

# MEMORIA

## DESARROLLO DE INTERFACES



Alejandro Roberto Chiralt  
2ºDAM

## INDICE

INTRODUCCIÓN:.....	3
OBJETIVOS:.....	3
TAREAS A REALIZAR:.....	3
ACTIVIDADES :.....	3
FUNCIONAMIENTO:.....	4
CODIGO:.....	9
DIAGRAMA DE GANT:.....	13
PROBLEMAS:.....	13
CONCLUSIÓN:.....	13
WEBGRAFIA:.....	14

## INTRODUCCIÓN:

Este trabajo tiene como objetivo implementar una aplicación móvil para la gestión de tareas utilizando Firebase como backend. La aplicación permite a los usuarios agregar, actualizar, eliminar y visualizar tareas en tiempo real. Con esta solución, se busca proporcionar una herramienta práctica y eficiente para organizar actividades diarias, facilitando el seguimiento y la edición de tareas desde cualquier dispositivo compatible.

## OBJETIVOS:

Diseñar una interfaz intuitiva que permita a los usuarios interactuar fácilmente con las tareas.

- Implementar un servicio Firebase para gestionar tareas de forma segura y escalable.
- Facilitar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) con sincronización en tiempo real.
- Promover una experiencia de usuario fluida y responsiva.

## TAREAS A REALIZAR:

Configurar el servicio Firebase y establecer la conexión con la aplicación móvil.

- Implementar la funcionalidad de agregar tareas desde la interfaz de usuario.
- Desarrollar la lógica para actualizar tareas seleccionadas.
- Integrar la funcionalidad para eliminar tareas existentes.
- Implementar la sincronización en tiempo real para mantener la lista de tareas actualizada.
- Realizar pruebas para asegurar el correcto funcionamiento en diferentes dispositivos.

## ACTIVIDADES :

- **Configuración inicial:**

Configurar Firebase con las credenciales necesarias.  
Crear la base de datos y establecer las reglas de acceso.

- **Desarrollo de la aplicación:**

Diseñar el layout principal de la app con Xamarin/.NET MAUI.  
Implementar el binding con una colección observable para gestionar la lista de tareas.

- **Implementación de funcionalidades:**

- **Agregar tareas:** Insertar tareas ingresadas en el campo de texto.
- **Actualizar tareas:** Editar la descripción de una tarea seleccionada.
- **Eliminar tareas:** Borrar tareas específicas de la base de datos.
- **Sincronización:** Actualizar automáticamente la lista tras cualquier operación.

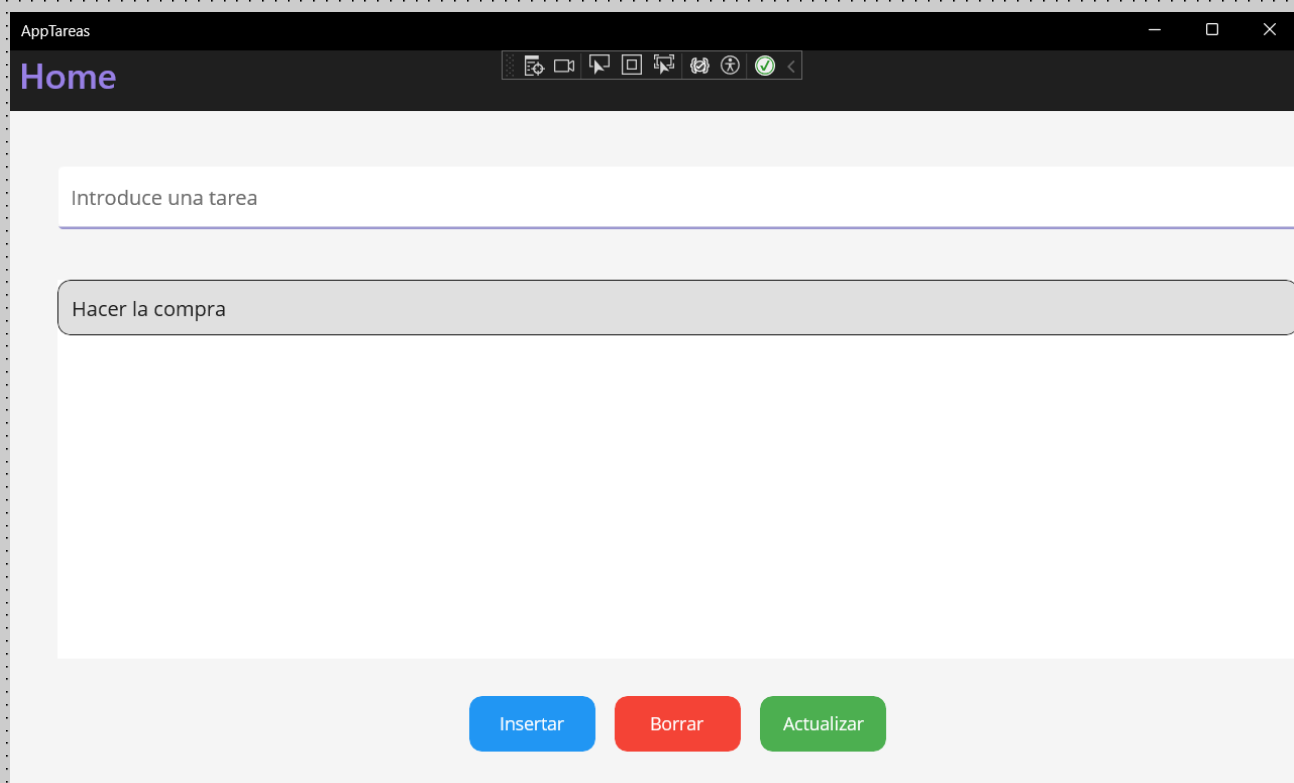
- **Pruebas y ajustes:**

Probar las funcionalidades en diferentes escenarios.  
Solucionar posibles errores y optimizar el rendimiento.

## FUNCIONAMIENTO:

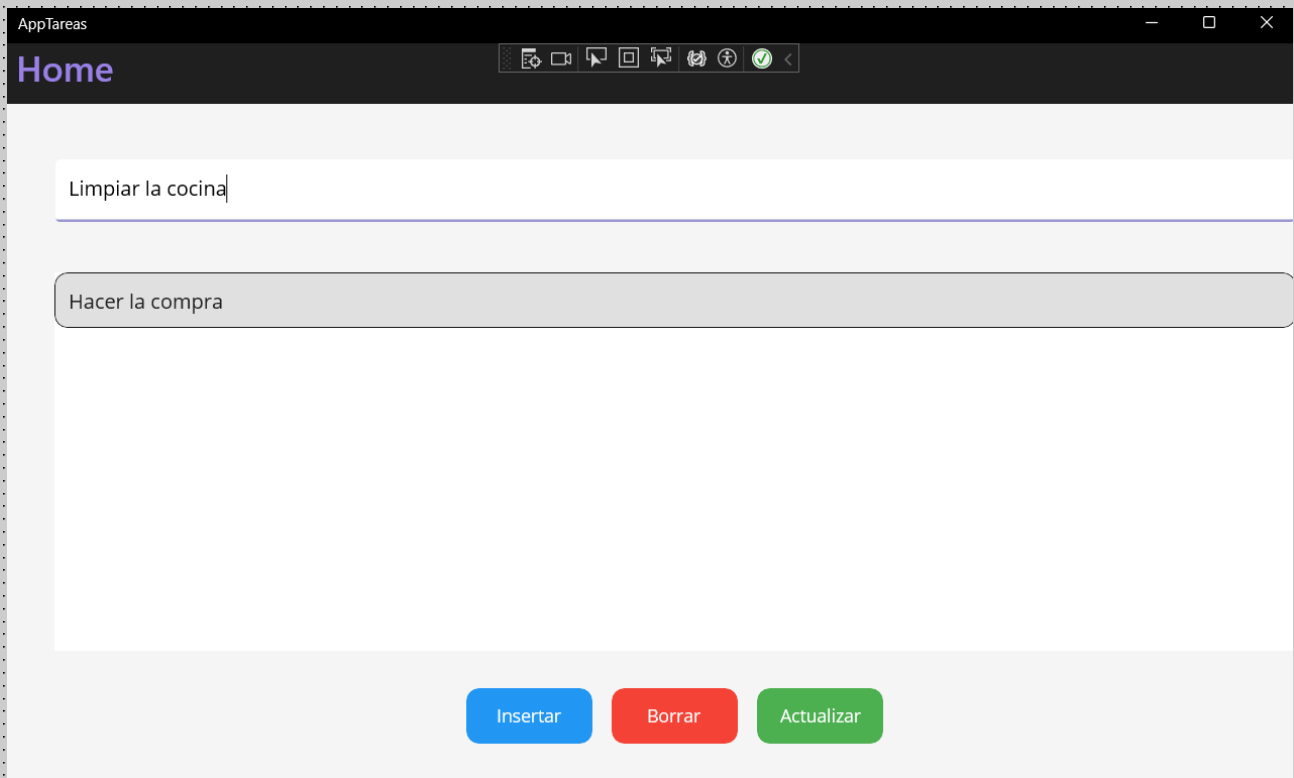
- **Inicio de la aplicación:**

Al abrir la app, la lista de tareas se carga automáticamente desde Firebase gracias a la llamada al método LoadTareas() en el evento OnAppearing.



- **Agregar tarea:**

El usuario escribe una descripción en el campo de texto.  
Al presionar el botón "Insertar", se guarda la tarea en Firebase mediante el método AddTarea.



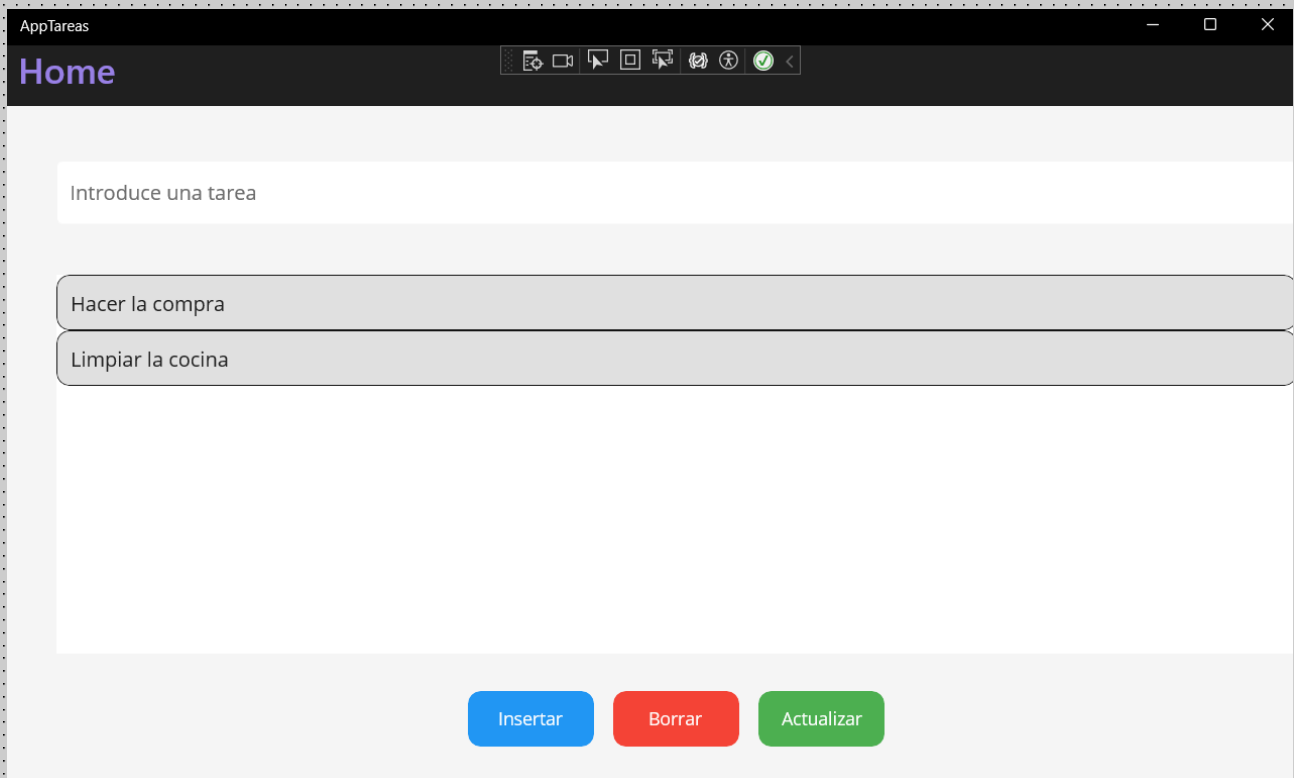
AppTareas

Home

Limpiar la cocina

Hacer la compra

Insertar Borrar Actualizar



AppTareas

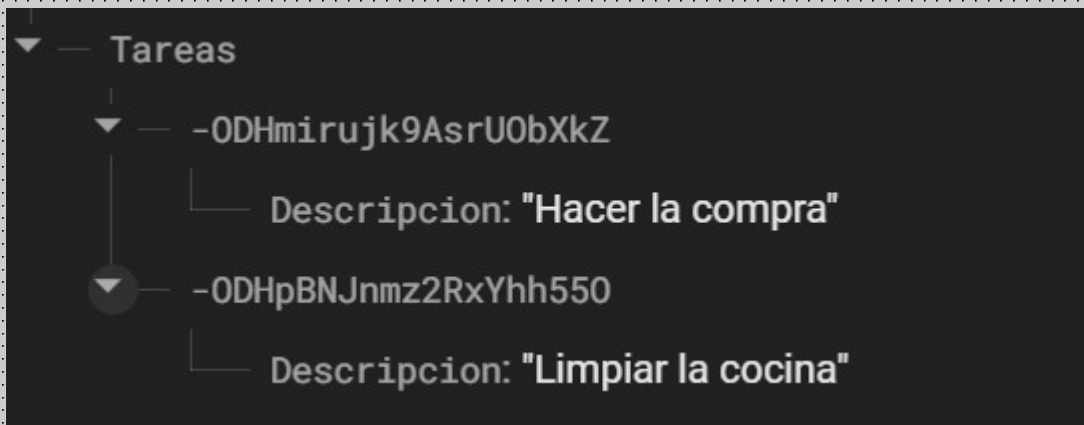
Home

Introduce una tarea

Hacer la compra

Limpiar la cocina

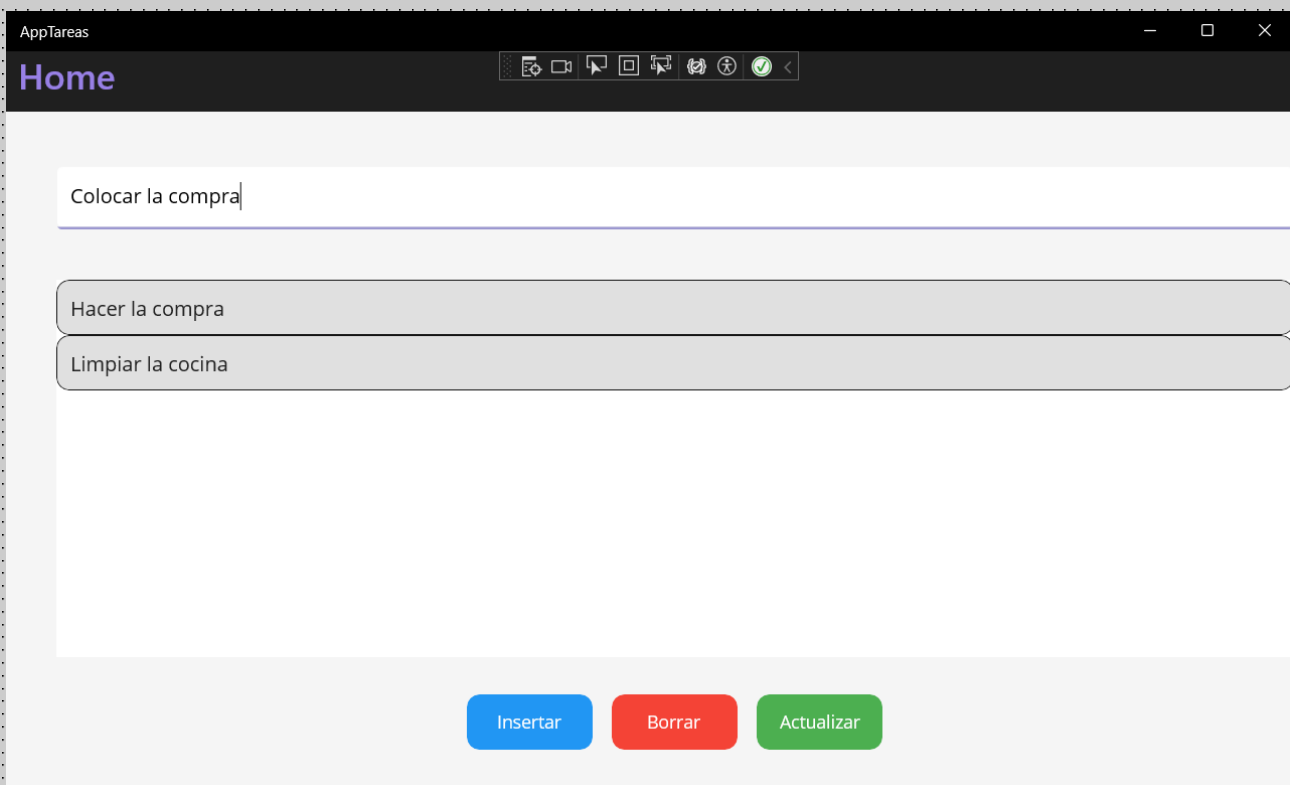
Insertar Borrar Actualizar

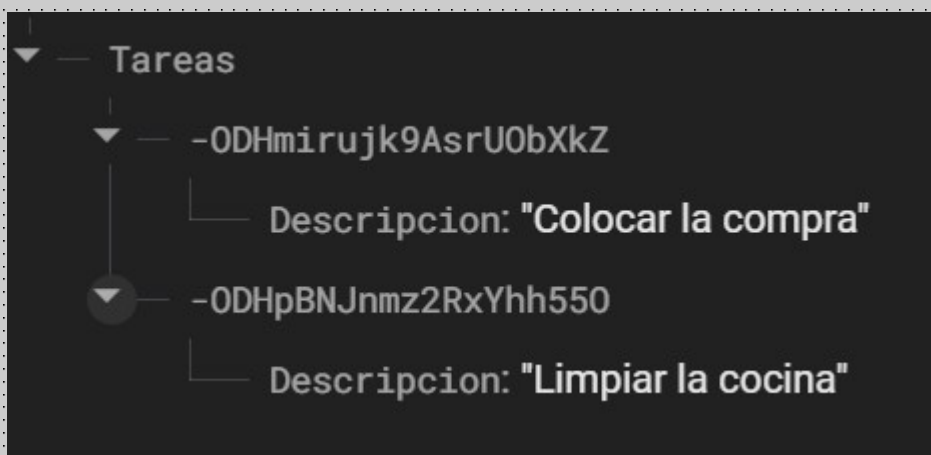
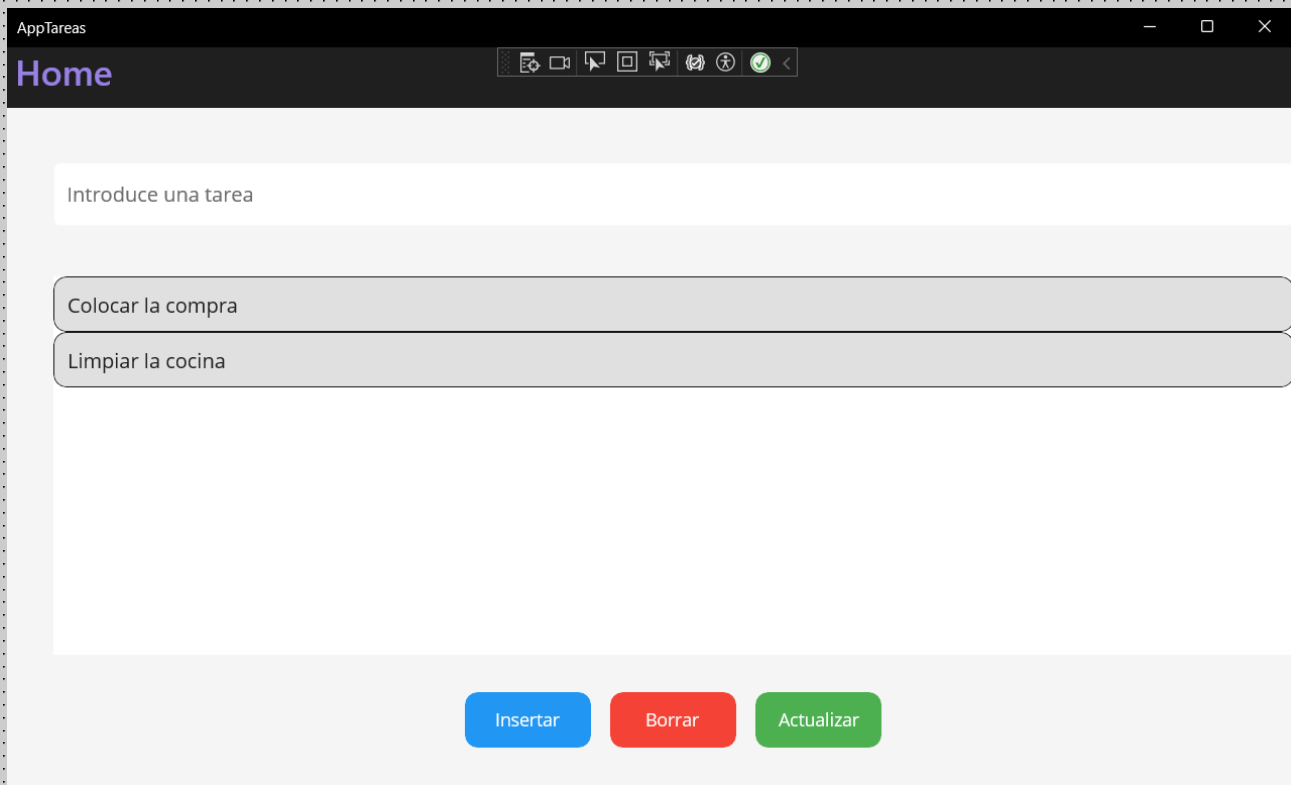


- **Actualizar tarea:**

El usuario selecciona una tarea de la lista, lo que rellena el campo de texto con la descripción de la tarea.

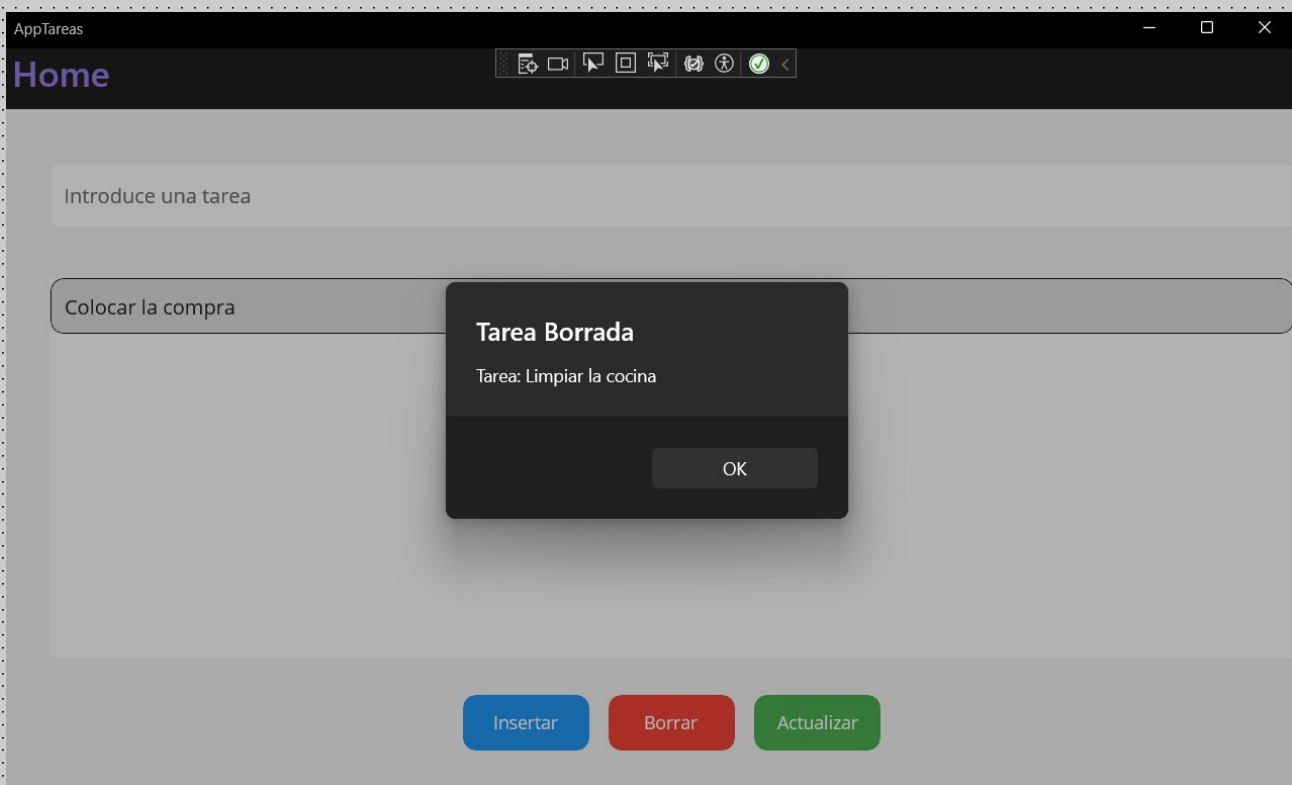
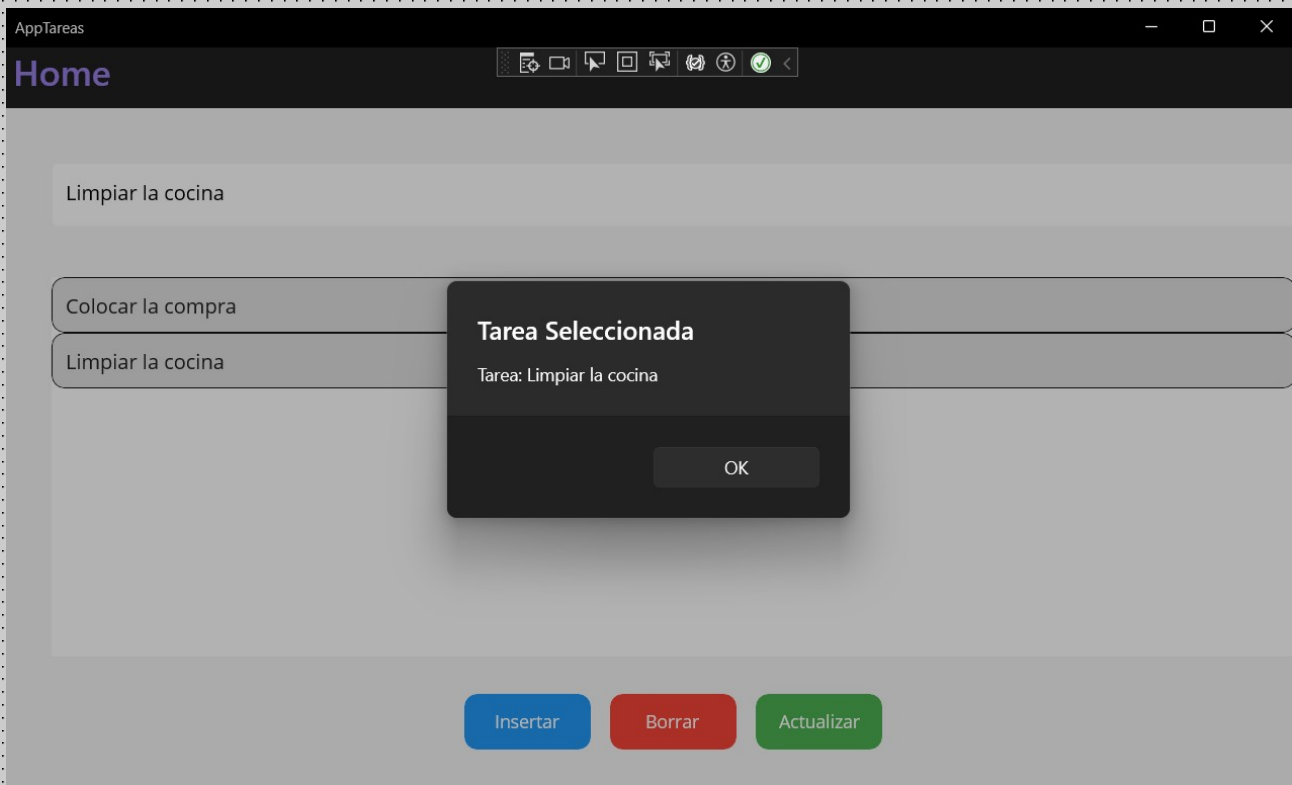
Luego, edita el texto y presiona el botón "Actualizar". Esto modifica la tarea en Firebase mediante el método `UpdateTarea`.



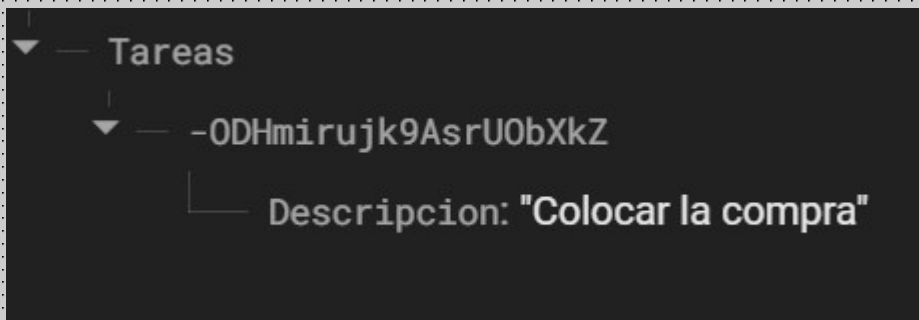


- **Eliminar tarea:**

El usuario selecciona una tarea de la lista.  
Al presionar el botón "Borrar", se elimina la tarea en Firebase mediante el método DeleteTarea.







```
<?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="AppTareas.MainPage"
BackgroundColor="#f5f5f5">
<ScrollView>
<VerticalStackLayout
Padding="20"
Spacing="20"
VerticalOptions="Center"
HorizontalOptions="Center">
<!-- Entrada de texto para la tarea -->
<Entry x:Name="etTarea"
Placeholder="Introduce una tarea"
BackgroundColor="#ffffff"
TextColor="Black"
Margin="0,10"
HeightRequest="50"
FontSize="16"
HorizontalOptions="FillAndExpand"/>
<!-- Lista de tareas -->
<CollectionView x:Name="cvTareas"
ItemsSource="{Binding Tareas}"
BackgroundColor="#ffffff"
HeightRequest="300"
Margin="0,10"
HorizontalOptions="FillAndExpand"
VerticalOptions="Start"
SelectionMode="Single"
SelectedItem="{Binding SelectedTarea}">
<CollectionView.ItemTemplate>
<DataTemplate>
<Frame Margin="10"
Padding="10"
CornerRadius="10"
BackgroundColor="#e0e0e0">
<Label Text="{Binding Descripcion}"
```

```

                FontSize="16"
                TextColor="#212121"/>
            </Frame>
        </DataTemplate>
    </CollectionView.ItemTemplate>
</CollectionView>
<!-- Botones -->
<HorizontalStackLayout
    Spacing="15"
    HorizontalOptions="Center">
    <Button x:Name="btnInsertar"
        Text="Insertar"
        BackgroundColor="#2196F3"
        TextColor="White"
        CornerRadius="10"
        Padding="10,5"
        WidthRequest="100"
        Clicked="btnInsertar_Clicked"/>
    <Button x:Name="btnBorrar"
        Text="Borrar"
        BackgroundColor="#F44336"
        TextColor="White"
        CornerRadius="10"
        Padding="10,5"
        WidthRequest="100"
        Clicked="btnBorrar_Clicked"/>
    <Button x:Name="btnActualizar"
        Text="Actualizar"
        BackgroundColor="#4CAF50"
        TextColor="White"
        CornerRadius="10"
        Padding="10,5"
        WidthRequest="100"
        Clicked="btnActualizar_Clicked"/>
    </HorizontalStackLayout>
</VerticalStackLayout>
</ScrollView>
</ContentPage>

```

MainPage.xaml.cs:

```

using AppTareas.Services;
using System.Collections.ObjectModel;
namespace AppTareas;
public partial class MainPage : ContentPage
{
    private readonly FirebaseService _firebaseService;
    public ObservableCollection<Tarea> Tareas { get; set; }
    public String tareaSeleccionada;
    public String idTareaSeleccionada;

    private Tarea _selectedTarea;
    public Tarea SelectedTarea
    {
        get => _selectedTarea;
        set
        {
            _selectedTarea = value;
            OnPropertyChanged();
        }
    }
    public MainPage()
    {
        InitializeComponent();
        _firebaseService = new FirebaseService();
        Tareas = new ObservableCollection<Tarea>();
    }
}

```

```

        BindingContext = this; // Establecer el BindingContext
    }
    private async void btnInsertar_Clicked(object sender, EventArgs e)
    {
        //Inserta la tarea escrita en el Entry
        if (!string.IsNullOrEmpty(etTarea.Text))
        {
            await _firebaseService.AddTarea(etTarea.Text);
            await LoadTareas();
            etTarea.Text = string.Empty;
        }
    }
    private async void btnBorrar_Clicked(object sender, EventArgs e)
    {
        await _firebaseService.DeleteTarea(idTareaSeleccionada);
        DisplayAlert("Tarea Borrada",
            $"Tarea: {SelectedTarea.Descripcion}",
            "OK");
        await LoadTareas();
        etTarea.Text = string.Empty;
    }
    private async void btnActualizar_Clicked(object sender, EventArgs e)
    {
        if (SelectedTarea != null && !string.IsNullOrEmpty(etTarea.Text))
        {
            SelectedTarea.Descripcion = etTarea.Text;
            // Llama al servicio para actualizar la tarea en Firebase
            await _firebaseService.UpdateTarea(SelectedTarea);
            // Recarga la lista de tareas para reflejar los cambios
            await LoadTareas();
            // Limpia el Entry después de actualizar
            etTarea.Text = string.Empty;
        }
        else
        {
            await DisplayAlert("Error", "Selecciona una tarea y asegúrate de que la descripción no esté vacía.", "OK");
        }
    }
    private async Task LoadTareas()
    {
        var tareasFirebase = await _firebaseService.GetTareas();
        Tareas.Clear();
        foreach (var tarea in tareasFirebase)
        {
            Tareas.Add(tarea);
        }
    }
    protected override async void OnAppearing()
    {
        base.OnAppearing();
        await LoadTareas();
    }
    private void OnPropertyChanged()
    {
        base.OnPropertyChanged();
        // Cuando la propiedad selectedTarea cambia, actualiza el Entry
        if (SelectedTarea != null)
        {
            DisplayAlert("Tarea Seleccionada",
                $"Tarea: {SelectedTarea.Descripcion}",
                "OK");
            idTareaSeleccionada = SelectedTarea.Id;
            tareaSeleccionada = SelectedTarea.Descripcion;
            etTarea.Text = tareaSeleccionada;
        }
    }

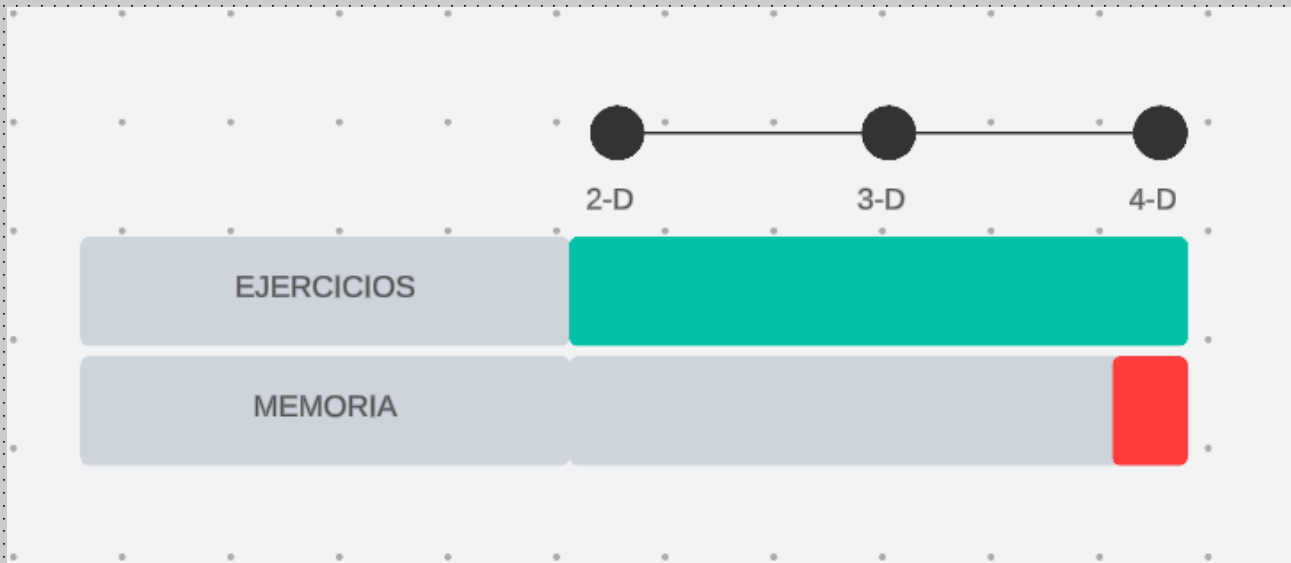
```

```
    }  
  }  
}
```

#### FirestoreService.cs

```
using Firebase.Database;  
using Firebase.Database.Query;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
namespace AppTareas.Services  
{  
    public class FirestoreService  
    {  
        private readonly FirebaseClient _firebaseClient;  
        public FirestoreService()  
        {  
            _firebaseClient = new FirebaseClient("https://ejerbotones12-a3b636ec-  
default-rtdb.europe-west1.firebaseio.com/");  
        }  
        public async Task AddTarea(string descripcion)  
        {  
            await _firebaseClient  
                .Child("Tareas")  
                .PostAsync(new Tarea { Descripcion = descripcion });  
        }  
        public async Task UpdateTarea(Tarea tarea)  
        {  
            if (string.IsNullOrEmpty(tarea.Id))  
            {  
                throw new ArgumentException("La tarea debe tener un ID válido.");  
            }  
            // Actualizamos solo la descripción de la tarea  
            await _firebaseClient  
                .Child("Tareas")  
                .Child(tarea.Id)  
                .PatchAsync(new { Descripcion = tarea.Descripcion });  
        }  
        public async Task DeleteTarea(string id)  
        {  
            await _firebaseClient  
                .Child("Tareas")  
                .Child(id)  
                .DeleteAsync();  
        }  
        public async Task<List<Tarea>> GetTareas()  
        {  
            var tareas = await _firebaseClient  
                .Child("Tareas")  
                .OnceAsync<Tarea>();  
            return tareas.Select(t => new Tarea  
            {  
                Id = t.Key,  
                Descripcion = t.Object.Descripcion  
            }).ToList();  
        }  
    }  
}
```

## DIAGRAMA DE GANT:



## PROBLEMAS:

Los mayores problemas que he tenido ha sido a la hora de comprender como funciona MAUI , y el como almacenar y hacer un setter de los datos. A la hora de actualizar, se actualiza de la base de datos, pero del Observable Collection.

## CONCLUSIÓN:

La implementación de esta aplicación móvil para la gestión de tareas demuestra cómo las herramientas modernas como Firebase y .NET MAUI pueden integrarse para ofrecer soluciones eficientes y multiplataforma. El desarrollo permitió crear una interfaz intuitiva y funcionalidades completas para la administración de tareas, asegurando una experiencia de usuario fluida y una sincronización en tiempo real.

El proyecto cumple con los objetivos planteados, proporcionando un sistema robusto que facilita la organización personal o profesional de las actividades diarias. Además, este desarrollo pone en evidencia la importancia de adoptar tecnologías escalables y flexibles, sentando las bases para futuras ampliaciones, como la incorporación de autenticación de usuarios o notificaciones push.

En resumen, este trabajo resalta el impacto positivo de las aplicaciones móviles bien diseñadas en la productividad y la organización, brindando a los usuarios una herramienta confiable y accesible en cualquier momento y lugar.

## WEBGRAFIA:

<https://learn.microsoft.com/es-es/dotnet/maui/get-started/first-app?view=net-maui-8.0&tabs=vswin&pivots=devices-android>