# Техника безопасности на рабочем месте.

Перед тем, как включить компьютер, необходимо уделить внимание следующим действиям:

* Нужно убедиться, что в зоне досягаемости отсутствуют оголенные провода и различные шнуры;
* Нельзя начинать работу на технике с видимым повреждением;
* Предметы на столе не должны мешать обзору, пользованию мышкой и клавиатурой, поверхность экрана должна быть абсолютно чистой;
* Запрещается начинать работу в помещениях с повышенной влажностью;

При выполнении работы:

* Рекомендуется избегать расположения жидкостей рядом с модулями компьютера;
* Нельзя очищать поверхность компьютера от загрязнений, когда он включен;
* Недопустимо снимать корпус любой из составных частей ПК во время его работы;
* В помещении с компьютерами непозволительно курить или употреблять пищу на рабочем месте;
* При ощущении запаха гари, необходимо выключить ПК из сети;

Чтобы минимизировать негативные влияния при длительном нахождении за монитором, стоит знать:

* Расстояние между глазами пользователя и экраном составляет не менее полуметра;
* Клавиатура размещается за 20-30 сантиметров от края стола;
* Локти согнуты под прямым углом;
* Стул стоит таким образом, что спина немного упирается в его спинку;
* Ноги упираются в твердую поверхность, распрямлены вперед;

Оглавление

[Техника безопасности на рабочем месте. 1](#_Toc167403310)

[Практическая работа №1 Разработка приложению по созданию и управлению базами данных 4](#_Toc167403311)

[1. Создание WinForms проекта: 4](#_Toc167403312)

[2. Добавление компонентов для главной формы: 5](#_Toc167403313)

[3. Функционал вывода доступных баз: 5](#_Toc167403314)

[4. Функционал создания файлов баз данных: 6](#_Toc167403315)

[5. Функционал удаления файлов баз данных: 6](#_Toc167403316)

[6. Создание формы «Редактор» и добавление компонентов: 6](#_Toc167403317)

[7. Создание формы «АИС» и добавление компонентов: 7](#_Toc167403318)

[8. Переход в «АИС»: 7](#_Toc167403319)

[9. Переход в «Редактор»: 7](#_Toc167403320)

[10. Вывод имен сущностей в ListView (Редактор): 8](#_Toc167403321)

[11. Вывод атрибутов выбранной сущности в ListView (Редактор): 9](#_Toc167403322)

[12. Добавление нового атрибута сущности в базу данных (Редактор): 9](#_Toc167403323)

[13. Добавление новой сущности в базу данных (Редактор): 10](#_Toc167403324)

[14. Удаление сущности базы данных (Редактор): 10](#_Toc167403325)

[15. Модифицирование сущностей (Редактор): 11](#_Toc167403326)

[16. Добавление нового атрибута сущности в базу данных (Редактор): 12](#_Toc167403327)

[17. Удаление атрибутов сущности базы данных (Редактор): 12](#_Toc167403328)

[18. Вывод имен сущностей в ListView (АИС): 13](#_Toc167403329)

[19. Вывод данных в ListView (АИС): 13](#_Toc167403330)

[20. Добавление, удаление, обновление (АИС): 13](#_Toc167403331)

[21. Фильтрация данных (АИС): 14](#_Toc167403332)

[22. Динамическое обновление: 14](#_Toc167403333)

[23. Обработка исключений: 15](#_Toc167403334)

[24. Оптимизация и комментирование кода: 15](#_Toc167403335)

## Практическая работа №1 Разработка приложению по созданию и управлению базами данных

1. Создание WinForms проекта:
   * Создайте новый проект типа Windows Forms App в Visual Studio
2. Добавление компонентов для главной формы:
   * Разместите ListView и кнопки на форме:
   * Добавление компонента кнопки на форму:
   * Перетащите компонент Button на форму с помощью средства визуального редактора.
   * Настройка свойств кнопки:
   * Установите свойства Location, Name, Size, TabIndex, Text и UseVisualStyleBackColor для каждой кнопки.
   * Location определяет положение кнопки на форме.
   * Name - уникальное имя для идентификации кнопки в коде.
   * Size - размер кнопки.
   * TabIndex - порядок перехода между элементами формы при нажатии на клавишу Tab.
   * Text - текст, отображаемый на кнопке.
   * UseVisualStyleBackColor - определяет, будет ли кнопка использовать стиль оформления операционной системы.
   * Подписка на событие Click:
   * Для каждой кнопки добавьте обработчик события Click, который будет вызываться при нажатии на кнопку.
   * В обработчике события Click выполняйте необходимые действия при нажатии на кнопку.
   * Код для обработчиков событий:
   * В коде формы создайте методы для обработки событий Click для каждой кнопки.
   * Внутри этих методов реализуйте логику, которая должна выполняться при нажатии на кнопку. (Рисунок 1)

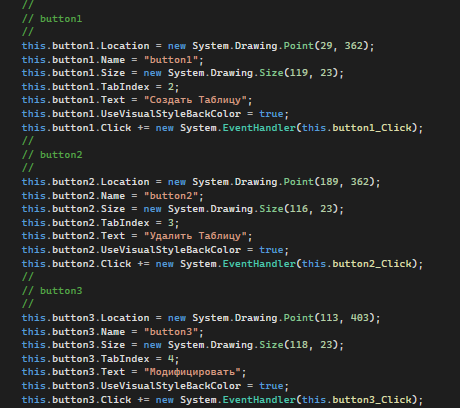


Рисунок 1 - Добавление компонентов

1. Функционал вывода доступных баз:
   * Алгоритм для вывода доступных файлов:

* Очистка предыдущих элементов ComboBox:
  + - Используется метод Clear для удаления всех элементов из ComboBox.
* Создание экземпляра DirectoryInfo:
  + - Создается новый экземпляр DirectoryInfo, используя путь к каталогу, который должен быть заполнен.
* Получение списка файлов в каталоге:
  + - Используется метод GetFiles для получения массива FileInfo объектов, представляющих файлы в каталоге.
* Добавление имен файлов в ComboBox:
  + - Проходимся по каждому файлу в массиве files и добавляем его имя в ComboBox с помощью метода Add.
* Возврат результата:
  + - Метод ничего не возвращает, так как все изменения вносятся непосредственно в ComboBox, переданный в качестве аргумента. (Рисунок 2)

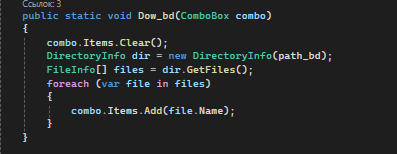


Рисунок 2 – Функция вывода

1. Функционал создания файлов баз данных:
   * Создание базы данных:
   * Определение пути к каталогу баз данных:
   * Используется статическая переменная path\_bd, которая объединяет текущий рабочий каталог приложения с подкаталогом "DATABASE".
   * Создание полного пути к файлу базы данных:
   * Используется метод Path.Combine для объединения пути к каталогу баз данных и имени базы данных, переданного в параметре name\_bd.
   * Проверка существования файла базы данных:
   * Используется метод File.Exists для проверки, существует ли файл базы данных по указанному пути.
   * Создание файла базы данных, если он не существует:
   * Если файл базы данных не существует, вызывается статический метод SQLiteConnection.CreateFile из библиотеки SQLite для создания файла базы данных.
   * После создания файла базы данных вызывается метод AddTable для добавления таблицы в базу данных.
   * Обработка ситуации, когда файл базы данных уже существует:
   * Если файл базы данных уже существует, выводится сообщение об ошибке с помощью MessageBox.Show. (Рисунок 3)

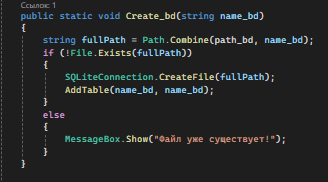


Рисунок 3 – Создание файлов баз данных

1. Функционал удаления файлов баз данных:
   * Удаление базы данных:

* Определение пути к каталогу баз данных:
  + Используется статическая переменная path\_bd, которая должна быть определена и инициализирована где-то в вашем коде.
* Перебор выбранных элементов в ListView:
  + Метод принимает ListView в качестве параметра и проходит по каждому выбранному элементу в списке SelectedItems.
* Удаление файла базы данных:
  + Для каждого выбранного элемента метод использует File.Delete для удаления файла базы данных, объединяя путь к каталогу баз данных с именем файла, полученным из Text выбранного элемента. (Рисунок 4)

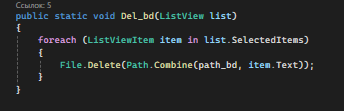
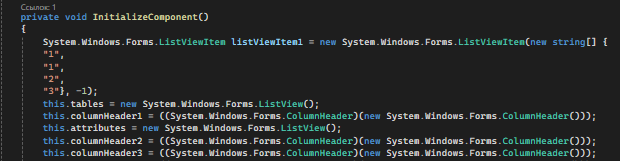
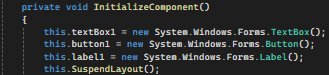


Рисунок 4 – Удаление БД

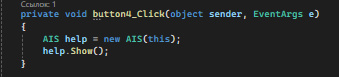
1. Создание формы «Редактор» и добавление компонентов:
   * Разместите ListView и кнопки на форме:



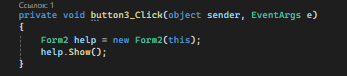
1. Создание формы «АИС» и добавление компонентов:
   * Разместите ListView и кнопки на форме:



1. Переход в «АИС»:
   * Реализуйте выбор базы данных и переход в форму «АИС»:



1. Переход в «Редактор»:
   * Реализуйте выбор базы данных и переход в форму «Редактор»:



1. Вывод имен сущностей в ListView (Редактор):
   * Получение имен сущностей:

* Формирование строки подключения к базе данных:
  + Строка подключения создается путем объединения пути к каталогу баз данных и имени выбранной базы данных.
* Создание и открытие соединения с базой данных:
  + Создается экземпляр SQLiteConnection с использованием строки подключения.
  + Соединение открывается с помощью метода Open.
* Выполнение запроса PRAGMA:
  + Выполняется SQL-запрос PRAGMA table\_info(selected\_bd), который возвращает информацию о структуре таблицы базы данных.
* Чтение результатов запроса:
  + Результаты запроса считываются с помощью ExecuteReader.
  + Для каждой строки результата извлекаются данные о названии столбца, типе данных, признаке NOT NULL и признаке первичного ключа.
* Добавление информации в ListView:
  + Для каждой строки результата создается новый экземпляр ListViewItem с информацией о столбце и добавляется в ListView.
* Закрытие соединения:
  + Соединение автоматически закрывается в блоке using, так как SQLiteConnection реализует интерфейс IDisposable.

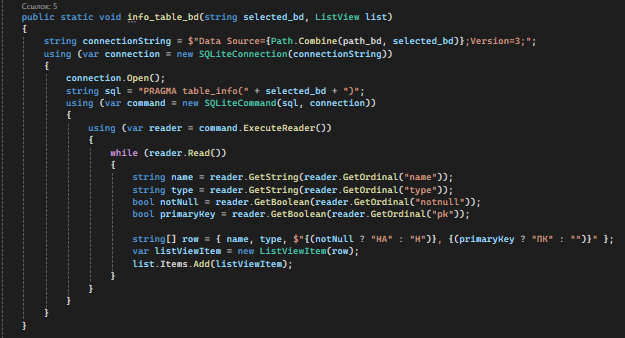


Рисунок 10 – Вывод имен сущностей

1. Вывод атрибутов выбранной сущности в ListView (Редактор):
   * Получение атрибутов сущности:

* Формирование строки подключения к базе данных:
  + Строка подключения создается путем объединения пути к каталогу баз данных и имени выбранной базы данных.
* Создание и открытие соединения с базой данных:
  + Создается экземпляр SQLiteConnection с использованием строки подключения.
  + Соединение открывается с помощью метода Open.
* Выполнение запроса PRAGMA:
  + Выполняется SQL-запрос PRAGMA table\_info(selected\_bd), который возвращает информацию о структуре таблицы базы данных.
* Чтение результатов запроса:
  + Результаты запроса считываются с помощью ExecuteReader.
  + Для каждой строки результата извлекаются данные о названии столбца, типе данных, признаке NOT NULL и признаке первичного ключа.
* Добавление информации в ListView:
  + Для каждой строки результата создается новый экземпляр ListViewItem с информацией о столбце и добавляется в ListView.
* Закрытие соединения:
  + Соединение автоматически закрывается в блоке using, так как SQLiteConnection реализует интерфейс IDisposable.

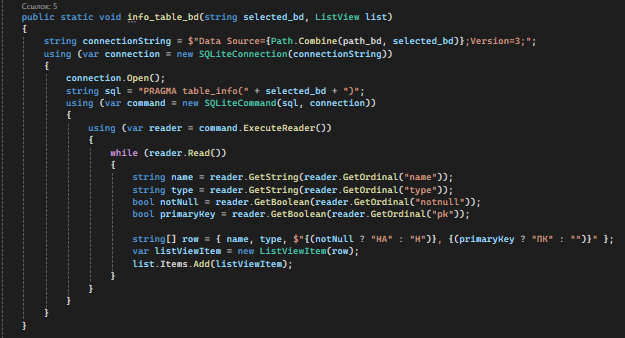


Рисунок 11 – Вывод атрибутов

1. Добавление нового атрибута сущности в базу данных (Редактор):
   * Добавление атрибута:

* Формирование строки подключения к базе данных:
  + Строка подключения создается путем объединения пути к каталогу баз данных и имени базы данных.
* Создание и открытие соединения с базой данных:
  + Создается экземпляр SQLiteConnection с использованием строки подключения.
  + Соединение открывается с помощью метода Open.
* Формирование SQL-запроса для добавления столбца:
  + SQL-запрос формируется на основе переданных параметров, включая имя столбца, его тип, признак NOT NULL, значение по умолчанию и признак первичного ключа.
* Выполнение SQL-запроса:
  + SQL-запрос выполняется с помощью ExecuteNonQuery для выполнения команд, которые не возвращают результаты запроса, таких как INSERT, UPDATE, DELETE, ALTER TABLE и т.д.
* Закрытие соединения:
  + Соединение автоматически закрывается в блоке using, так как SQLiteConnection реализует интерфейс IDisposable.

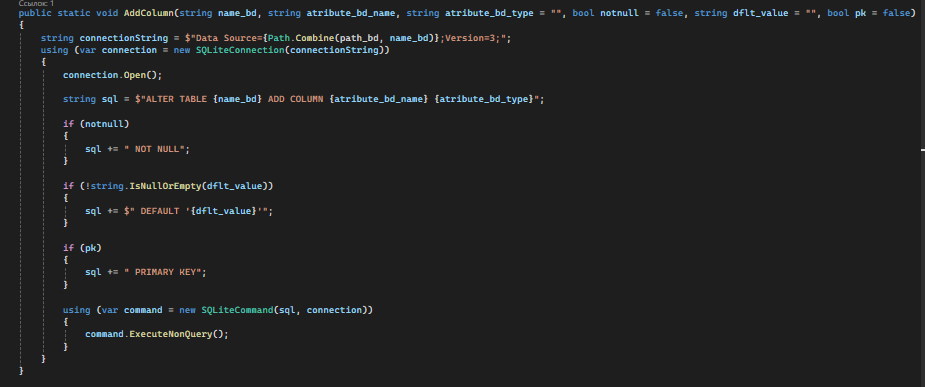


Рисунок 12 – Добавление нового атрибута в бд

1. Добавление новой сущности в базу данных (Редактор):
   * Добавление сущности:

* Формирование строки подключения к базе данных:
  + Строка подключения создается путем объединения пути к каталогу баз данных и имени базы данных.
* Создание и открытие соединения с базой данных:
  + Создается экземпляр SQLiteConnection с использованием строки подключения.
  + Соединение открывается с помощью метода Open.
* Формирование SQL-запроса для создания таблицы:
  + SQL-запрос формируется для создания таблицы с указанным именем и столбцом ID, который является первичным ключом и автоматически увеличивается при каждом добавлении новой записи.
* Выполнение SQL-запроса:
  + SQL-запрос выполняется с помощью ExecuteNonQuery для выполнения команд, которые не возвращают результаты запроса, таких как INSERT, UPDATE, DELETE, ALTER TABLE и т.д.
* Закрытие соединения:
  + Соединение автоматически закрывается в блоке using, так как SQLiteConnection реализует интерфейс IDisposable.

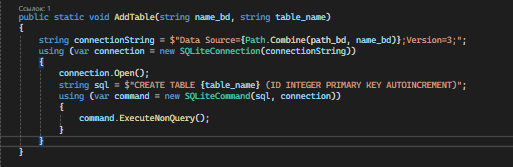


Рисунок 13 - Добавление новой сущности в базу данных (Редактор)

1. Удаление сущности базы данных (Редактор):
   * Удаление сущности:

* Формирование строки подключения к базе данных:
  + Строка подключения создается путем объединения пути к каталогу баз данных и имени базы данных.
* Создание и открытие соединения с базой данных:
  + Создается экземпляр SQLiteConnection с использованием строки подключения.
  + Соединение открывается с помощью метода Open.
* Формирование SQL-запроса для создания таблицы:
  + SQL-запрос формируется для создания таблицы с указанным именем и столбцом ID, который является первичным ключом и автоматически увеличивается при каждом добавлении новой записи.
* Выполнение SQL-запроса:
  + SQL-запрос выполняется с помощью ExecuteNonQuery для выполнения команд, которые не возвращают результаты запроса, таких как INSERT, UPDATE, DELETE, ALTER TABLE и т.д.
* Закрытие соединения:
  + Соединение автоматически закрывается в блоке using, так как SQLiteConnection реализует интерфейс IDisposable.

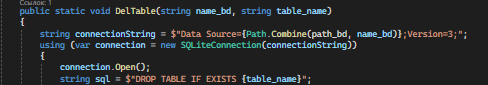
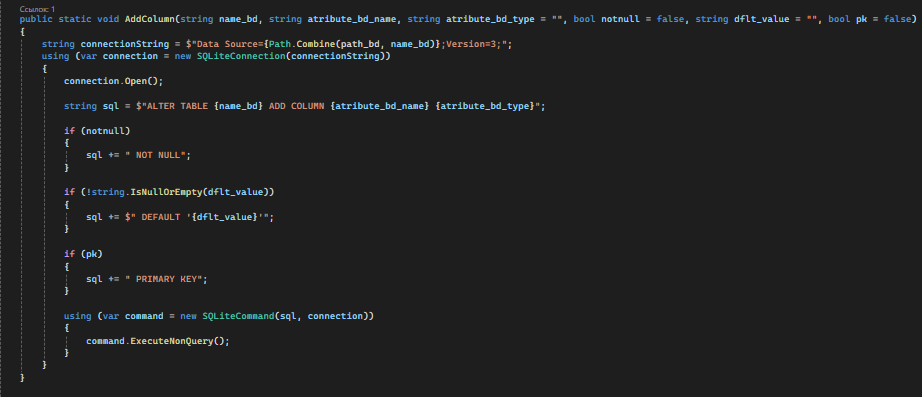


Рисунок 14 - Удаление сущности базы данных (Редактор):

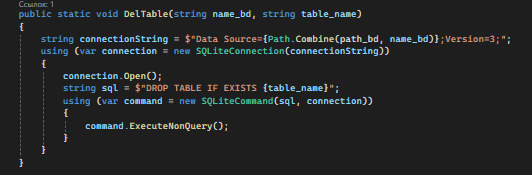
1. Модифицирование сущностей (Редактор):
   * Модификация сущности:

public static void AlterAttributeType(string dbName, string tableName, string attributeName, string newType)  
 {  
 string connectionString = $"Data Source={Path.Combine(path\_bd, dbName)};Version=3;";  
 using (var connection = new SQLiteConnection(connectionString))  
 {  
 connection.Open();  
  
 // Create a new table with the updated column  
 string createTempTableSql = $"CREATE TABLE temp\_\_{tableName} AS SELECT \* FROM {tableName}";  
 using (var createTempTableCommand = new SQLiteCommand(createTempTableSql, connection))  
 {  
 createTempTableCommand.ExecuteNonQuery();  
 }  
  
 // Drop the old column  
 string dropColumnSql = $"ALTER TABLE temp\_\_{tableName} DROP COLUMN {attributeName}";

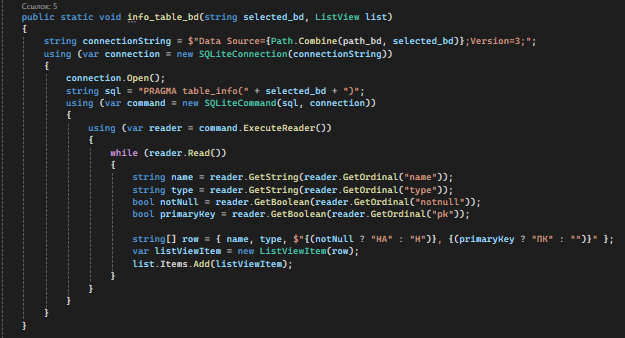
1. Добавление нового атрибута сущности в базу данных (Редактор):
   * Добавление атрибута:



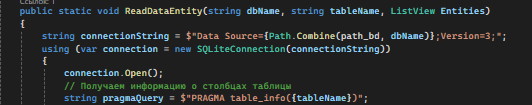
1. Удаление атрибутов сущности базы данных (Редактор):
   * Удаление атрибута:



1. Вывод имен сущностей в ListView (АИС):
   * Получение имен сущностей:

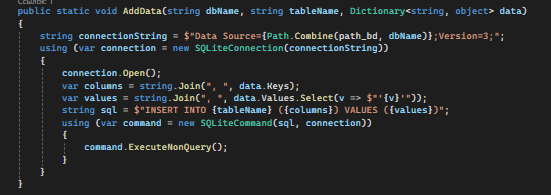


1. Вывод данных в ListView (АИС):
   * Получение данных сущности:

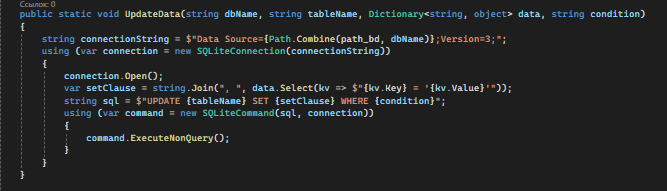


1. Добавление, удаление, обновление (АИС):

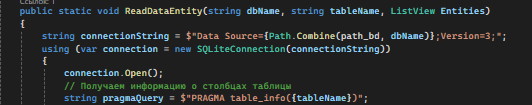
* Добавление данных:



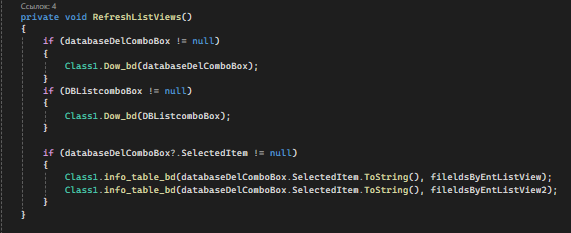
- Обновление данных:



1. Фильтрация данных (АИС):
   * Фильтрация данных:



1. Динамическое обновление:
   * Обновление информации в ListView:

Я

1. Обработка исключений:
   * Обработка ошибок:



1. Оптимизация и комментирование кода:
   * Комментарии и оптимизация:

