



Jorge Quintana González

Introducción

Vamos a hacer un trabajo en el módulo Diseño de interfaces (DI) en el cual tendremos que aprender cómo interactúa MAUI con la base de datos sqllite

objetivo

Tendremos que utilizar la herramienta Microsoft Visual Studio para completar los apartados del ejercicio.

Actividad

mainPage.xaml

Primero, defino esta página como una `ContentPage` en XAML, utilizando `.NET MAUI`. Esto me permite crear interfaces de usuario de forma declarativa. Para estructurar el contenido, utilizo un `ScrollView` que contiene un `VerticalStackLayout`, lo cual asegura que los elementos se muestren de arriba a abajo con espacio entre ellos, y que puedan desplazarse verticalmente si no caben en la pantalla.

El primer elemento dentro del `VerticalStackLayout` es un `Label` con el texto "Empleados". Esto actúa como título principal de la página, indicándole al usuario que la sección está destinada a gestionar la información de empleados. A continuación, agrego un `CollectionView` que se enlaza a la colección `OcTrabajadores` en el contexto de datos. El `CollectionView` permite mostrar la lista de empleados de forma vertical, y su propiedad `SelectionChanged` está configurada para ejecutar el método `OnTrabajadorSelected` cuando

se selecciona un empleado. Esto me permite capturar y manipular los datos del empleado seleccionado.

Dentro del CollectionView, uso un DataTemplate para definir cómo se muestra cada elemento de la lista de empleados. Para cada empleado, configuro un StackLayout horizontal con dos etiquetas (Label): una para el nombre (Nombre) y otra para los apellidos (Apellidos). Este diseño muestra el nombre y apellidos del empleado en una sola línea y con formato simple.

Después de la lista de empleados, agrego dos campos de entrada (Entry), uno para el nombre y otro para el apellido del trabajador. Los Entry tienen Placeholder que indican el tipo de dato que deben contener y están vinculados a las propiedades Nombre y Apellidos, permitiendo así que estos datos se capturen y se muestren cuando se selecciona un empleado o se añade uno nuevo.

Para finalizar, incluyo tres botones: "Añadir Trabajador", "Actualizar Trabajador" y "Eliminar Trabajador". Cada botón está configurado con un evento Clicked que llama a un método diferente en el código subyacente (code-behind). Estos botones permiten agregar, actualizar y eliminar empleados de la lista, lo que proporciona al usuario una forma sencilla de gestionar la información del personal desde una única pantalla.

En conjunto, esta interfaz de usuario está diseñada para simplificar la administración de empleados, permitiendo agregar, ver, actualizar y eliminar empleados de manera intuitiva y eficiente.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="SQLite03.MainPage">

    <ScrollView>
        <VerticalStackLayout
            Spacing="25"
            Padding="30,0"
            VerticalOptions="Start">
            <Label FontSize="Header"
                Text="Empleados" />
            <CollectionView ItemsSource="{Binding OcTrabajadores}"
                SelectionChanged="OnTrabajadorSelected"
                SelectionMode="Single">
                <CollectionView.ItemTemplate>
                    <DataTemplate>
                        <StackLayout Orientation="Horizontal"
                            Padding="10">
                            <!-- Muestra nombre -->
                            <Label Text="{Binding Nombre}"
                                FontSize="Medium"
```

```

        VerticalOptions="Center" />
        <!-- Muestra apellidos -->
        <Label Text="{Binding Apellidos}"
            FontSize="Medium"
            VerticalOptions="Center"
            Margin="10,0,0,0" />

    </StackLayout>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
<Entry Placeholder="Nombre del trabajador" x:Name="entryNombre" Text="{Binding
Nombre}"></Entry>
<Entry Placeholder="Apellido del trabajador" x:Name="entryApellido" Text="{Binding
Apellidos}"></Entry>
<Button Text="Añadir Trabajador" Clicked="OnAddTrabajadorClicked" />
<Button Text="Actualizar Trabajador" Clicked="OnUpdateTrabajadorClicked"/>
<Button Text="Eliminar Trabajador" Clicked="OnDeleteTrabajadorClicked"/>

</VerticalStackLayout>
</ScrollView>

</ContentPage>

```

mainPage.xaml.cs

Primero, este código comienza con la declaración de varios using, para poder acceder a funcionalidades clave de C#, como las colecciones (System.Collections.ObjectModel), el soporte para notificación de cambios de propiedades (System.ComponentModel), y la conexión a SQLite (System.Data.SQLite). Todo el código se organiza en el espacio de nombres SQLite03.

La clase principal MainPage hereda de ContentPage e implementa INotifyPropertyChanged para que la interfaz de usuario actualice automáticamente cuando cambian las propiedades enlazadas. Dentro de esta clase, defino una propiedad Nombre, que almacena el nombre del trabajador actual y está configurada para notificar cambios cuando el valor se actualiza. También declaro OcTrabajadores, una colección observable de objetos Trabajador que representa la lista de empleados, permitiendo que la interfaz de usuario refleje automáticamente los cambios.

En el constructor MainPage, inicializo OcTrabajadores y configuro la conexión con la base de datos SQLite. Primero, obtengo la ruta de la base de datos en el sistema de archivos para que esté en el directorio de la aplicación, lo cual facilita el acceso en un entorno de pruebas. Creo la cadena de conexión de SQLite con esta ruta y abro la conexión. A continuación, llamo al método CrearTablaTrabajador para asegurarme de que la tabla de trabajadores exista, y luego inserto algunos datos de ejemplo llamando a

InsertarDatosEjemplo. Después, realizo una consulta `SELECT * FROM Trabajador` para recuperar los trabajadores y los agrego a la colección `OcTrabajadores`.

Para agregar trabajadores nuevos, implemento el método `OnAddTrabajadorClicked`, que se activa al hacer clic en el botón de añadir trabajador en la interfaz. Si los campos de entrada no están vacíos, crea un nuevo objeto `Trabajador`, lo añade a `OcTrabajadores`, y luego limpia los campos de entrada. Esto permite que el usuario agregue nuevos trabajadores de manera interactiva.

El método `OnDeleteTrabajadorClicked` se activa cuando el usuario hace clic en el botón de eliminar trabajador. Este método recorre `OcTrabajadores` y elimina el trabajador cuyos nombres y apellidos coinciden con los valores introducidos en los campos de texto. Al igual que la función de añadir, esto ayuda a mantener la lista de trabajadores actualizada en la interfaz.

Para manejar la selección de un trabajador en la interfaz, defino el método `OnTrabajadorSelected`, que guarda el trabajador seleccionado en `_trabajadorSeleccionado`. También actualiza los campos de texto de nombre y apellido con los datos del trabajador seleccionado, y habilita los botones de actualización y eliminación.

En el método `CrearTablaTrabajador`, verifico y creo la tabla `Trabajador` en la base de datos si no existe, usando un comando SQL `CREATE TABLE IF NOT EXISTS`. Esto asegura que la base de datos esté lista para recibir registros de trabajadores. Además, en `InsertarDatosEjemplo`, inserto algunos trabajadores de ejemplo para que la aplicación tenga datos iniciales.

Por último, `EjecutarNonQuery` se utiliza como método auxiliar para ejecutar consultas SQL sin resultados de retorno, como las instrucciones `INSERT` o `CREATE TABLE`.

```
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Data.SQLite;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace SQLite03
{
    public partial class MainPage : ContentPage, INotifyPropertyChanged
    {
        private string _nombre;
        private Trabajador _trabajadorSeleccionado;

        public string Nombre
        {
            get => _nombre;
            set
```

```

    {
        if (_nombre != value)
        {
            _nombre = value;
            OnPropertyChanged(nameof(Nombre));
        }
    }
}

private ObservableCollection<Trabajador> _ocTrabajadores;
public ObservableCollection<Trabajador> OcTrabajadores
{
    get { return _ocTrabajadores; }
    set
    {
        _ocTrabajadores = value;
        OnPropertyChanged();
    }
}

public MainPage()
{
    InitializeComponent();

    OcTrabajadores = new ObservableCollection<Trabajador>();

```

```

"D:\Datos\proyectos_DI2425\ud04part01ExempleSQLite\SQLite03\bin\Debug\net8.0-windows10.0.19041.0\win10-x64\AppX\"

```

```

    string rutaDirectorioApp = System.AppContext.BaseDirectory;

```

```

    DirectoryInfo directorioApp = new DirectoryInfo(rutaDirectorioApp);

```

```

    string connectionString = $"Data Source={databasePath};Version=3;";

```

```

    using (SQLiteConnection connection = new SQLiteConnection(connectionString))
    {

```

```

        connection.Open();
        CrearTablaTrabajador(connection);

```

```

        InsertarDatosEjemplo(connection);

```

```

        string sql = "SELECT * FROM Trabajador";
        SQLiteCommand command = new SQLiteCommand(sql, connection);
        SQLiteDataReader reader = command.ExecuteReader();

```

```

        while (reader.Read())
        {
            int idTrabajador = reader.GetInt32(0);

```

```

        string nombreTrabajador = reader.GetString(1);
        string apellidosTrabajador = reader.GetString(2);
        Trabajador trabajador = new Trabajador
        {
            Id = idTrabajador,
            Nombre = nombreTrabajador,
            Apellidos = apellidosTrabajador,
        };

        OcTrabajadores.Add(trabajador);
    }

    reader.Close();
    connection.Close();
}

BindingContext = this;
}
private void OnAddTrabajadorClicked(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(entryNombre.Text) &&
        !string.IsNullOrEmpty(entryApellido.Text))
    {
        var nuevoTrabajador = new Trabajador
        {
            Nombre = entryNombre.Text,
            Apellidos = entryApellido.Text
        };

        OcTrabajadores.Add(nuevoTrabajador);

        entryNombre.Text = string.Empty;
        entryApellido.Text = string.Empty;
    }
}
private void OnDeleteTrabajadorClicked(object sender, EventArgs e)
{
    for (int i = 0; i < _ocTrabajadores.Count; i++)
    {
        if ((_ocTrabajadores.ElementAt(i).Nombre == entryNombre.Text) &&
            (_ocTrabajadores.ElementAt(i).Apellidos == entryApellido.Text))
        {
            _ocTrabajadores.RemoveAt(i);
        }
    }
}

```

```

    }
}
private void OnTrabajadorSelected(object sender, SelectionChangedEventArgs e)
{
    _trabajadorSeleccionado = e.CurrentSelection.FirstOrDefault() as Trabajador;

    if (_trabajadorSeleccionado != null)
    {
        entryNombre.Text = _trabajadorSeleccionado.Nombre;
        entryApellido.Text = _trabajadorSeleccionado.Apellidos;
        ((Button)FindByName("OnUpdateTrabajadorClicked")).IsEnabled = true;
        ((Button)FindByName("OnDeleteTrabajadorClicked")).IsEnabled = true;
    }
}

private void CrearTablaTrabajador(SQLiteConnection connection)
{
    string queryCrearTablaTrabajador = "CREATE TABLE IF NOT EXISTS Trabajador ("
+
        "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
        "nombre TEXT, " +
        "apellidos TEXT)";
    EjecutarNonQuery(connection, queryCrearTablaTrabajador);
}
private async void OnUpdateTrabajadorClicked(object sender, EventArgs e)
{
    if (_trabajadorSeleccionado != null && !string.IsNullOrEmpty(entryNombre.Text)
    && !string.IsNullOrEmpty(entryApellido.Text))
    {
        _trabajadorSeleccionado.Nombre = entryNombre.Text;
        _trabajadorSeleccionado.Apellidos = entryApellido.Text;

        int index = _ocTrabajadores.IndexOf(_trabajadorSeleccionado);
        _ocTrabajadores[index] = _trabajadorSeleccionado;

        entryNombre.Text = string.Empty;
        entryApellido.Text = string.Empty;

        ((Button)FindByName("OnUpdateTrabajadorClicked")).IsEnabled = false;
        ((Button)FindByName("OnDeleteTrabajadorClicked")).IsEnabled = false;
    }
}
private void InsertarDatosEjemplo(SQLiteConnection connection)
{

```

```



        EjecutarNonQuery(connection, "insert into Trabajador (nombre, apellidos) values
('Ana', 'Gómez')");
        EjecutarNonQuery(connection, "insert into Trabajador (nombre, apellidos) values
('Juan', 'Pérez')");
    }

    private void EjecutarNonQuery(SQLiteConnection connection, string query)
    {

        using (SQLiteCommand command = new SQLiteCommand(query, connection))
        {
            command.ExecuteNonQuery();
        }
    }
}
}

```

Diagrama de Gantt

Tarea	Fecha de Inicio	Fecha de Fin	Duración (días)	Días
Inicio del Proyecto	Miércoles 13	-	-	-
Diseño XAML	Miércoles 13	Sábado 23	10	
Programación en C#	Sábado 23	Lunes 18	4	

Problemas

Creo que no he podido conectar la lista de trabajadores con la base de datos.

Conclusión