



Natalia Coronado Romero 1ºDAM

ÍNDICE

Introducción.....	3
Objetivo de la memoria.....	3
Material utilizado.....	3
GitHub.....	3
Desarrollo.....	4
Interfaz Visual.....	4
-Vista de la aplicación.....	4
Archivo XAML.....	5
Código del CS.....	7
Clase Tasca.....	10
Problemas encontrados y sugerencias.....	11
Conclusión.....	11
Bibliografía/webgrafía.....	11
Video del funcionamiento.....	11

Introducción

Proyecto de una aplicación con MAUI en Visual Studio para crear un CRUD de una base de datos conectada a FireBase.

Objetivo de la memoria

El objetivo de esta práctica es diseñar una aplicación que se sincronice una base de datos de FireBase en la página web de google y hacer modificaciones en él.

Material utilizado

Marca y Modelo del Procesador: Intel Pentium CPU G4400 @ 3.30GHz

Tipo Memoria RAM (memoria y slots): 4GB RAM, 2400 MHz

Tipo de dispositivo de almacenamiento

capacidad (GiB): HDD 1TB

Programas utilizados: Spectacle, Visual Studio 2022, OBS Studio

GitHub

Es es el enlace de github al repositorio de la asignatura:

<https://github.com/MalvaLego/Desarrollo-de-Interfaces.git>

Desarrollo

Interfaz Visual

-Vista de la aplicación

Esta es la vista previa de como se vería la aplicación para el usuario.



Figura 1: Vista de la interfaz de la aplicación en ejecución

Archivo XAML

Este es el código del archivo MainPage.xaml donde se encuentran los elementos visuales y sus características.



```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             x:Class="Tema2Exercici5Segundo.MainPage"
5             BackgroundColor="#ECEFF1"
6             >
7
8     <ScrollView>
9         <VerticalStackLayout
10             Padding="30,0"
11             Spacing="25">
12             <Image
13                 Source="Logo.png"
14                 HeightRequest="120"
15                 WidthRequest="120"
16                 Aspect="AspectFit"
17             />
18
19             <Label
20                 Text="Tareas a relizar"
21                 Style="{StaticResource SubHeadline}"
22                 Margin="0,-10,0,0"
23                 SemanticProperties.HeadingLevel="Level2"></Label>
24
25             <Entry
26                 x:Name="eNombreTarea"
27                 BackgroundColor="White"
28             />
29
30         </VerticalStackLayout>
31     </ScrollView>
32 </ContentPage>
```

Figura 2: Código de los dos Entry junto con el texto en la parte de arriba



```
<HorizontalStackLayout HorizontalOptions="Center" Margin="0,20,0,0">
    <Button
        Clicked="btnAnyadir"
        Text="Añadir"
        Margin="0,0,30,0"
        WidthRequest="150"
        BackgroundColor="#388E3C"
    />
    <Button
        Clicked="btnModificar"
        Text="Modificar"
        Margin="0,0,30,0"
        WidthRequest="150"
        BackgroundColor="#FF9800"
    />
    <Button
        Clicked="btnBorrar"
        Text="Borrar"
        WidthRequest="150"
        BackgroundColor="#D32F2F"
    />
</HorizontalStackLayout>
```

Figura 3: Código de los tres botones de la aplicación

```

60
61
62 <CollectionView ItemsSource="{Binding Tasques}"
63 SelectionMode="Single"
64 SelectionChanged="btnMostrarSeleccion"
65 x:Name="cvTareas"
66 WidthRequest="500"
67 >
68   <CollectionView.ItemTemplate>
69     <DataTemplate>
70       <StackLayout Orientation="Horizontal"
71         Padding="10"
72         WidthRequest="500"
73         BackgroundColor="#d1ede4">
74         <Label Text="{Binding NombreTarea}"
75           FontSize="15"
76           VerticalOptions="Center"
77         />
78
79         <VisualStateManager.VisualStateGroups>
80           <VisualStateGroup Name="CommonStates">
81             <VisualState Name="Normal"></VisualState>
82             <VisualState Name="Selected">
83               <VisualState.Setters>
84                 <Setter Property="BackgroundColor" Value="#f6d49d"></Setter>
85               </VisualState.Setters>
86             </VisualState>
87           </VisualStateGroup>
88         </VisualStateManager.VisualStateGroups>
89       </StackLayout>
90     </DataTemplate>
91   </CollectionView.ItemTemplate>
92 </CollectionView>
93 </VerticalStackLayout>
94 </ScrollView>
95 </ContentPage>
96
97

```

Figura 4: Este es el final del código que incluye el CollectionView de la base de datos de Firebase. Dentro de este está el label del nombre de la Tarea actualizandose con binding con un ObservableCollection.

Código del CS

Aquí se encuentra el código para que pueda funcionar la aplicación y se puede ejecutar, añadiendo también acciones que se quieran realizar en ella.

```
1 using Firebase.Database;
2 using Firebase.Database.Query;
3 using System.Collections.ObjectModel;
4 using System.Diagnostics;
5
6 namespace Tema2Ejercicio5Segundo
7 {
8     public partial class MainPage : ContentPage
9     {
10         FirebaseClient firebaseClient = new FirebaseClient("https://ejercicio5tema2-default-rtdb.europe-west1.firebaseio.com/");
11         public ObservableCollection<Tasca> Tasques { get; set; } = new ObservableCollection<Tasca>();
12         public MainPage()
13         {
14             InitializeComponent();
15             BindingContext = this;
16             SubscribeAFirebase();
17         }
18     }
19 }
```

Figura 5: En el MainPage se ha creado una variable se sincroniza con la base de datos de firebase mediante su URL. También un ObservableCollection que guardará todos los objetos Tasca que haya en el firebase, este ObservableCollection se verá al estar conectado al CollectionView. En el constructor se actualiza el binding y llama a una variable que actualizará también el firebase.

```
17     }
18
19     private void SubscribeAFirebase()
20     {
21         var collection = firebaseClient
22             .Child("Tasques")
23             .AsObservable<Tasca>()
24             .Subscribe((item) =>
25             {
26                 if (item.Object != null)
27                 {
28                     if (item.EventType == Firebase.Database.Streaming.FirebaseEventType.Delete)
29                     {
30                         var deletedItem = Tasques.FirstOrDefault(t => t.IdTarea == item.Key);
31                         if (deletedItem != null)
32                         {
33                             Tasques.Remove(deletedItem);
34                         }
35                     }
36                     else
37                     {
38                         var existingItem = Tasques.FirstOrDefault(t => t.IdTarea == item.Key);
39                         if (existingItem == null)
40                         {
41                             Tasques.Add(new Tasca
42                             {
43                                 NombreTarea = item.Object.NombreTarea,
44                                 IdTarea = item.Key
45                             });
46                         }
47                         else
48                         {
49                             existingItem.NombreTarea = item.Object.NombreTarea;
50                         }
51                     }
52                 }
53             });
54     }
55 }
```

Figura 6: Esta es la función anterior del constructor, que añade todas las tareas que hay en la base de datos de FireFase al ObservableCollection. Así cada vez que hay algún cambio lanza un evento a la base de datos y se actualiza.


```

na2Exercici5Segundo (net8.0-android)  Tema2Exercici5Segundo.MainPage  btnBon
existingItem.NombreTarea = item.Object.NombreTarea;
    }
    }
    });
}
0 referencias
private void btnAnyadir(object sender, EventArgs e)
{
    if (eNombreTarea.Text != null)
    {
        Tasca tasca = new Tasca
        {
            IdTarea = string.Empty,
            NombreTarea = eNombreTarea.Text,
        };

        string key = firebaseClient.Child("Tasques").PostAsync(tasca).Result.Key;
        if (key != null)
        {
            firebaseClient.Child("Tasques").Child(key).PutAsync(
                new Tasca
                {
                    IdTarea = key,
                    NombreTarea = tasca.NombreTarea
                });
        }
        eNombreTarea.Text = string.Empty;
    }
}

```

Figura 7: Al pulsar el botón de añadir llamará a la función `btnAnyadir` la cual creará un nuevo objeto `Tasca` con el nombre puesto en el Entry de la aplicación. Luego esta tasca se añadirá también al FireBase


```

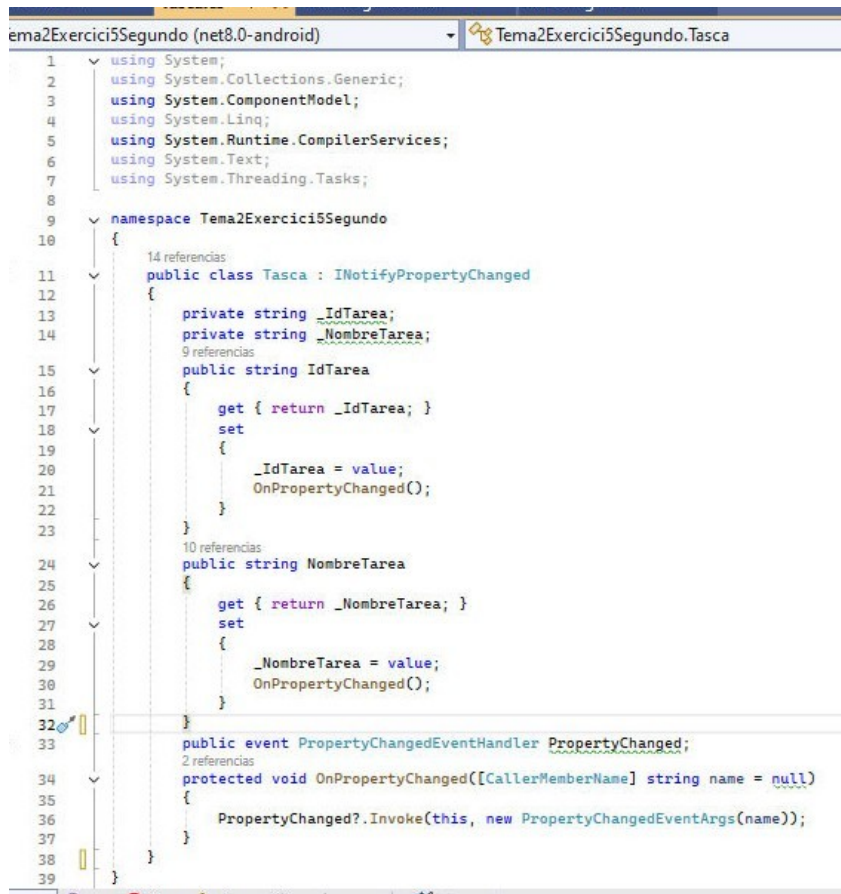
82 private void btnMostrarSeleccion(object sender, EventArgs e)
83 {
84     Tasca itemSeleccionado = cvTareas.SelectedItem as Tasca;
85     if (itemSeleccionado != null)
86     {
87         // Muestra el nom de la tasca seleccionada al Entry
88         eNombreTarea.Text = itemSeleccionado.NombreTarea;
89         Debug.WriteLine($"Tasca seleccionada: {itemSeleccionado.NombreTarea}");
90     }
91 }
92 private void btnModificar(object sender, EventArgs e)
93 {
94     Tasca itemSeleccionado = cvTareas.SelectedItem as Tasca;
95     if (itemSeleccionado != null)
96     {
97         var collection = firebaseClient
98             .Child("Tasques").Child(itemSeleccionado.IdTarea).PutAsync(
99             new Tasca
100             {
101                 IdTarea = itemSeleccionado.IdTarea,
102                 NombreTarea = eNombreTarea.Text
103             });
104         eNombreTarea.Text = string.Empty;
105     }
106 }
107 private void btnBorrar(object sender, EventArgs e)
108 {
109     Tasca itemSeleccionado = cvTareas.SelectedItem as Tasca;
110     if (itemSeleccionado != null)
111     {
112         var collection = firebaseClient
113             //Esborra l'item de Firebase amb la clau itemseleccionado.id
114             .Child("Tasques").Child(itemSeleccionado.IdTarea).DeleteAsync();
115     }
116     eNombreTarea.Text = string.Empty;
117 }
118 }
119
120

```

Figura 8: Estas son las tres últimas funciones. La primera llamada *btnMostrarSección* está conectada al *CollectionView* de forma que cada vez que se selecciona una tarea, se muestra su nombre en el *Entry*. La segunda función enlazada a un botón hace que si está seleccionado un botón y has querido modificar el nombre en el *Entry* este se actualiza directamente. Y en la última llamada *btnBorrar* simplemente elimina la tarea seleccionada

Clase Tasca

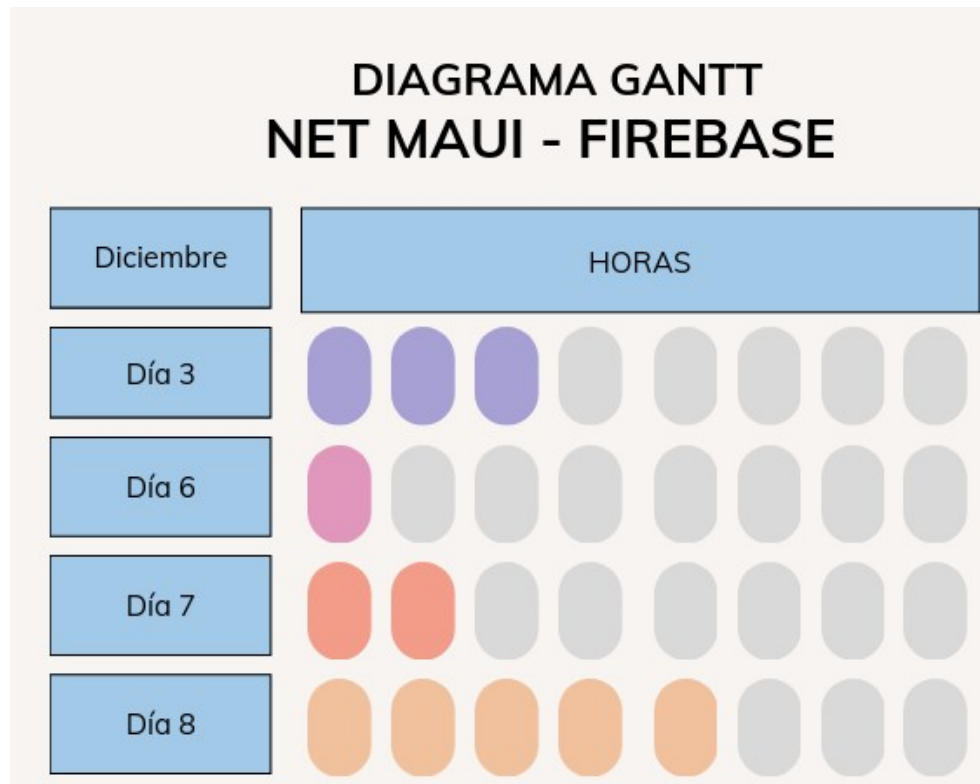
Esta clase no ejecuta nada simplemente crea objetos tipo Tasca con los valores que se indiquen.



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Linq;
5 using System.Runtime.CompilerServices;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace Tema2Exercici5Segundo
10 {
11     14 referencias
12     public class Tasca : INotifyPropertyChanged
13     {
14         private string _IdTarea;
15         private string _NombreTarea;
16         9 referencias
17         public string IdTarea
18         {
19             get { return _IdTarea; }
20             set
21             {
22                 _IdTarea = value;
23                 OnPropertyChanged();
24             }
25         }
26         10 referencias
27         public string NombreTarea
28         {
29             get { return _NombreTarea; }
30             set
31             {
32                 _NombreTarea = value;
33                 OnPropertyChanged();
34             }
35         }
36         public event PropertyChangedEventHandler PropertyChanged;
37         2 referencias
38         protected void OnPropertyChanged([CallerMemberName] string name = null)
39         {
40             PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
41         }
42     }
43 }
```

Figura 9: En este código el objeto tipo Tasca contiene un Id y un nombre y contiene BindingContext para que se puedan crear en el Main.

Diagrama de Gantt



Problemas encontrados y sugerencias

Ha habido distintos errores al igual que siempre, después de ir probando, hacer otras veces el proyecto y ayudas externas se ha conseguido.

Conclusión

Buena actividad para para comprender la integración de Firebase en aplicaciones .NET MAUI, gestionando tareas en tiempo real.

Bibliografía/webgrafía

<https://firebase.google.com/?hl=es-419>

·Pdfs del profesor de mi profesor.

Video del funcionamiento



Dibujo 1: <https://youtu.be/UuJcJgr6NrE>