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

Йовенко Дмитро

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:

**1.1. Створення моделі даних**

Створено клас Habit для представлення звички в базі даних. Клас включає основні властивості: ідентифікатор, назву та частоту виконання (наприклад, щоденно або щотижня).

**Код:**

using SQLite;

namespace HabitTracker.Models

{

public class Habit

{

[PrimaryKey, AutoIncrement]

public int Id { get; set; }

public string Name { get; set; }

public string Frequency { get; set; }

public DateTime CreatedDate { get; set; }

}

}

**1.2. Налаштування SQLite**

Створено клас DatabaseService для роботи з базою даних SQLite. Цей клас відповідає за ініціалізацію бази даних і операції CRUD (Create, Read, Update, Delete).

**Код:**

using HabitTracker.Models;

using SQLite;

using System;

using System.IO;

using System.Threading.Tasks;

namespace HabitTracker.Services

{

public class DatabaseService

{

private readonly SQLiteAsyncConnection \_database;

public DatabaseService()

{

var dbPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "habits.db3");

\_database = new SQLiteAsyncConnection(dbPath);

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

}

public Task<int> AddHabitAsync(Habit habit)

{

return \_database.InsertAsync(habit);

}

public Task<List<Habit>> GetHabitsAsync()

{

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

}

}

}

**1.3. Інтерфейс користувача**

Модифіковано MainPage.xaml для додавання інтерфейсу створення звички. Логіка обробки натискання кнопки додавання звички реалізовано в MainPage.xaml.cs На рисунку 3.1 зображено інтерфейс програми на системі Android.

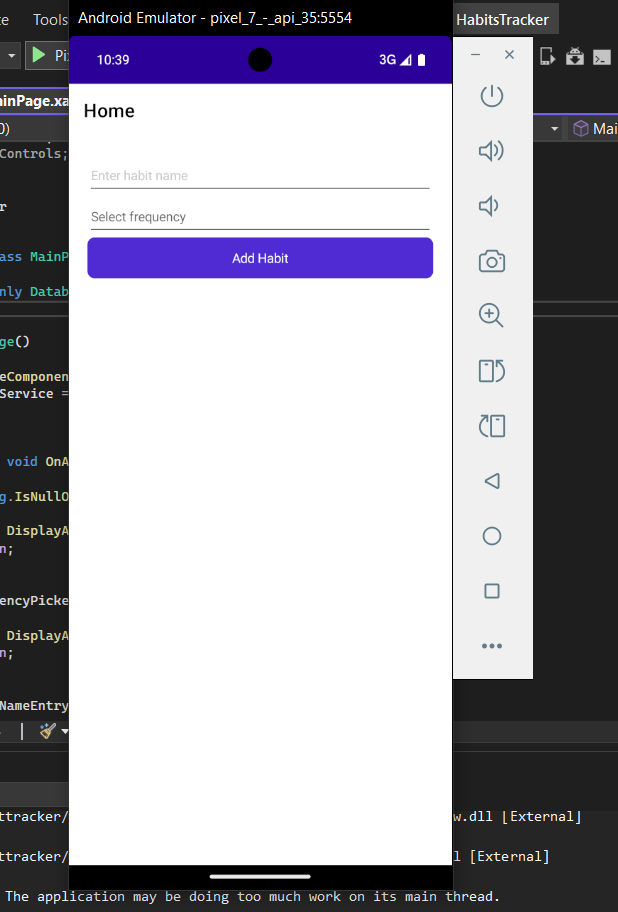


Рисунок 3.1 – Інтерфейс програми

**2. Використання інструментів налагодження**

Для забезпечення якості коду виконано налагодження за допомогою інструментів Visual Studio:

Встановлено точку зупинки в методі OnAddHabitClicked на рядку if (string.IsNullOrWhiteSpace(HabitNameEntry.Text)) для перевірки коректності введених даних(рис 3.2).

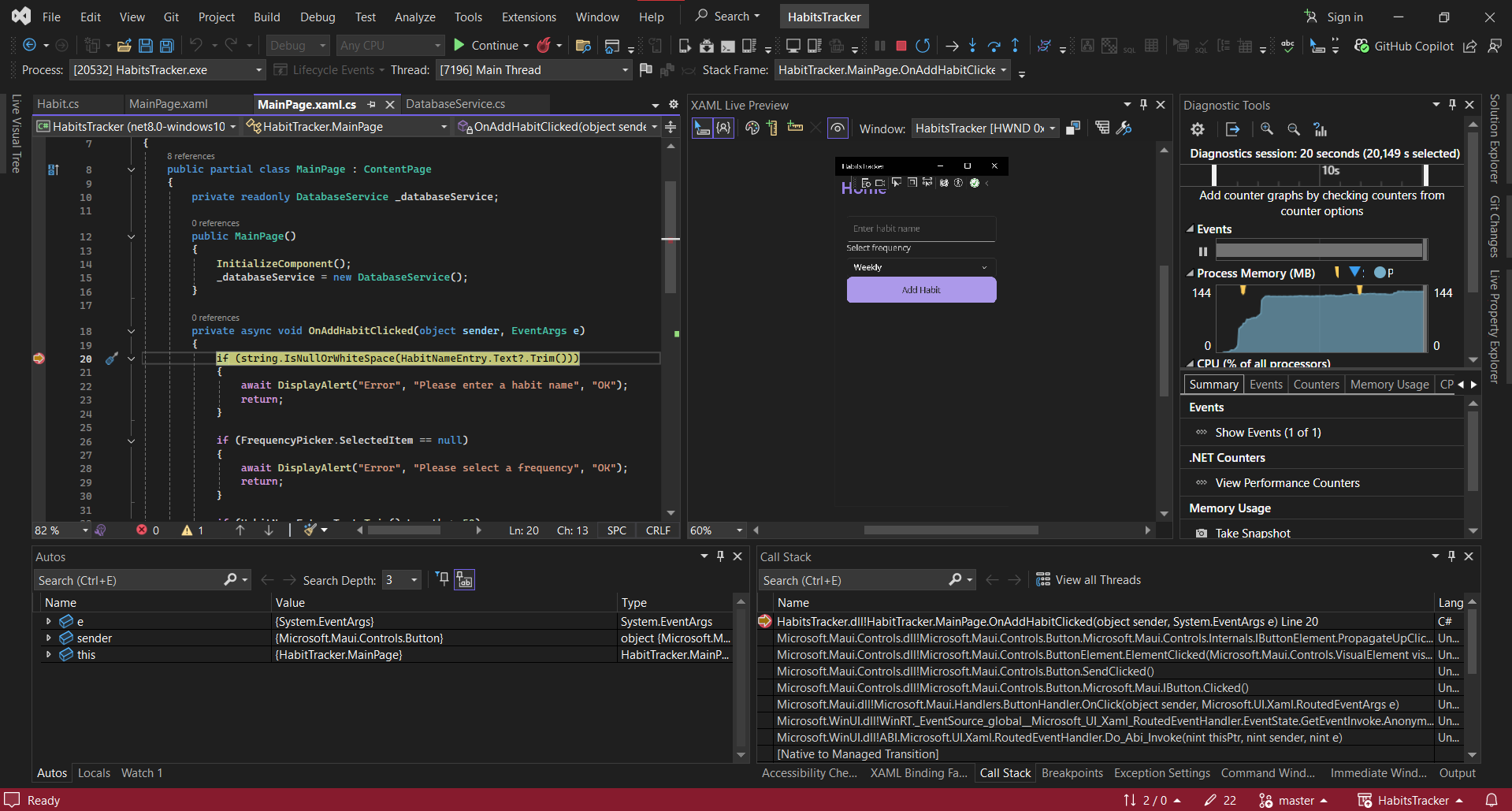


Рисунок 3.2 – Точка зупики

Далі було використано покрокове виконання (F10 для Step Over, F11 для Step Into) в методі OnAddHabitClicked для відстеження створення об’єкта Habit(рис 3.3).

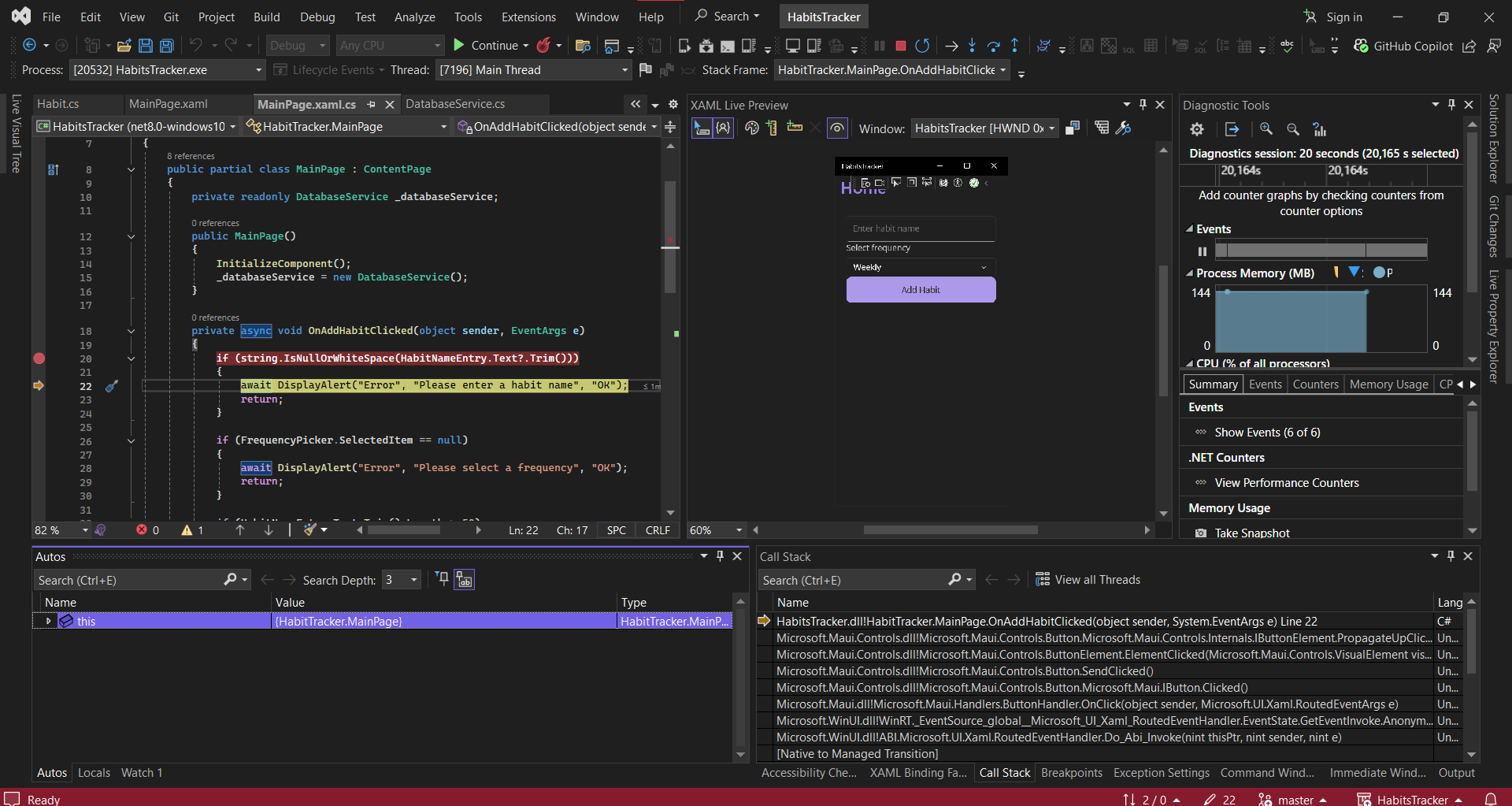


Рисунок 3.3 – Покровкове виконання

Встановлено умовну точку зупинки в методі OnAddHabitClick на рядку var Habi = new Habit() t з умовою NameLength > 5 для перевірки довгих назв звичок(рис. 3.4).

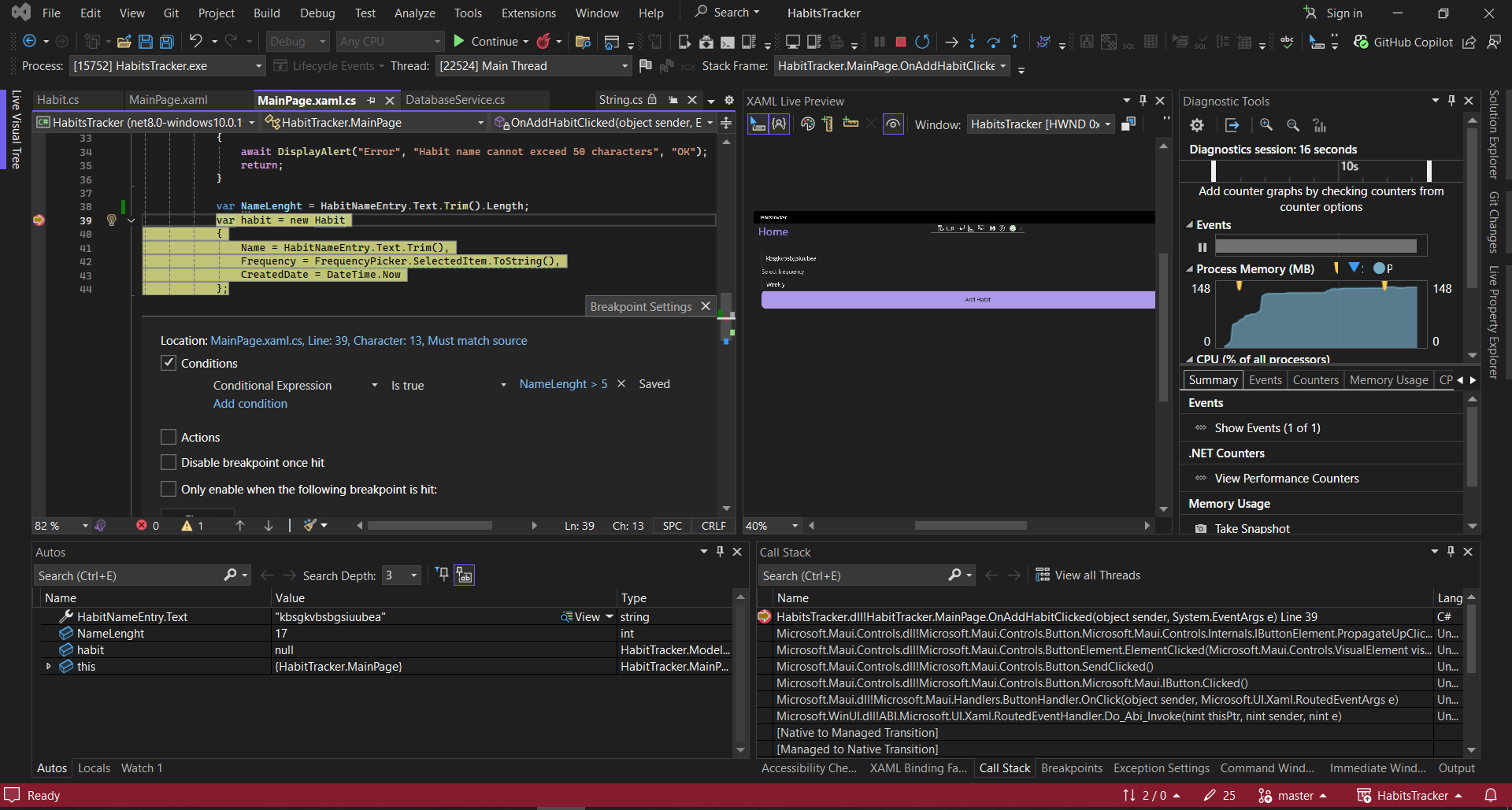


Рисунок 3.4 – Умовна точка зупинки

Після реалізації та налагодження зміни закомічено до репозиторію на GitHub з повідомленням «Added habit creation functionality with SQLite integration and debugging fixes»(рис.3.5).

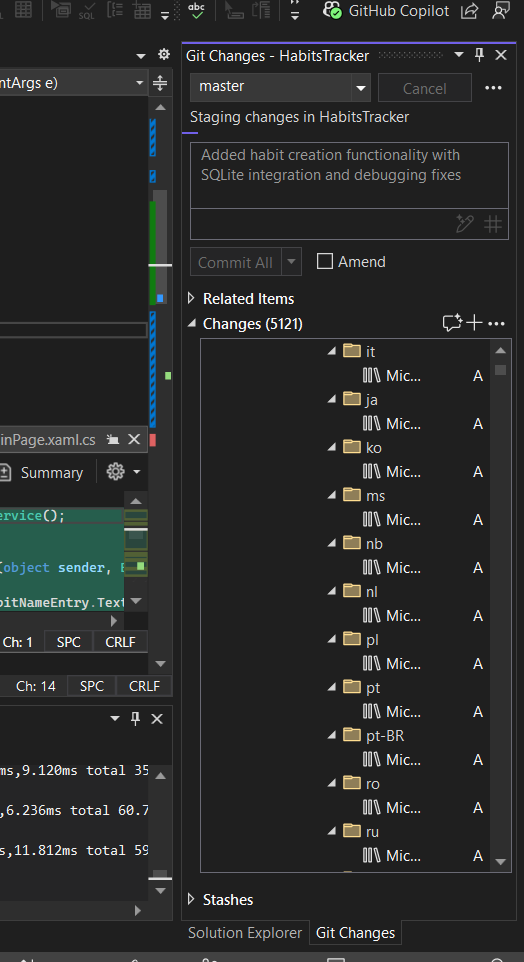


Рисунок 3.5 – Виконання git commit

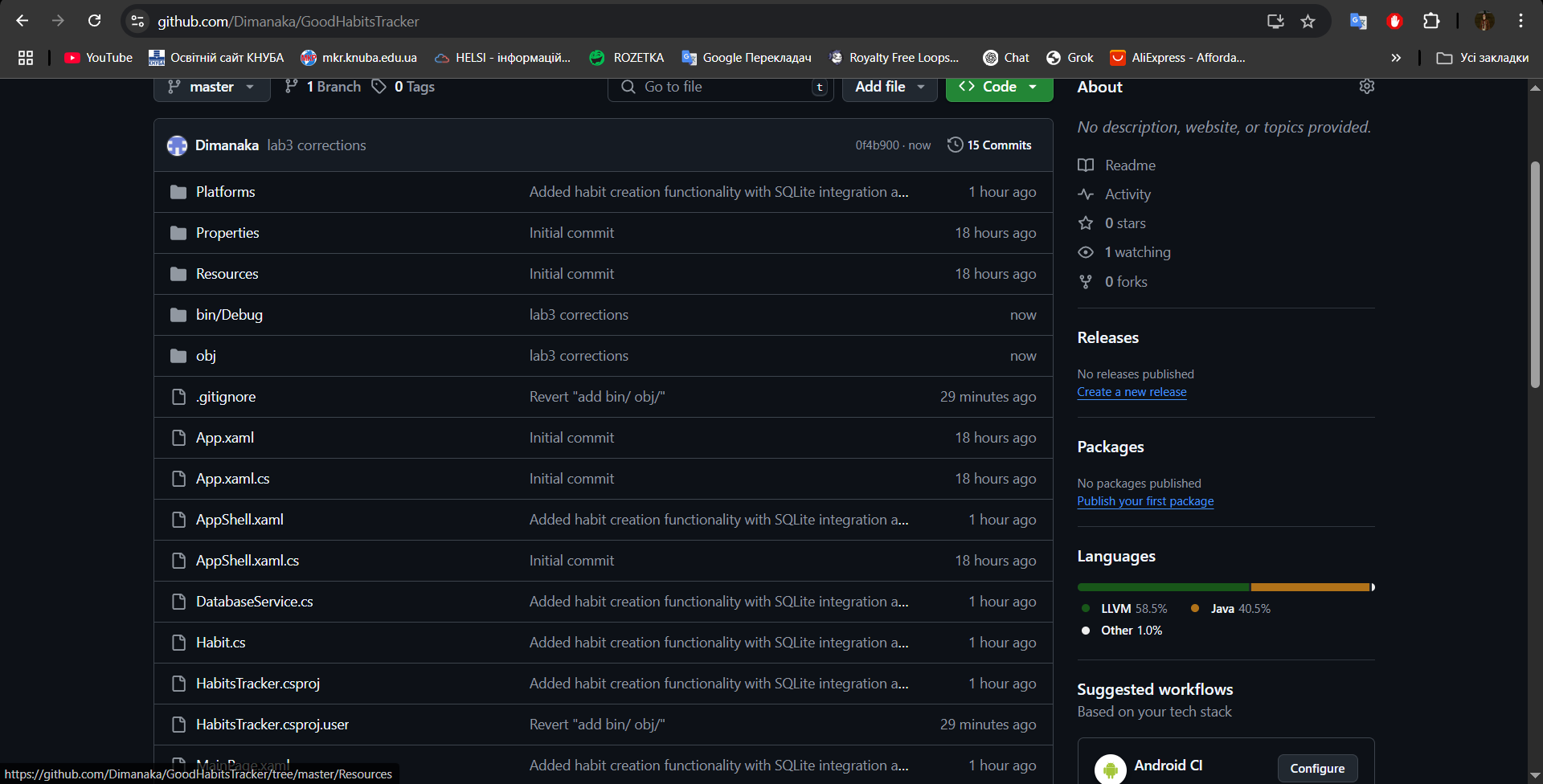


Рисунок 3.6 – Оновлений репозиторій на GitHub

**Посилання на репозиторій:** <https://github.com/Dimanaka/GoodHabitsTracker>

**Висновок**

У результаті виконання лабораторної роботи реалізовано базовий функціонал додавання звичок для мобільного застосунку "Трекер корисних звичок". Створено модель даних Habit для представлення звичок, клас DatabaseService для роботи з локальною базою даних, а також інтерфейс користувача для введення назви та частоти звички. Функціонал протестовано на Android-емуляторі

Для забезпечення якості коду використано інструменти налагодження Visual Studio, зокрема точки зупинки, умовні точки зупинки та покрокове виконання, що дозволило виявити та виправити потенційні помилки, такі як некоректна обробка введених даних. Зміни успішно закомічено до GitHub-репозиторію, що забезпечує належне управління версіями проєкту.

Виконана робота закладає основу для подальшого розвитку застосунку, зокрема для реалізації функцій відображення списку звичок, push-нотифікацій та статистики прогресу.