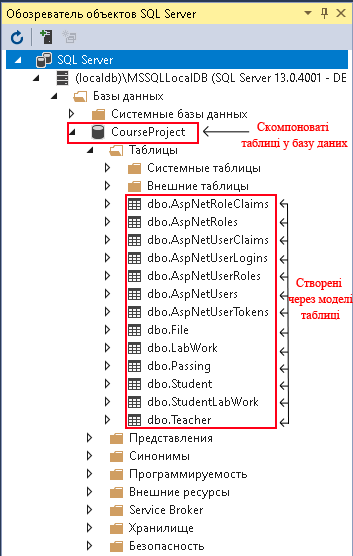


Рисунок 3.4.1 – Створена база даних

В реалізаціїї самого застосунку, маємо спеціальні поля для пошуку саме тих студентів, які нас цікавлять. Також варто зазначити, що є активними назви «Full Name», «Group», «DOB», що натякає на реалізований алгоритм сортування за ім’ям, групою та датою народження, що перетворює запити LINQ[[23]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) у запити SQL[[24]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ). Для активації сортування за тим чи іншим критерієм необхідно натиснути на назву того чи іншого стовпчика. Перейдемо до додавання студента. Через сайт створюємо трьох нових студентів та перевіряємо адекватність моделі таблиці «Студенти»

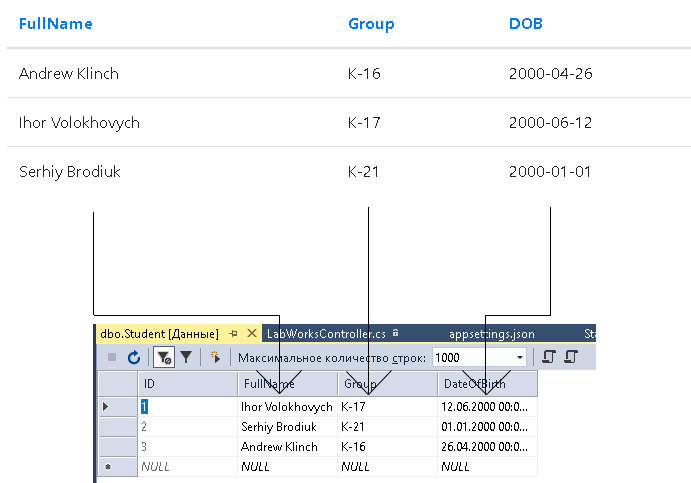


Рисунок 3.4.2 – Ідеальне перенесення даних

Метод додавання студента справно працює(рис 3.4.2), що натякає на правильність створенного контролера та адекватність моделі. Аналогічно створені інші контролери. Також існує можливість переглянути інформацію кожного студента та за необхідністю змінити чи видалити. Перевіримо, чи справно та адекватно працює задана модель та чи справно виконуються написані команди до бази даних(рис 3.4.4).

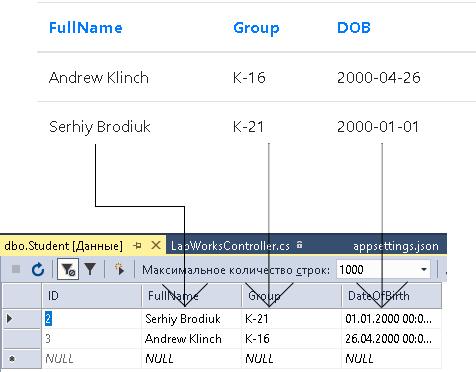


Рисунок 3.4.4 – Видалення студента Ihor Volokhovych

Після натискання кнопки «Delete» відбувається запит на видалення студента з таблиці та рекурсивно видаляються всі його завантажені роботи та значення з таблиць, тобто усі його роботи теж видаляються. Для того, щоб впевнитися чи працює справно рекурсивне видалення, створимо декілька запитів на перевірку лабораторних робіт та спробуємо їх видалити

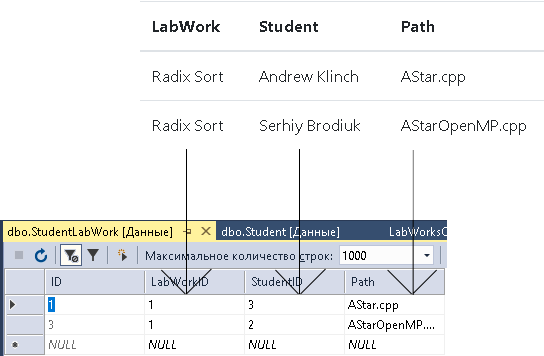


Рисунок 3.4.5 – Створена прив’язка «Студент-Робота»

Отже у нас є створений запис студента та зв'язок цього студента із роботою(рис 3.4.5), яку він завантажив. Як бачимо, написані запити адекватно записують правильний ідентифікатор лабораторної роботи, студента та шлях до файлу. При видаленні студента, рекурсивно видаляється всі інші записи, пов’язані з цим студентом. В деяких випадках, коли це дійсно неможливо, веб сайт видає користувачу помилку про неможливість виконання даної операції. Покажемо, що при видаленні студента видаляються усі його роботи з таблиці «StudentLabWorks». Для цього додам ще декілька лабораторних робіт, щоб було очевидно, що алгоритм працює (рис.3.4.6).

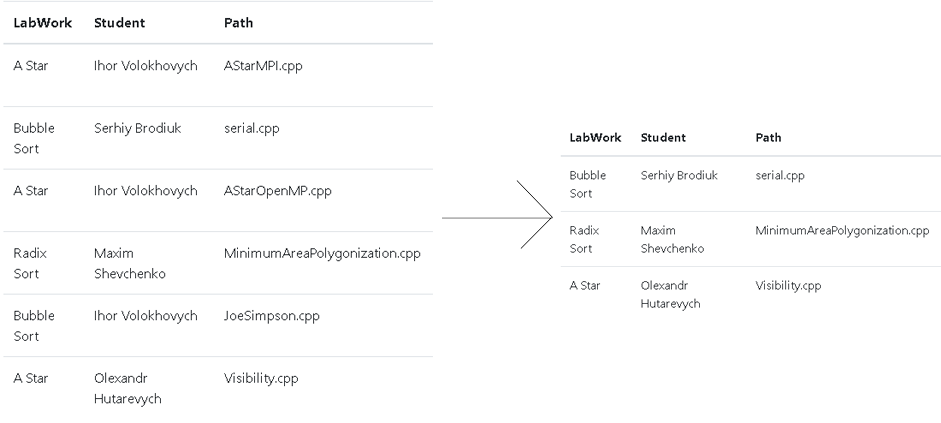


Рисунок 3.4.6 – Рекурсивне видалення користувача Ihor Volokhovych

## 3.5 Реалізація алгоритму пошуку

Алгоритм пошуку – важливий момент знаходження саме тих полів, які треба користувачу. Маємо реалізувати можливість знаходити дані використовуючи один із відомих полів. Видавати результати навіть частково співпадіння. Зважаючи на те, це повністю самостійно написаний контролер, він підтримує перетворення запитів LINQ у запити SQL, що значно спростить написання складних запитів до бази даних. Для маніпуляції з даними, створимо та напишемо власні контролери, які забезпечують сортування та пошук даних у таблиці[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) (див. Додаток I), полегшають користувачу знаходження корисних для нього даних.

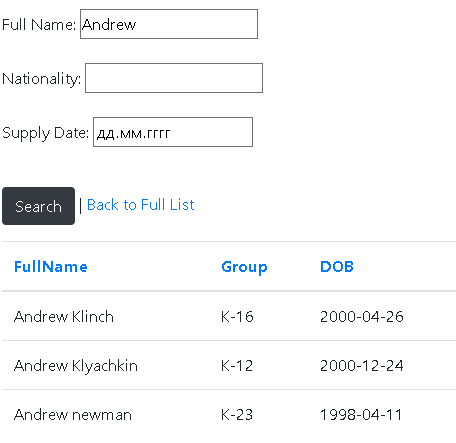


Рисунок 3.5.1 – Перевірка алгоритму пошуку

З наведеної ілюстрації видно, що алгоритм пошуку працює справно та шукає співпадіння вхідного значення зі значенням у базі даних.



Рисунок 3.5.2 – Алгоритм сортування

Збоку програмної частини, даний алгоритм виглядає як запити до моделі відсортувати дані. Реалізовано із кожною таблицею. Даний результат є досить зрозумілим для користувача(див. Додаток Д).

Дана курсова робота ставить на меті створення бази даних та гарного методу сортування та пошуку цільових даних для полегшення та збільшення привабливості використання саме цієї бази даних. Отже, підсумовуючи, я проаналізував існуючі моделі баз даних, реалізував власно створену базу даних для веб застосунку, який приймає значення студента, вчителя, лабораторної роботи, користувача. Забезпечує зв'язок між таблицями та рекурсивне видалення за допомогою написаних LINQ[[23]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) запитів, які перекладаються на команди до бази даних, пошук бажаних даних за будь яким із критеріїв, сортування даних.

# ВИСНОВКИ

Отже, в даній курсовій роботі був виконаний загальний огляд баз даних та фреймворку Entity Framework[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ). Були надані теоретичні дані та знання щодо баз даних, систем керування базами даних, фреймворку EF[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) та аналіз виконаної роботи. Були детально розглянуті теоретичні відомості про бази даних та засоби розробки ASP.NET CORE[[7]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ), представлені корпорацією Microsoft для безкоштовного використання.

Було створено повністю з нуля, перевірено та досліджено базу даних, яка приймає та обробляє дані. Створено моделі бази даних, методи сортування та пошуку за будь яким із значень таблиці у кожному з контролерів. Була протестована на працездатність та адекватність модель.

За допомогою розробленого проекту користувач може додавати, видаляти, шукати, редагувати та сортувати дані на сайті, переглядати дані студентів, лабораторних робіт, вчителів та інші задля перевірки. Завантажувати власні лабораторні роботи на сайт та перевіряти їх на плагіат. Справність цих методів забезпечує розумно розроблена проста модель бази даних та контролери для бази даних, реалізовані з перетворенням запитів LINQ[[23]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ) безпосередньо у запити мови SQL[[]](#_ПЕРЕЛІК_ВИКОРИСТАНИХ_ДЖЕРЕЛ).

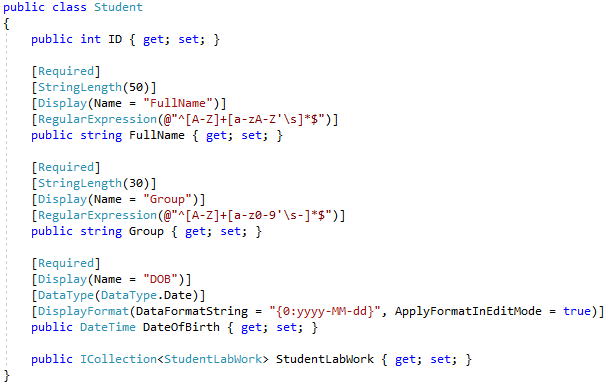
Результати створення показали, що створена база даних працює адекватно, має місце бути у даному проекті та не викликає ніяких проблем з модифікацією, готова до подальших модифікацій.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

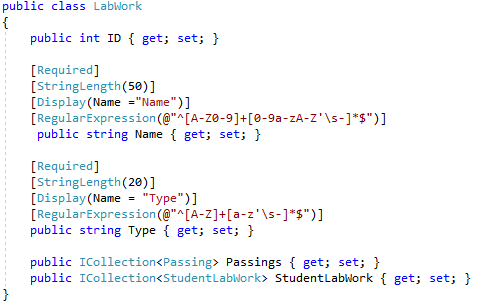
1. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: <https://visualstudio.microsoft.com/vs/>
2. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com
3. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
4. The Official Microsoft IIS Site [Електронний ресурс] – Режим доступу до ресурсу: https://www.iis.net
5. Windows [Електронний ресурс] – Режим доступу до ресурсу: <https://www.microsoft.com/uk-ua/windows/>
6. C# [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/
7. ASP.NET CORE [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1
8. A Brief History of Database Management [Електронний ресурс] – Режим доступу до ресурсу: https://www.dataversity.net/brief-history-database-management/
9. EF – CodeFirst Approach [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/entity\_framework/entity\_framework\_code\_first\_approach.htm
10. C# Corner [Електронний ресурс] – Режим доступу до ресурсу: https://www.c-sharpcorner.com
11. І.О. Завадський - Основи Баз [Електронний ресурс] – Режим доступу до ресурсу: http://itknyga.com.ua/docs/db\_comm\_final.pdf
12. М.А. Демиденко – Введення в сучасні бази даних [Електронний ресурс] – Режим доступу до ресурсу: http://ir.nmu.org.ua/handle/123456789/154887
13. История развития баз даных [Електронний ресурс] – Режим доступу до ресурсу: https://bourabai.ru/dbt/dbms/1.htm
14. История возникновения баз даных [Електронний ресурс] – Режим доступу до ресурсу: https://www.slideshare.net/DarynaMykytyn/ss-28377039
15. Що таке база даних? [Електронний ресурс] – Режим доступу до ресурсу: <http://apeps.kpi.ua/shco-take-basa-danykh>
16. NoSQL [Електронний ресурс] – Режим доступу до ресурсу:  [https://en.wikipedia.org/wiki/NoSQL](%20https://en.wikipedia.org/wiki/NoSQL)
17. ДДКБМТА – Основні поняття баз даних [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/view/ddkbmta-info/лекції/системи-керування-базами-даних-microsoft-access/основні-поняття-баз-даних
18. Entity Framework – Database First Approach [Електронний ресурс] – Режим доступу до ресурсу: <https://www.tutorialspoint.com/entity_framework/entity_database_first_approach.htm>
19. Creately – UML Diagram Tool [Електронний ресурс] – Режим доступу до ресурсу: https://creately.com/lp/uml-diagram-tool/
20. Как и зачем писать Use Cases – DOU [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/lenta/articles/use-cases/>
21. IBM Information Management System [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/IBM\_Information\_Management\_System
22. CODASYL [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/CODASYL>
23. What is LINQ [Електронний ресурс] – Режим доступу до ресурсу: <https://www.tutorialsteacher.com/linq/what-is-linq>
24. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/SQL

# ДОДАТКИ

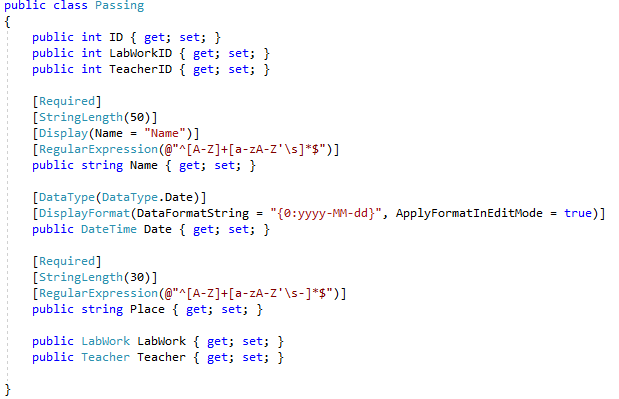
ДОДАТОК А



ДОДАТОК Б



ДОДАТОК В



ДОДАТОК Д

