

Unidad 5 - Actividad 2:

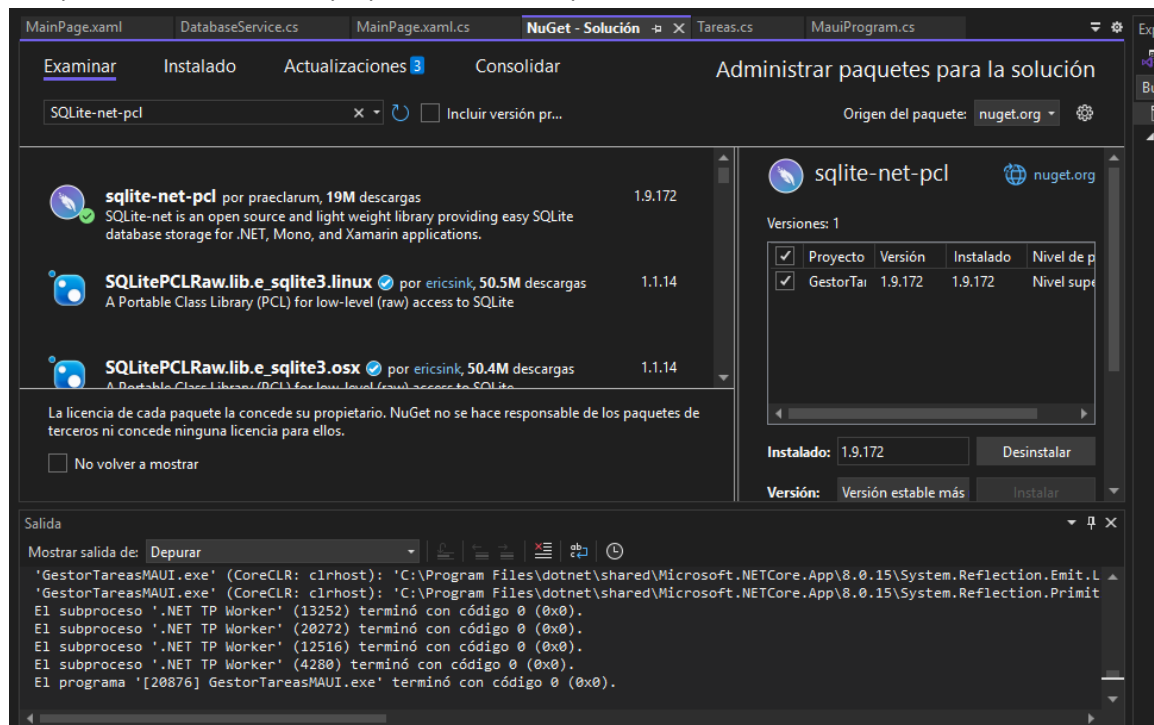
Conexión a base de datos SQLite

Pasos realizados

1. Instalación del paquete SQLite-net-pcl desde NuGet en el proyecto MAUI.
2. Creación de la clase modelo (Tarea.cs) con atributos decorados con SQLite.
3. Creación de una clase DatabaseService para manejar la conexión, inserción, listado y eliminación de tareas.
4. Modificación de la interfaz para incluir botones que interactúan con la base de datos.
5. Pruebas de inserción, visualización y eliminación de tareas en la base de datos.
6. Verificación de que los datos persisten al cerrar y abrir la aplicación.

Evidencias

- Captura de NuGet con el paquete SQLite-net-pcl instalado.



- Vista del código del modelo de datos (Tarea.cs).

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SQLite;

namespace GestorTareasMAUI
{
    [Table("tareass")]
    public class Tarea
    {
        [PrimaryKey, AutoIncrement]
        public int Id { get; set; }

        [MaxLength(200), NotNull]
        public string Texto { get; set; }

        public bool Completada { get; set; }

        public DateTime FechaCreacion { get; set; }

        public Tarea()
        {
            FechaCreacion = DateTime.Now;
        }
    }
}

```

- Código de la clase DatabaseService.cs con operaciones CRUD.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SQLite;

namespace GestorTareasMAUI
{
    public class DatabaseService
    {
        private SQLiteAsyncConnection _database;

        public DatabaseService()
        {
        }

        private async Task Init()
        {
            if (_database is not null)
                return;

            // Crear la base de datos en la carpeta de la app
            var databasePath = Path.Combine(FileSystem.AppDataDirectory, "Tareas.db");
            _database = new SQLiteAsyncConnection(databasePath);

            // Crear la tabla si no existe
            await _database.CreateTableAsync<Tarea>();

            // Obtener todas las tareas
            public async Task<List<Tarea>> GetTareasAsync()
            {
                await Init();
                return await _database.Table<Tarea>().OrderByDescending(t => t.FechaCreacion).ToListAsync();
            }

            // Guardar una tarea nueva
            public async Task<int> SaveTareaAsync(Tarea tarea)
            {
                await Init();

                if (tarea.Id != 0)
                {
                    // Actualizar tarea existente
                    return await _database.UpdateAsync(tarea);
                }
                else
                {
                    // Insertar nueva tarea
                    return await _database.InsertAsync(tarea);
                }
            }

            // Eliminar una tarea
            public async Task<int> DeleteTareaAsync(Tarea tarea)
            {
                await Init();
                return await _database.DeleteAsync(tarea);
            }

            // Marcar tarea como completada
            public async Task<int> ToggleCompletadaAsync(Tarea tarea)
            {
                await Init();
                tarea.Completada = !tarea.Completada;
                return await _database.UpdateAsync(tarea);
            }
        }
    }
}

```

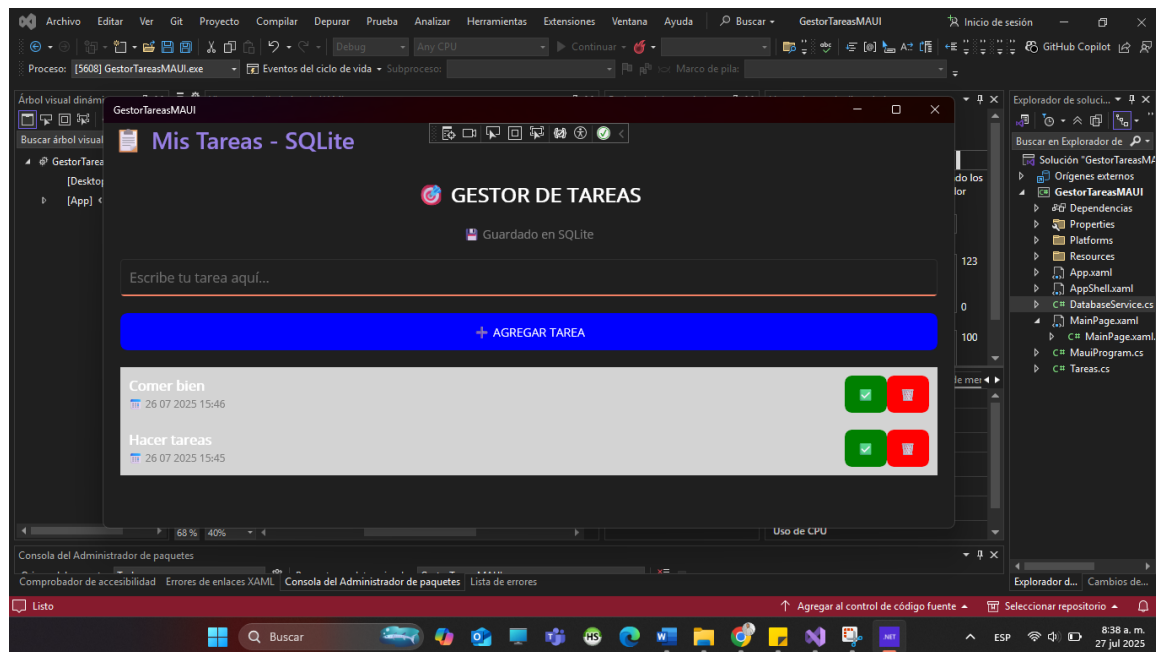
```

            // Marcar tarea como completada
            public async Task<int> ToggleCompletadaAsync(Tarea tarea)
            {
                await Init();
                tarea.Completada = !tarea.Completada;
                return await _database.UpdateAsync(tarea);
            }
        }
    }
}

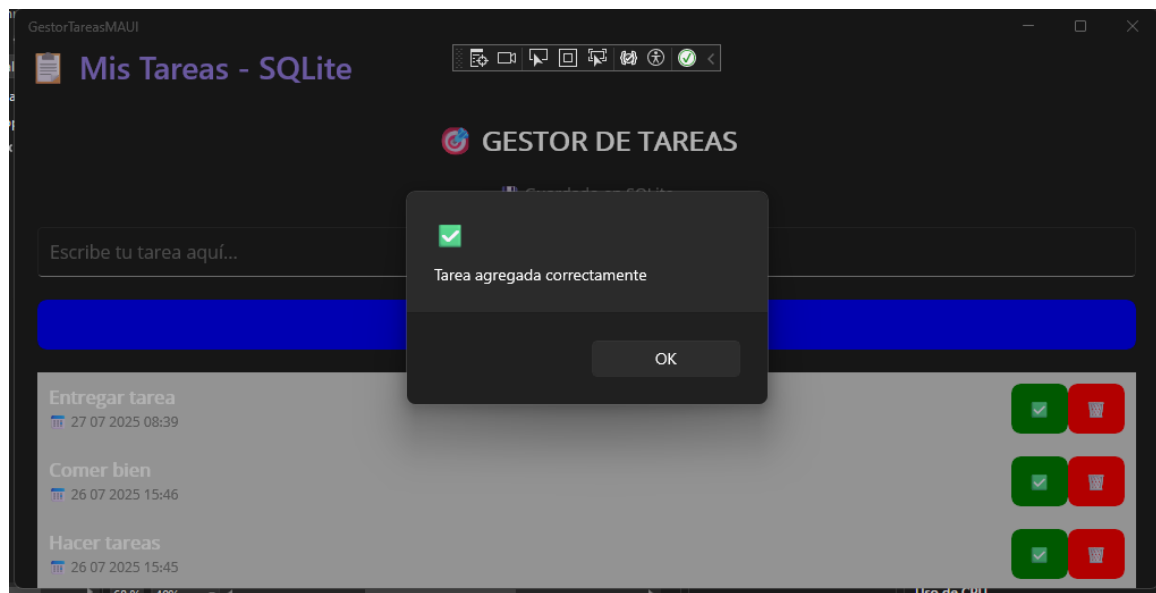
```

- Captura de la app en funcionamiento con tareas agregadas y eliminadas.

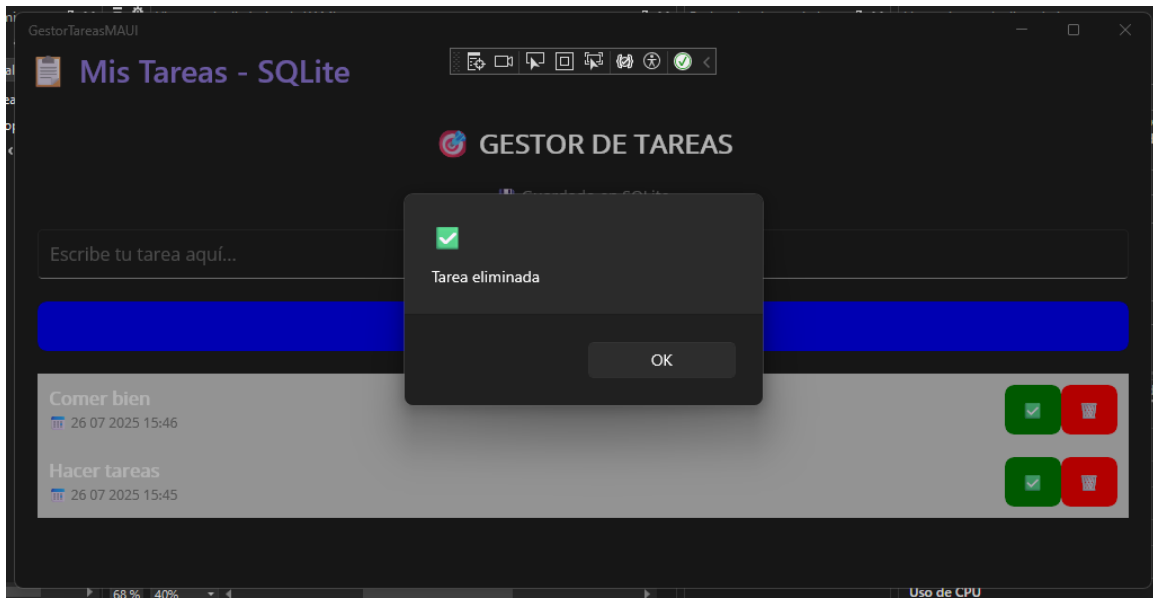
EJECUCIÓN DE APP:



FUNCIÓN AGREGAR TAREA:



FUNCIÓN ELIMINAR TAREA:



TACHAR TAREA:

