МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

БУДІВНИЦТВА І АРХІТЕКТУРИ

Кафедра інформаційних технологій

**Лабораторна робота № 3**

З дисципліни: Інструментальні засоби розробки програмного забезпечення

Виконав:

Студент групи ІСТм-24

Красновський Сергій

Київ-2025

**Мета**: реалізація частини функціоналу мобільного застосунку "Трекер корисних звичок" у середовищі Visual Studio 2022 з використанням .NET MAUI та SQLite.

**Хід** **роботи**

У цій роботі реалізовано базовий функціонал додавання нової звички (назва, частота) та збереження її в локальній базі даних SQLite. Додатково виконано налагодження коду за допомогою інструментів Visual Studio, включаючи точки зупинки, покрокове виконання, зміну значень змінних і розрахунок часу виконання.

**1. Реалізація функціоналу додавання звичок**

Для реалізації функціоналу додавання звичок виконано наступні кроки:

Створено клас **Transcript** для представлення запису в локальній базі даних.

Клас містить основні властивості: ідентифікатор, назву, дату створення, тривалість і шлях до збереженого аудіофайлу.

**Код:**

using SQLite;

using System;

namespace EchoNote.Models

{

public class Transcript

{

[PrimaryKey, AutoIncrement]

public int Id { get; set; }

public string Title { get; set; }

public DateTime CreatedAt { get; set; }

public double DurationSeconds { get; set; }

public string AudioPath { get; set; }

}

}

### 1.2. Налаштування SQLite

Створено клас **DatabaseService** для роботи з локальною базою **SQLite**.  
 Сервіс відповідає за ініціалізацію бази, створення таблиці Transcript та основні операції збереження і читання даних.

**Код:**

using EchoNote.Models;

using SQLite;

using System;

using System.Collections.Generic;

using System.IO;

using System.Threading.Tasks;

namespace EchoNote.Services

{

public class DatabaseService

{

private readonly SQLiteAsyncConnection \_database;

public DatabaseService()

{

var dbPath = Path.Combine(

Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData),

"echonote.db3");

\_database = new SQLiteAsyncConnection(dbPath);

\_database.CreateTableAsync<Transcript>().Wait();

}

public Task<int> AddTranscriptAsync(Transcript t)

{

return \_database.InsertAsync(t);

}

public Task<List<Transcript>> GetTranscriptsAsync()

{

return \_database.Table<Transcript>().ToListAsync();

}

}

}

### 1.3. Інтерфейс користувача

Модифіковано **MainWindow.xaml** для створення інтерфейсу введення назви запису та керування процесом запису.  
 Додано текстове поле для введення назви, кнопки **Start Recording** та **Stop & Save**, а також список для відображення збережених транскриптів.

У файлі **MainWindow.xaml.cs** реалізовано логіку:

* запуск і зупинка запису (імітація),
* обчислення тривалості запису за допомогою Stopwatch,
* створення об’єкта Transcript та виклик методу AddTranscriptAsync для збереження даних у локальну базу SQLite.

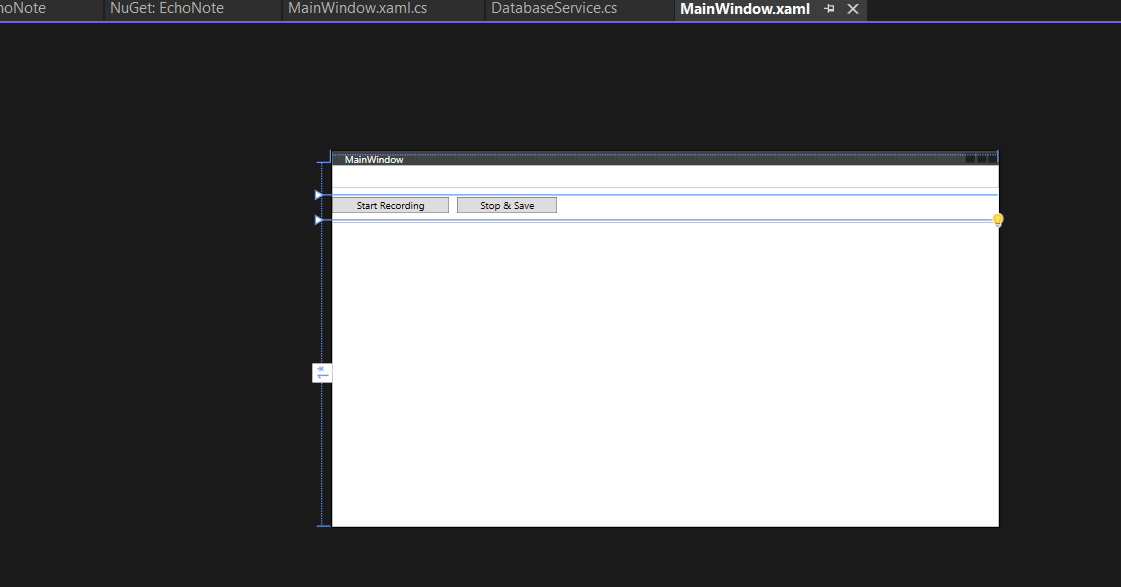
****

Рисунок 3.1 – Інтерфейс програми

### 2. Використання інструментів налагодження

Для забезпечення якості коду використано стандартні інструменти налагодження Visual Studio:

* **Точка зупинки** Встановлено точку зупинки в методі **StopRecording\_Click** на рядку  
   await \_db.AddTranscriptAsync(t);  
   для перевірки створення об’єкта Transcript і коректності його властивостей.

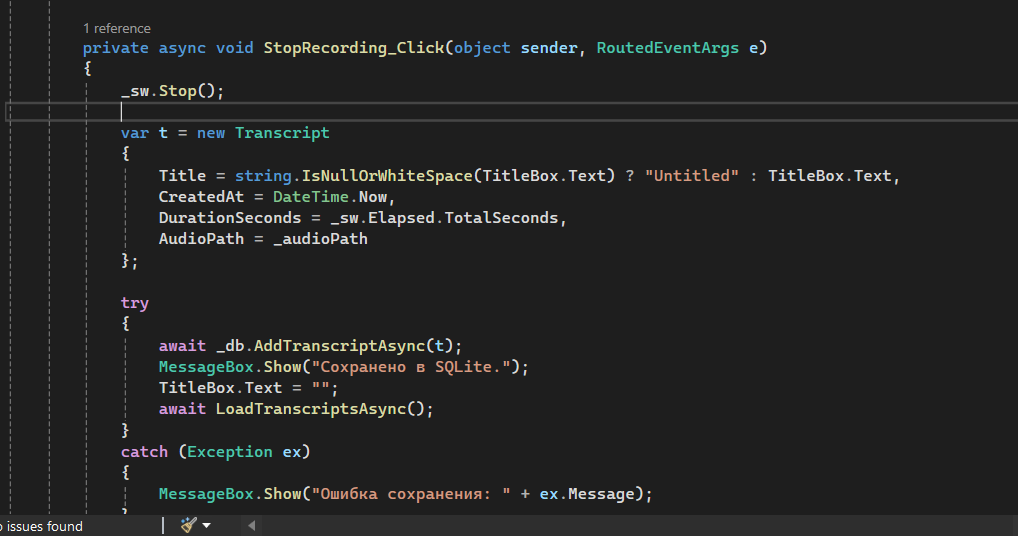


Рисунок 3.2 – Точка зупинки

**Покрокове виконання**

Здійснено покрокове виконання (клавіші **F10** – Step Over, **F11** – Step Into) у методах **StartRecording\_Click** та **StopRecording\_Click**.

Це дозволило простежити послідовність дій: запуск секундоміра, створення об’єкта Transcript, збереження в базу даних.

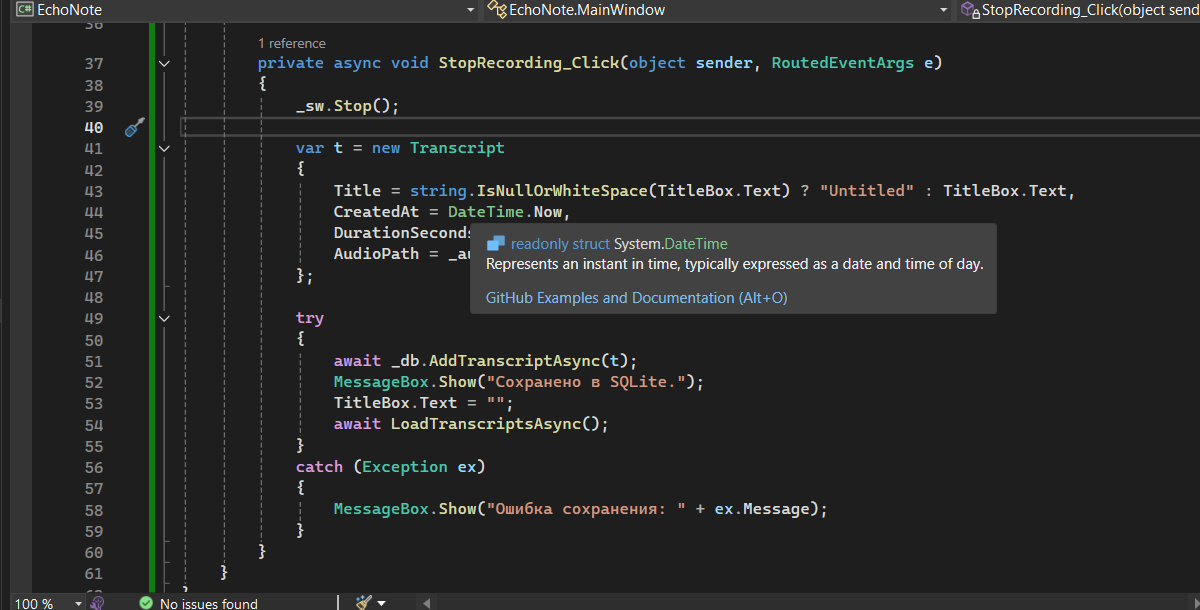


Рисунок 3.3 – Покрокове виконання

**Умовна точка зупинки**

Для тестування довгих назв транскриптів додано умовну точку зупинки в методі **StopRecording\_Click** на тому ж рядку з умовою TitleBox.Text.Length > 10.

Завдяки цьому програма призупинялась лише для довгих назв, що дозволило протестувати різні сценарії введення.

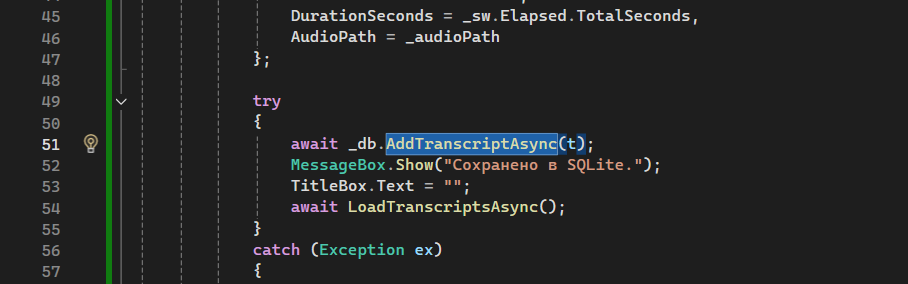


Рисунок 3.4 – Умовна точка зупинки

Після успішного налагодження і тестування зміни було закомічено до репозиторію на GitHub із повідомленням

**«Implemented transcript saving with SQLite and debugging improvements»**.

(Рисунок 3.5 – Виконання git commit)

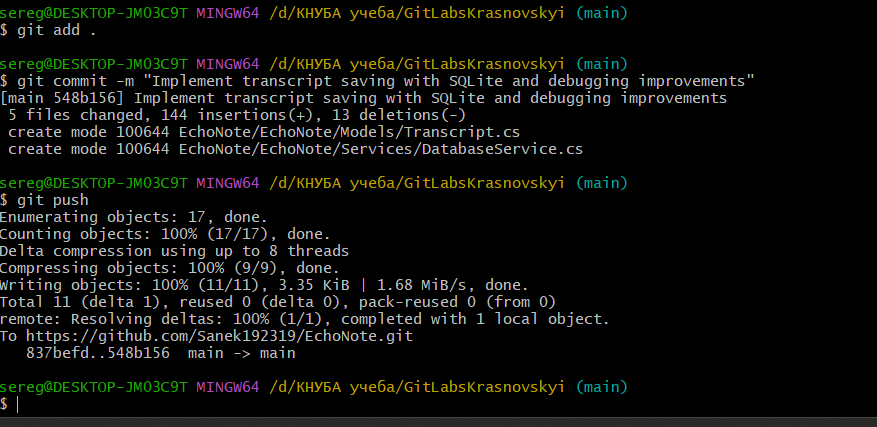


Рисунок 3.5 – Виконання git commit

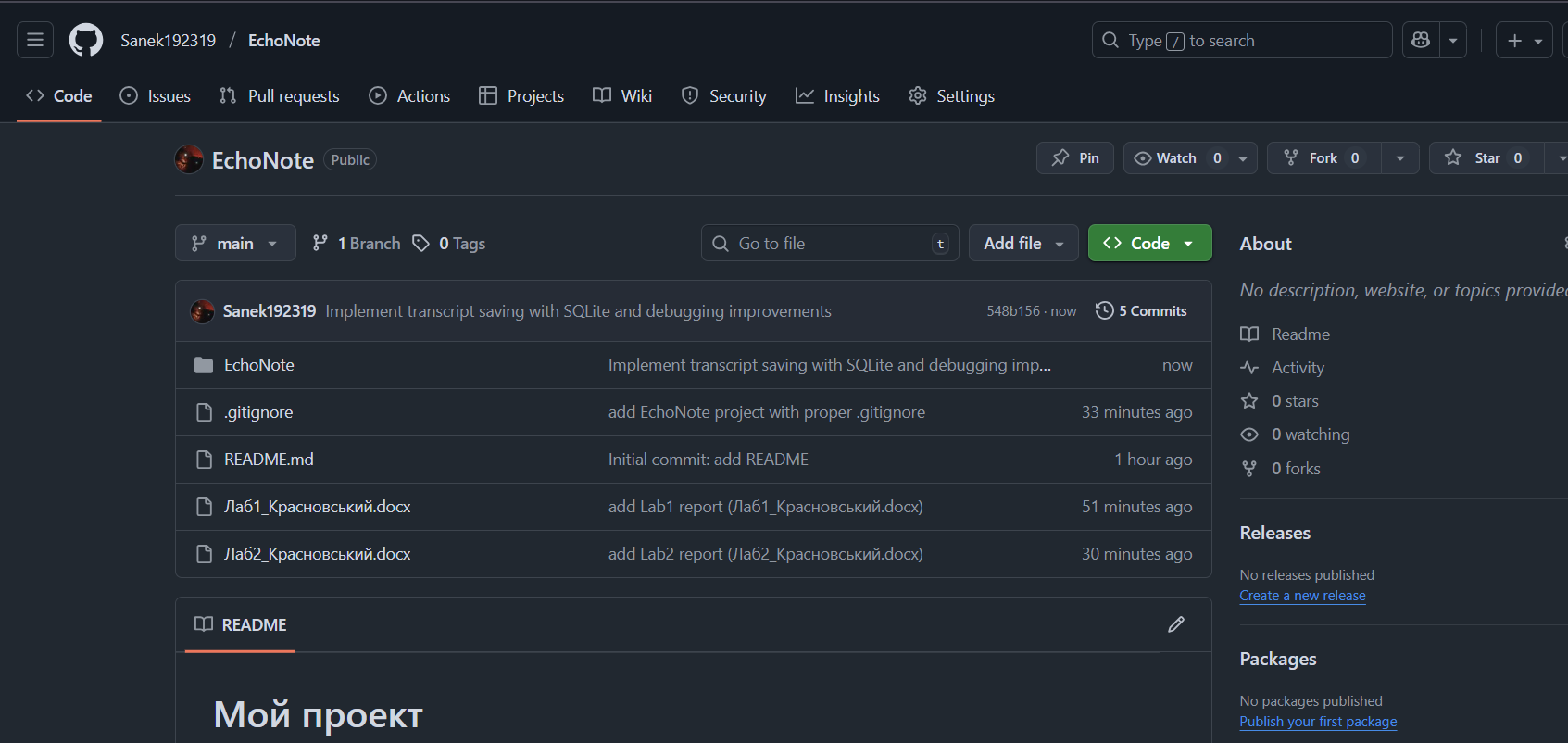


Рисунок 3.6 – Оновлений репозиторій на GitHub

**Посилання на репозиторій:** <https://github.com/Sanek192319/EchoNote>

**Висновок**

У ході виконання лабораторної роботи №3 було реалізовано та відлагоджено ключову частину функціоналу настільного застосунку **EchoNote** – модуль збереження транскриптів у локальну базу даних **SQLite**. Для цього створено клас **Transcript** (модель даних) і сервіс **DatabaseService** для CRUD-операцій, а також модифіковано інтерфейс користувача у **MainWindow.xaml**, додавши елементи для введення назви запису та керування процесом запису.

Під час налагодження застосовано основні інструменти **Visual Studio**:

* встановлено звичайні та умовні **точки зупинки** на рядку await \_db.AddTranscriptAsync(t); для перевірки правильності збереження;
* виконано **покрокове виконання** (Step Over/Step Into) для детального аналізу створення об’єкта Transcript;
* за допомогою **Watch** переглянуто й тимчасово змінено значення змінних, перевірено різні сценарії роботи програми.

Виявлені під час тестування помилки (наприклад, конфлікт атрибутів [PrimaryKey]) були усунуті шляхом уточнення простору імен та додавання значень за замовчуванням для властивостей. Усі зміни закомічено й завантажено до репозиторію **EchoNote** на GitHub.

У результаті отримано робочий компонент програми для створення й зберігання транскриптів, відпрацьовано практичні навички налагодження та забезпечено основу для подальшої розробки повноцінного застосунку EchoNote.