# Розділ 1

**1.1 Огляд даних**

Завданням курсової роботи є розробка прикладної програми довідника мікросхем.

Довідник повинен містити найважливішу для користувача інформацію.

Основна інформація - це інформація про саму мікросхему - її позначення, вид, тип корпусу, виробника, а також інформація про використання мікросхеми, чи важливі властивості.

Опис типів корпусу і видів мікросхем теж повинен бути доступним для користувача.

Крім назви фірми виробника мікросхеми, в довіднику буде міститися посилання на логотип фірми, і країну де знаходяться потужності виробництва.

Отже довідник має містити інформацію про наступні об’єкти:

1. Мікросхема
2. Вид мікросхеми
3. Тип корпусу
4. Виробник

Інформація про мікросхему включає в себе інформацію про інші об’єкти, тобто між ними сформована залежність.

Інформація в довіднику постійно змінюється шляхом додаванням нових мікросхем, видаленням знятих з виробництва, оновленням властивостей старих. Тому реалізувати дані функції можна шляхом додавання, редагування, видалення даних з бази даних.

**1.2 Огляд інформаційних ресурсів**

Інтегральна схема - електронний прилад, який скла-дається з багатьох мініатюрних транзисто-рів та інших елементів схеми, об'єднаних у моноблок (чіп).

Види мікросхем:

· Аналогові

· Цифрові

· Аналого-цифрові

Аналогові мікросхеми — вхідні і вихідні сигнали змінюються за законом безупинної функції в діапазоні від позитивного до негативної напруги живлення.

Цифрові мікросхеми — вхідні і вихідні сигнали можуть мати два значення: логічний чи нуль логічна одиниця, кожному з який відповідає визначений діапазон напруги.

Аналого-цифрові мікросхеми сполучають у собі форми цифрової й аналогової обробки сигналів. В міру розвитку технологій одержують усе більше поширення.

Основним елементом аналогових мікросхем є транзистори (біполярні чи польові). Різниця в технології виготовлення транзисторів істотно впливає на характеристики мікросхем. Тому нерідко в описі мікросхеми вказують технологію виготовлення, щоб підкреслити тим самим загальну характеристику властивостей і можливостей мікросхеми. У сучасних технологіях поєднують технології біполярних і польових транзисторів, щоб домогтися поліпшення характеристик мікросхем.

Види за призначенням:

Інтегральна мікросхема може володіти закінченим, як завгодно складним, функціоналом — аж до цілого мікрокомп'ютера (однокристальний мікрокомп'ютер).

Аналогові схеми:

- Операційні підсилювачі ([підсилювач](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D1%81%D0%B8%D0%BB%D1%8E%D0%B2%D0%B0%D1%87) постійного струму з [диференційним входом](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BF%D1%96%D0%B4%D1%81%D0%B8%D0%BB%D1%8E%D0%B2%D0%B0%D1%87), що має високий [коефіцієнт підсилення](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B5%D1%84%D1%96%D1%86%D1%96%D1%94%D0%BD%D1%82_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D1%96). Призначений для виконання різноманітних операцій над [аналоговими сигналами](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2%D0%B8%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB), переважно, в схемах з [негативним зворотним зв'язком](https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B3%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B8%D0%B9_%D0%B7%D0%B2%D0%BE%D1%80%D0%BE%D1%82%D0%BD%D0%B8%D0%B9_%D0%B7%D0%B2'%D1%8F%D0%B7%D0%BE%D0%BA) (НЗЗ). Операційні підсилювачі застосовуються в різноманітних схемах радіотехніки, автоматики, інформаційно-вимірювальної техніки, — там, де необхідно підсилювати сигнали, в яких є постійна складова.)

- Генератори сигналів (пристрій, що дозволяє отримувати сигнал, що має задані характеристики (форму, енергетичні або статистичні характеристики і т. д.). Генератори широко використовуються для перетворення сигналів, для вимірювань і в інших областях)

**-** Фільтри (пристрій для виділення бажаних компонент [спектру](https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D0%BA%D1%82%D1%80) електричного сигналу та/або послаблення небажаних)

**-** Аналогові множники (служать для множення напруги або струмів)

**-** Стабілізатори джерел живлення ([перетворювач електричної енергії](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%82%D0%B2%D0%BE%D1%80%D1%8E%D0%B2%D0%B0%D1%87_%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%BE%D1%97_%D0%B5%D0%BD%D0%B5%D1%80%D0%B3%D1%96%D1%97), що дозволяє отримати на виході [напругу](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B0_%D0%BD%D0%B0%D0%BF%D1%80%D1%83%D0%B3%D0%B0), яка знаходиться в заданих межах, при значних коливаннях вхідної напруги і опору [навантаження](https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B2%D0%B0%D0%BD%D1%82%D0%B0%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F))

**-** Мікросхеми керування імпульсних блоків живлення

**-** Перетворювачі сигналів

Цифрові схеми:

- Логічні елементи ([пристрій](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%81%D1%82%D1%80%D1%96%D0%B9), призначений для обробки [інформації](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D1%8F) в [цифровій формі](https://uk.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB))

- Тригери ([електронна](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D1%96%D0%BA%D0%B0) логічна схема, яка має два стійкі стани, в яких може перебувати, доки не зміняться відповідним чином сигнали керування)

- Регістри (послідовний або паралельний логічний [пристрій](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%81%D1%82%D1%80%D1%96%D0%B9), який виконує функцію приймання, запам'ятовування і передавання інформації)

- Буферні перетворювачі

- Модулі пам'яті

- Мікроконтролери (спеціалізована [мікропроцесорна система](https://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), що включає [мікропроцесор](https://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80), блоки [пам'яті](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF'%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BF%D0%B0%D0%BC'%D1%8F%D1%82%D1%8C) для збереження [коду програм](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4) і даних, [порти](https://uk.wikipedia.org/w/index.php?title=%D0%9F%D0%BE%D1%80%D1%82_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)&action=edit&redlink=1) вводу-виводу і блоки зі спеціальними функціями ([лічильники](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%87%D0%B8%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA_(%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D1%96%D0%BA%D0%B0)), [компаратори](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BE%D1%80), [АЦП](https://uk.wikipedia.org/wiki/%D0%90%D0%A6%D0%9F) та інші))

- Мікропроцесори (у тому числі ЦПУ в комп'ютері)

- Однокристальні мікрокомп'ютери

Аналогово-цифрові:

- ЦАП

- АЦП

В електроніці цифро-аналоговий перетворювач (ЦАП) — пристрій для перетворення цифрового (звичайно двоичного) коду в аналоговий сигнал (струм, чи напруга заряд). Цифро-аналогові перетворювачі є інтерфейсом між абстрактним цифровим світом і реальними аналоговими сигналами.

Аналого-цифровий перетворювач (АЦП) робить зворотну операцію.

Звичайно ЦАП одержує на вхід цифровий сигнал в імпульсно-кодовій модуляції (PCM — pulse-code modulation). Задача перетворення різних стиснутих форматів у PCM виконується відповідними кодеками.

Аналого-цифровий перетворювач (АЦП) — пристрій, що перетворить вхідний аналоговий сигнал у дискретний код (цифровий сигнал). Зворотне перетворення здійснюється за допомогою ЦАП (цифро-аналогового перетворювача).

Як правило, АЦП — електронний пристрій, що перетворить напруга в двоичное цифровий код. Проте, деякі неелектронні пристрої, такі як перетворювач ріг-код, варто також відносити до АЦП.

Корпус [інтегральної мікросхеми](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) — це герметична несуча система і частина конструкції, призначена для захисту кристалу інтегральної схеми від зовнішніх впливів і для [електричного з'єднання](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D0%B0%D0%BA%D1%82) із зовнішніми [колами](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B5_%D0%BA%D0%BE%D0%BB%D0%BE) за допомогою виводів.

DIP (Dual In-line Package, також DIL) - тип корпусу мікросхем, микросборок і деяких інших електронних компонентів для монтажу в отвори друкованої плати. Має прямокутну форму з двома рядами виводів по довгим сторонам. Може бути виконаний з пластика (PDIP) або кераміки (CDIP).

SOIC або просто SO (small-outline integrated circuit), а також SOP (Small-Outline Package) корпус мікросхем, призначений для поверхневого монтажу, який займає на друкованій платі на 30-50% менше площі ніж аналогічний корпус DIP, а також має на 50 -70% меншу товщину

SIP (Single In-line Package) - плоский корпус для вертикального монтажу в отвори друкованої плати, з одним рядом виводів по довгій стороні.

QFP (Quad Flat Package) - плоский корпус з чотирма рядами контактів. Являє собою квадратний корпус з розташованими по краях контактами.

LCC (Leadless Chip Carrier) являє собою низькопрофільний квадратний керамічний корпус з розташованими на його нижній частині контактами, призначений для поверхневого монтажу.

PLCC (Plastic Leaded Chip Carrier) і СLCC (Ceramic Leaded Chip Carrier) представляють собою квадратний корпус з розташованими по краях контактами, призначений для установки в спеціальну панель (часто звану «ліжечком»).

TSOP (Thin Small-Outline Package) тонкий малогабаритний корпус, різновид SOP корпусу мікросхем. Часто застосовується в області DRAM, особливо для упаковки низьковольтних мікросхем через їх малого обсягу і великої кількості штирьків.

SSOP (Shrink small-outline package) (зменшений малогабаритний корпус) різновид SOP корпусу мікросхем, призначеного для поверхневого монтажу. Виводи розташовані по двох довгих сторонах корпусу.

ZIP (Zigzag-In-line Package) - плоский корпус для вертикального монтажу в отвори друкованої плати шнуром виводами, розташованими зигзагоподібно.

**Розділ 2**

**2.1 Обгрунтування вибору середовища розробки**

Для розробки програмного продукту було обрано мову програмування Microsoft Visual C#, відповідно, платформу .NET, інтегроване середовище розробки Microsoft Visual Studio 2019.

Вибір платформи .NET зумовлений рядом переваг:

- вся платформа .NET ґрунтується на єдиній обєктно-орієнтованій моделі;

- в склад платформи .NET входить «збиральник сміття», який звільняє ресурси, що захищає програми від втрат памяті і від необхідності звільняти ресурси;

- будь-яка програма, розроблена з допомогою .NET є автономною, в тому смислі, що не залежить від інших програм та від ОС;

- встановлення програми може бути проведене звичайним копіюванням файлів;

- використання безпечних типів даних, що підвищує надійність програми та сумісність;

- програма взаємодіє з єдиною моделлю обробки помилок;

- програми, написані на різних мовах можуть легко взаємодіяти;

- абсолютно всі помилки оброблюються механізмом виключних ситуацій, що дозволяє запобігти неоднозначностям, які виникали інколи при програмуванні під Win32;

- зручний спосіб повторного використання коду;

- можливість використання C# для написання динамічних web-сторінок ASP.NET.

- вбудована підтримка автоматичної генерації XML-документації.

Зокрема в С# є зручна технологія зв’язку програми з базою даних - ADO.NET, що важливо для виконання поставленого завдання.

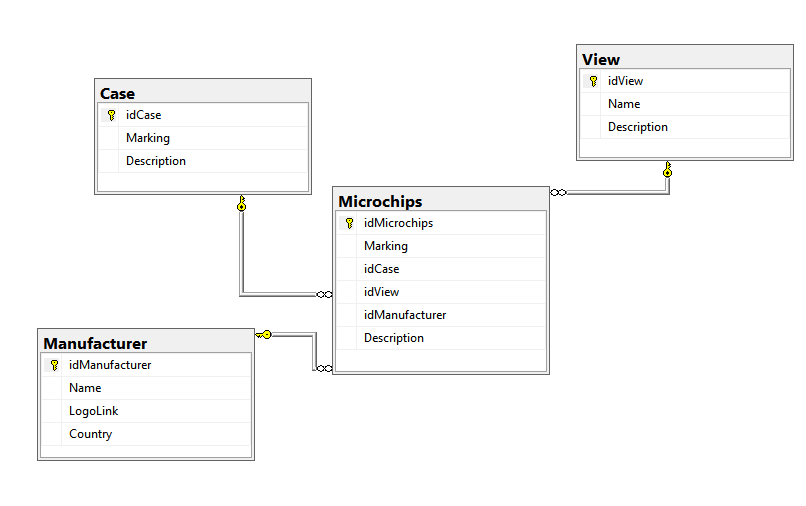
ADO.NET представляє собою технологію роботи з даними, яка заснована на платформі .NET Framework. Ця технологія являє нам набір класів, через які ми можемо відправляти запити до баз даних, встановлювати підключення, отримувати відповідь від бази даних і виробляти ряд інших операцій.

Причому важливо зазначити, що систем управління баз даних може бути безліч. У своїй сутності вони можуть відрізнятися. MS SQL Server, наприклад, для створення запитів використовує мову T-SQL, а MySQL і Oracle застосовують мову PL-SQL. Різні системи баз даних можуть мати різні типи даних. Також можуть відрізнятися якісь інші моменти. Однак функціонал ADO.NET побудований таким чином, щоб надати розробникам уніфікований інтерфейс для роботи з самими різними СУБД.

Основу інтерфейсу взаємодії з базами даних в ADO.NET представляє обмежене коло об'єктів: Connection, Command, DataReader, DataSet і DataAdapter. За допомогою об'єкта Connection відбувається установка підключення до джерела даних. Об'єкт Command дозволяє виконувати операції з даними з БД. Об'єкт DataReader зчитує отримані в результаті запиту дані. Об'єкт DataSet призначений для зберігання даних з БД і дозволяє працювати з ними незалежно від БД. І об'єкт DataAdapter є посередником між DataSet і джерелом даних. Головним чином, через ці об'єкти і буде йти робота з базою даних.

**2.2 Структурна схема бази даних**

Перед створенням самої бази даних потрібно визначити опис змісту, структури і обмежень цілісності БД. Всі ці характеристики включає в себе структурна схема БД, представлена на рисунку.



Згідно зі схемою, база даних складається з чотирьох таблиць: Microchips, Case, View, Manufacturer.

Таблиця Microchips містить всю необхідну інформацію про мікросхеми.

idMicrochips (тип int) - поле з цілочисельним типом даних, первинний ключ таблиці, має властивість автоінкременту.

Marking (тип varchar) - поле з символьним типом даних, в ньому зберігається позначення мікросхеми.

idCase (тип int) - зовнішній ключ таблиці Case

idView (тип int) - зовнішній ключ таблиці View

idManufacturer (тип int) - зовнішній ключ таблиці Manufacturer

Description (тип varchar) - поле з символьним типом даних, в ньому зберігається опис мікросхеми.

Таблиця Case містить інформацію про типи корпусів мікросхем.

idCase (тип int) - поле з цілочисельним типом даних, первинний ключ таблиці, має властивість автоінкременту.

Marking (тип varchar) - поле з символьним типом даних, в ньому зберігається позначення типу корпусу мікросхеми.

Description (тип varchar) - поле з символьним типом даних, в ньому зберігається опис корпусу.

Таблиця View містить інформацію про види мікросхем.

idView (тип int) - поле з цілочисельним типом даних, первинний ключ таблиці, має властивість автоінкременту.

Name (тип varchar) - поле з символьним типом даних, в ньому зберігається назва виду мікросхеми.

Description (тип varchar) - поле з символьним типом даних, в ньому зберігається опис виду мікросхеми.

Таблиця Manufacturer містить інформацію про виробників мікросхем.

idManufacturer (тип int) - поле з цілочисельним типом даних, первинний ключ таблиці, має властивість автоінкременту.

Name (тип varchar) - поле з символьним типом даних, в ньому зберігається назва фірми-виробника.

LogoLink (тип varchar) - поле з символьним типом даних, в ньому зберігається посилання на логотип фірми.

Country (тип varchar) - поле з символьним типом даних, містить інформацію про країну де розташовані потужності виробництва фірми.

Від таблиць Case (від поля idCase), View (від поля idView), Manufacturer (від поля idManufacturer) іде зв’язок один до багатьох (до полів idCase, idView, idManufacturer відповідно), оскільки є велика кількість різних мікросхем з одним типом корпусу, є велика кількість різних мікросхем одного виду, і одна фірма виготовляє багато різних мікросхем.

**2.2 UML-діаграма класів**

Програма складається з п‘яти класів (DataBaseOperations, Manufacturer, View, Case, Microchips), та семи класів Windows Form (MicrochipsOperations, ManufacturerOperations, ViewOperations, CaseOperations, Form1, FiltersMicrochips, FiltersOther).

Форма MicrochipsOperations пов‘язана асоціацією з класами:

1. DataBaseOperations (використовує об‘єкти цього класу),
2. Manufacturer (використовує властивості об‘єктів класу),
3. View (використовує властивості об‘єктів класу),
4. Case (використовує властивості об‘єктів класу),
5. Microchips (використовує властивості об‘єктів класу).

Всі інші форми теж пов‘язані асоціаціями з класом DataBaseOperations, оскільки використовують його об‘єкти.

Також форми MicrochipsOperations, CaseOperations, ViewOperatios, ManufacturerOperations пов‘язані з класами Microchips, Case, View, і Manufacturer відповідно, тому що використовують властивості об‘єктів відповідних класів.

Форма FiltersMicrochips також пов‘язана з класами Microchips, Case, View, Manufacturer. Форма FiltersOther пов‘язана і відповідно використовує властивості об‘єктів класів Case, View, Manufacturer.

РОЗДІЛ 3

**3.1 Огляд фрагментів коду**

Метод відкриття з відкриття з’єднання з базою даних:

public void OpenConnection()

{

\_SqlConnection = new SqlConnection(connectionsString);

try

{

\_SqlConnection.Open();

}

catch(SqlException)

{

MessageBox.Show("Невдалося підключитися до бази даних");

}

}

В методі ініціалізується новий екземпляр класу SqlConnection даними, отриманими з рядка connectionsString, в який записаний рядок з’єднання з базою даних. Далі викликається метод \_SqlConnection.Open(), який відкриває підключення до бази даних зі значеннями, які визначаються об’єктом \_SqlConnection. При винекненні помилки методом Show() класу MassageBox відкривається діалогове вікно з текстом "Невдалося підключитися до бази даних".

Метод вибірки з бази даних:

public DataTable Select(string selectQuery)

{

SqlCommand command = new SqlCommand(selectQuery, \_SqlConnection);

DataTable dataTable = new DataTable();

SqlDataReader dataReader = command.ExecuteReader();

dataTable.Load(dataReader);

return dataTable;

}

В метод передається змінна типу string із запитом вибірки із БД. В методі ініціалізується об’єкт command класу SqlCommand методом SqlCommand() в який передається змінна типу string із запитом вибору із БД та об’єкт класу SqlConnection. Ініціалізується об’єкт dataTable класу DataTable. Створюється і ініціалізується об’єкт класу SqlDataReader методом ExecuteReader(). Далі об’єкт dataTable ініціалізується даними з об’єкту dataReader методом Load(). Метод повертає об’єкт dataTable.

Метод додавання даних в БД:

public void Insert(string sqlCommandInsert)

{

SqlCommand \_SqlCommand = new SqlCommand(sqlCommandInsert, \_SqlConnection);

\_SqlCommand.ExecuteNonQuery();

MessageBox.Show("Дані додано");

}

В метод передається змінна типу string із запитом додавання в БД. В методі створюється та ініціалізується об’єкт класу SqlCommand методом SqlCommand(). Викликається метод ExecuteNonQuery(), який виконує виконує для підключення інструкцію Transact-SQL. Метод Show() класу MassageBox відкриває діалогове вікно з текстом " Дані додано ".

Метод оновлення даних БД:

public void Update(string sqlCommandUpdate)

{

SqlCommand \_SqlCommand = new SqlCommand(sqlCommandUpdate, \_SqlConnection);

\_SqlCommand.ExecuteNonQuery();

MessageBox.Show("Дані оновлено");

}

В метод передається змінна типу string із запитом оновлення даних в БД. В методі створюється та ініціалізується об’єкт класу SqlCommand методом SqlCommand(). Викликається метод ExecuteNonQuery(), який виконує виконує для підключення інструкцію Transact-SQL. Метод Show() класу MassageBox відкриває діалогове вікно з текстом " Дані оновлено ".

Метод видалення з БД:

public void Delete(string sqlCommandDelete)

{

SqlCommand \_SqlCommand = new SqlCommand(sqlCommandDelete, \_SqlConnection);

\_SqlCommand.ExecuteNonQuery();

MessageBox.Show("Дані видалено");

}

В метод передається змінна типу string із запитом видалення із БД. В методі створюється та ініціалізується об’єкт класу SqlCommand методом SqlCommand(). Викликається метод ExecuteNonQuery(), який виконує виконує для підключення інструкцію Transact-SQL. Метод Show() класу MassageBox відкриває діалогове вікно з текстом " Дані видалено ".

Метод закриття з’єднання із БД:

public void CloseConnection()

{

\_SqlConnection.Close()

}

Метод закриває з’єднання з базою даних.

Кнопка виведення даних з БД на форму:

private void button1\_Click(object sender, EventArgs e)

{

DBOperations.OpenConnection();

string commandString = "SELECT Microchips.idMicrochips, Microchips.Marking, [dbo].[Case].Marking, View.Name, Manufacturer.Name, Microchips.Description from Microchips " +

"inner join [dbo].[Case] on Microchips.idCase = [dbo].[Case].idCase " +

"inner join View on Microchips.idView = View.idView " +

"inner join Manufacturer on Microchips.idManufacturer = Manufacturer.idManufacturer ";

dataGridView1.DataSource = DBOperations.Select(commandString);

dataGridView1.Refresh();

DBOperations.CloseConnection();

}

В методі відкривається з’єднання із БД методом OpenConnection() класу DataBaseOperations, Створюється змінна типу string та ініціалізується текстом SQL запиту вибірки потрібних полів. Властивість DataSource класу DataGridView ініціалізується методом вибірки із БД Select(), в який передається змінна commandString з текстом запиту. Закривається з’єднання із базою даних методом CloseConnection().

Кнопка додавання даних в БД:

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

{

DBOperations.OpenConnection();

View view = (View)comboBox1.SelectedItem;

Case case1 = (Case)comboBox2.SelectedItem;

Manufacturer manufacturer = (Manufacturer)comboBox3.SelectedItem;

if ( (textMarking.Text.ToString() == "") || (textDescription.Text.ToString() == ""))

{

MessageBox.Show("Ви не заповнили усі поля");

}

else

{

string queryInsert = "INSERT INTO Microchips(Marking, idView, idCase, idManufacturer, Description) VALUES( '" + textMarking.Text.ToString() + "', '" + view.Id + "', '" + case1.Id + "', '" + manufacturer.Id + "', '" + textDescription.Text.ToString() + "')";

DBOperations.Insert(queryInsert);

}

DBOperations.CloseConnection();

}

Відкривається з’єднання із БД. Об’єкти view, case1, manufacturer класів View, Case, Manufacturer ініціалізуються методом SelectedItem відповідних комбобоксів. Далі перевіряється, чи заповнені текстбокси. Якщо хоч один пустий, то виводиться повідомлення "Ви не заповнили усі поля". Якщо текстбокси заповнені, в текстову змінну queryInsert записується SQL запит додавання даних в БД, де в ролі даних виступають значення властивостей об’єктів view, case1, manufacturer. Змінна передається в метод Insert(). Закривається з’єднання із БД.

Кнопка оновлення даних:

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

{

DBOperations.OpenConnection();

View view = (View)comboBox1.SelectedItem;

Case case2 = (Case)comboBox2.SelectedItem;

Manufacturer manufacturer1 = (Manufacturer)comboBox3.SelectedItem;

Microchips microchips = (Microchips)comboBoxToUpdate.SelectedItem;

if ((textMarking.Text.ToString() == "") || (textDescription.Text.ToString() == ""))

{

MessageBox.Show("Ви не заповнили усі поля");

}

else

{

string sqlUpdate = $"Update Microchips set Marking = '{textMarking.Text.ToString()}', idView = '{ view.Id }' , idCase = '{ case2.Id }' , idManufacturer = '{ manufacturer1.Id }', Description = '{textDescription.Text.ToString()}' where idMicrochips =" + microchips.Id;

DBOperations.Update(sqlUpdate);

}

DBOperations.CloseConnection();

}

}

Відкривається з’єднання із БД. Об’єкти view, case2, manufacturer1, microchips класів View, Case, Manufacturer, Microchips ініціалізуються методом SelectedItem відповідних комбобоксів. Далі перевіряється, чи заповнені текстбокси. Якщо хоч один пустий, то виводиться повідомлення "Ви не заповнили усі поля". Якщо текстбокси заповнені, в текстову змінну sqlUpdate записується SQL запит оновлення даних в БД, де в ролі даних виступають значення властивостей об’єктів view, case1, manufacturer, microchips. Змінна передається в метод Update(). Закривається з’єднання із БД.

Метод видалення із БД:

private void Button1\_Click\_1(object sender, EventArgs e)

{

DBOperations.OpenConnection();

Microchips microchips = (Microchips)comboBoxToUpdate.SelectedItem;

string sqlDelete = $"DELETE FROM Microchips WHERE idMicrochips =" + microchips.Id;

DBOperations.Delete(sqlDelete);

DBOperations.CloseConnection();

}

Відкривається з’єднання із базою даних. Об’єкт microchips класу Microchips ініціалізується методом SelectedItem комбобоксу. Текстова змінна sqlDelete ініціалізується текстом SQL запиту видалення даних із БД. Змінна передається в метод Delete(). Закривається з’єднання із базою даних.

Кнопка пошуку:

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

{

DBOperations.OpenConnection();

View view = (View)comboBox1.SelectedItem;

if (comboBox1.Text.ToString() != "")

{

string commandString = "SELECT \* FROM [dbo].[View] WHERE Name = '" + comboBox1.Text.ToString() + "'";

dataGridView1.DataSource = DBOperations.Select(commandString);

dataGridView1.Refresh();

}

if (comboBox1.Text.ToString() == "" & view != null)

{

string commandString = "SELECT \* FROM [dbo].[View] WHERE Name = '" + view.Name + "'";

dataGridView1.DataSource = DBOperations.Select(commandString);

dataGridView1.Refresh();

}

if (comboBox1.Text.ToString() != "" & view != null)

{

string commandString = "SELECT \* FROM [dbo].[View] WHERE Name = '" + view.Name + "'";

dataGridView1.DataSource = DBOperations.Select(commandString);

dataGridView1.Refresh();

}

if (comboBox1.Text.ToString() == "" & view == null)

{

MessageBox.Show("Виберіть чи введіть значення");

}

DBOperations.CloseConnection();

}

Відкривається з’єднання з БД. Об’єкт view ініціалізується даними з комбобоксу. Далі перевіряється, чи введені значення в текстове поле комбобоксу, або чи вибрані значення із списку комбобоксу. Якщо текстове поле порожнє і значення не вибране, метод Show() класу MassageBox відкриває діалогове вікно з текстом "Виберіть чи введіть значення". Якщо значення введене, або вибране в комбобоксі, в змінну commandString записується SQL запит вибірки із БД. Змінна передається в метод Select(), який ініціалізує властивість DataSource. Закривається з’єднання із базою даних.