UNIVERSIDAD TECNOLOGICA DE HONDURAS

INFORME TECNICO

CLASE:

PROGRAMACION MOVIL II

CATDRATICO:

ING. RICARDO LAGOS

EQUIPO:

1. Josue Bladimir Sierra Maradiaga N° cuenta: 201810110287

2. Cindy Consuelo Caballero N° cuenta: 201920110225

3. Johan Odeth Torres López N° cuenta: 201920060124

4. Litzy Gissela Lainez Cruz N° cuenta: 201810110309

5. Linda Vanessa Gáleas N° cuenta: 201810110289

6. Jose Luis Ramos Perez N° cuenta: 201930010343

UNIVERSIDAD TECNOLOGICA DE HONDURAS

INFORME TECNICO

CLASE:

PROGRAMACION MOVIL II

CATDRATICO:

ING. RICARDO LAGOS

Tabla de Contenido

INTRODUCCION DEL PROYECTO

Una actividad indispensable en nuestro Smartphone es de crear notas, actualmente de una manera sencilla para que estas nos sirvan de recordatorio en algunos casos futuro.Al dia de hoy esta actividad ha ido mejorando hasta el punto de poder agregar audios, imágenes etc. Y no solo poder crear una nota simple, además de poder compartirla a través de distintas aplicaciones.

Es por ello que a traves de una asignación dada por el catedrático de la clase de programación móvil II hemos decidido crear como equipo EasyNotas he impulsar esta aplicación como una marca, es importante llegar a un buen número de personas, por lo que hemos empleado un enfoque multifacético tratándose de una aplicación móvil para dispositivos Android que permite registrarse como usuario de la apliación y asi poder crear notas de una manera interactiva.

En esta investigación se detalla el análisis de la problemática, y la manera en que nosotros estamos dando la solución esperada para el proyecto, donde enfatizamos las razones principales, más allá de una perspectiva de desarrollo y de diseño, puntualizamos las funciones de la aplicación, lo que se espera una vez finalizaba, y ver si esta cumple con los requerimientos y el resto del desarrollo de la aplicación, cuya base será **Visual Studio 2019/2022 – Xamarin** y se complementará con otros programas diseñados para el desarrollo.

OBJETIVOS

OBJETIVOS GENERALES

Desarrollar e implementar una aplicación Móvil en la plataforma **Visual Studio 2019/2022 – Xamarin** que permita crear notas de diferentes maneras, texto, audio y con imágenes. Esto para que dirija al estudiante al objetivo de aplicar lo aprendido en clase basándose en las necesidades del mercado real.

OBJETIVOS ESPECIFICOS

- ✓ Aplicar los conocimientos adquiridos en clase a problemas reales relacionados con aplicaciones móviles.
- ✓ Demostrar las habilidades adquiridas de programación durante la clase haciendo uso de la herramienta de Visual Studio 2019/2022 - Xamarin.
- ✓ Dirigir los proyectos relacionados con las aplicaciones para dispositivos móviles, validando casos de uso y asegurando la calidad del servicio.

ANÁLISIS DEL PROBLEMA

JUSTIFICACIÓN

Para nosotros estar al par con el mundo globalizado debemos implementar y poner en practica para ir desarrollando nuevas aplicaciones y novedosas de las cuales muchos usuarios las vean atractivas y hacer uso de ellas .Este proyecto tiene como finalidad identificar componentes y herramientas que se utilizan para poder llevar a cabo una aplicación de notas que en nuestro caso lleva por nombre "EasyNotas".

Teniendo como objetivo determinar la viabilidad económica y financiera logrando un consenso entre los colaboradores de la empresa para la adecuada ejecución de la aplicación, en la cual fortalecemos nuestros conocimientos y las bases para la formación como futuros ingenieros en computación que seremos. La idea nace principalmente como una asignación, de la cual nosotros acogemos y la hacemos nuestras, sabemos que aplicaciones de notas hay muchas pero nuestro estilo es lo que marcara la diferencia.

ALCANCE DEL PROYECTO

Presentamos la Aplicación Móvil de creación de notas con los procesos de crear, guardar y compartir. El problema al cual le damos respuestas es para una persona natural que necesita de recordatorios por medio de 3 diferentes vías: Texto, audio e imágenes , en nuestro caso con la aplicación que lleva por nombre EasyNotas, deseamos expandir nuestra marca de manera digital, dada la situación que, como aplicación creada legalmente en el campo digital, suplirá mucha necesidades a quien utilice nuestra aplicación ademas de que podrá vivir una experiencia bastante interactiva, ya que nuestra aplicación tiene los siguientes procesos

1.Solicitar datos del usuario para poder iniciar sesión dentro de la aplicación.

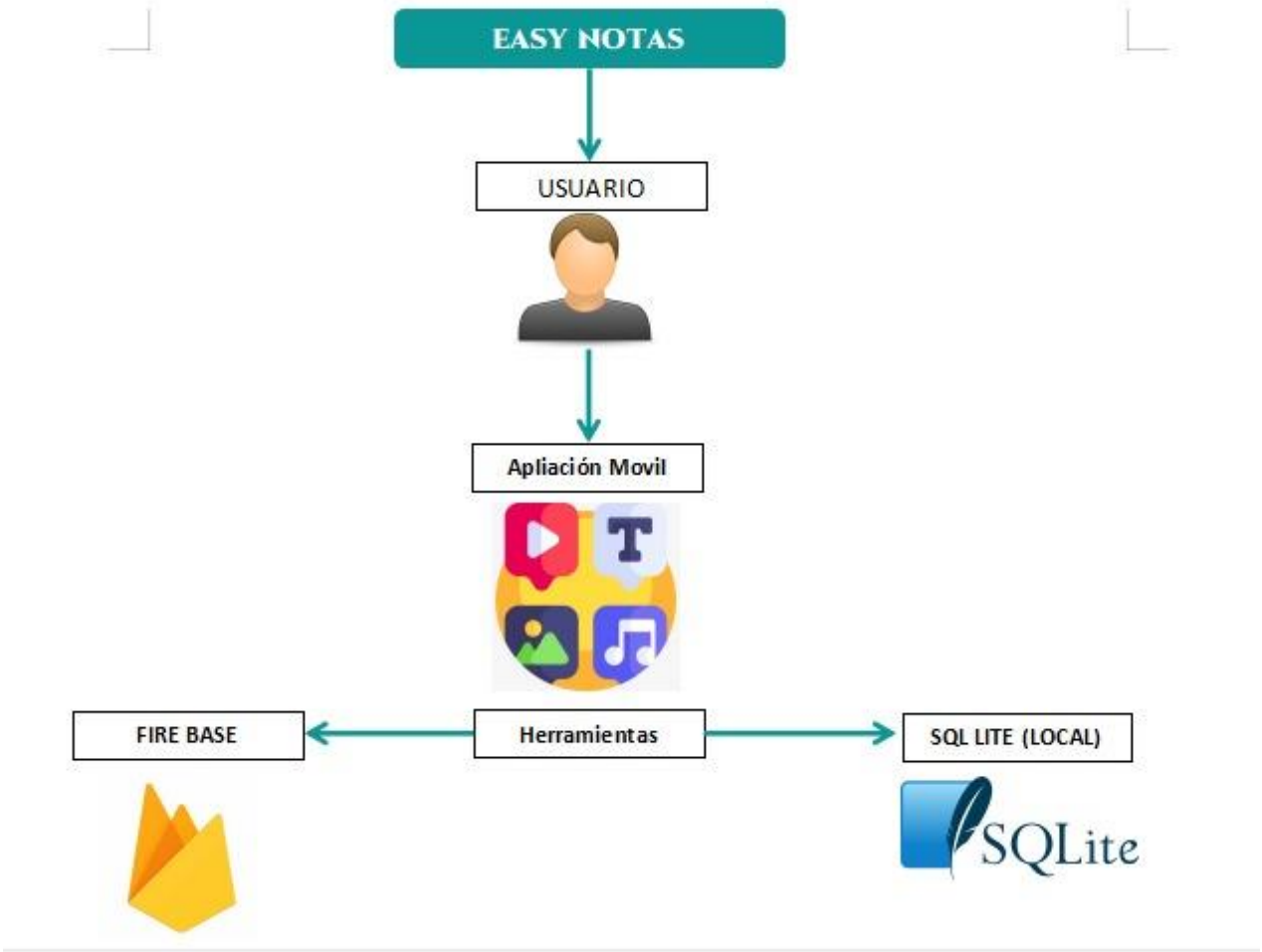
2.Seleccionar el tipo de nota (Texto, imagen o audio) a crear.

3. Seleccionar la nota creada para modificarla.

4.Dentro de la nota seleccionada se encuentran opciones como eliminar la nota, compartirla.

Para desarrollar un mejor manejo de la aplicación, se realizará una investigación para poder agregar más opciones interactivas para el usuario.

ARQUITECTURA DEL PROYECTO



DESARROLLO

Herramientas

- Android Studio
- FireBase
- SQL LITE (Local)

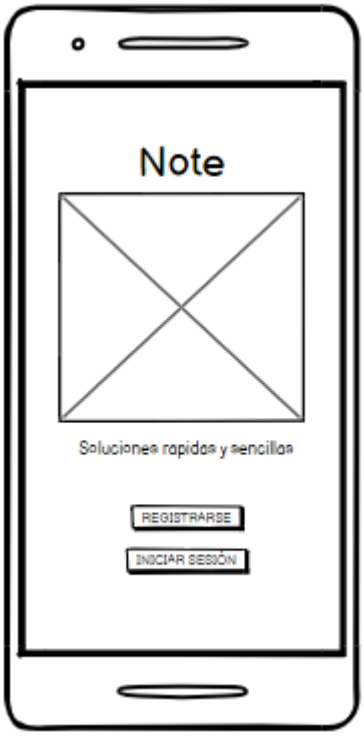
Librerías

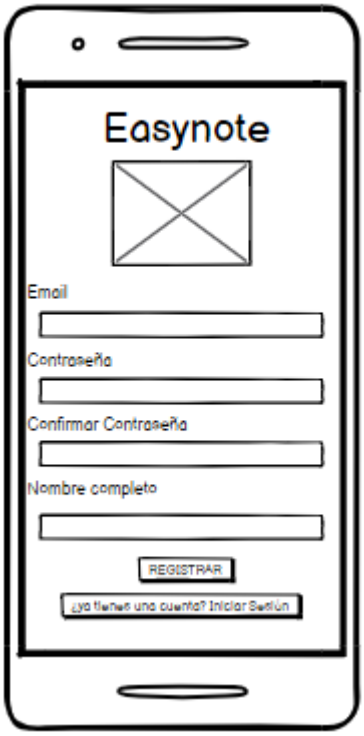
- FireBase
- Sqlite
- NETStandart

DISEÑO

PROTOTIPO EXPLORATORIO – BAJA CALIDAD

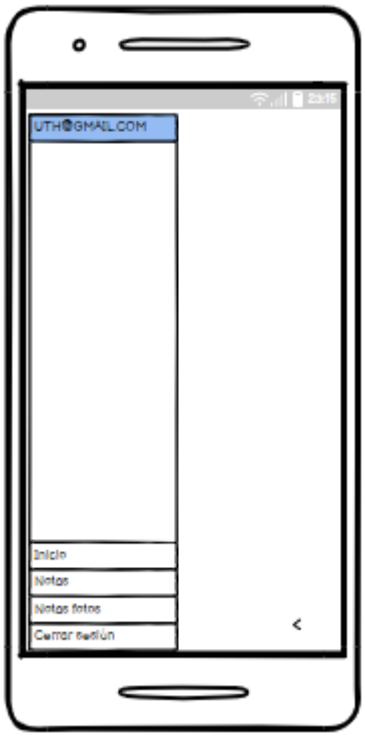
Pantalla principal y de registro





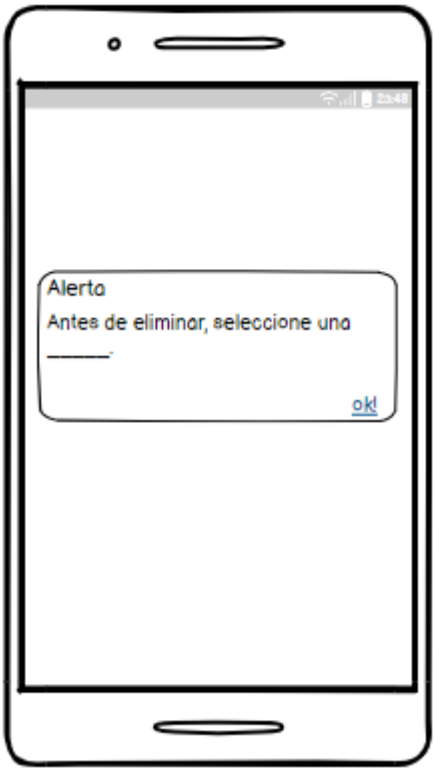
Pantalla inicio de sesión y recuperación de cuenta

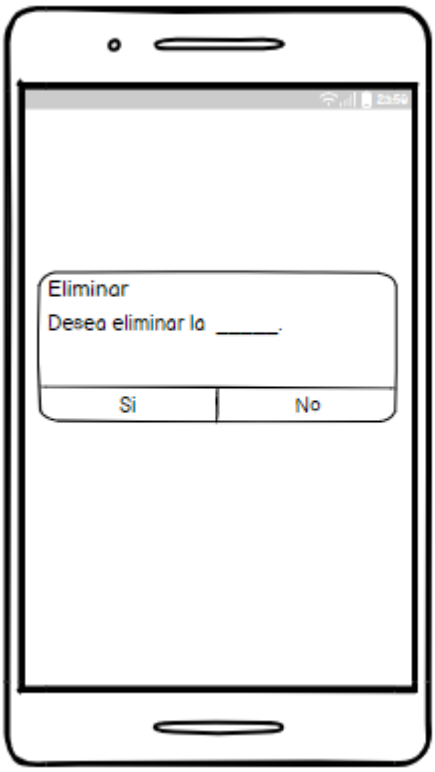
Pantalla menú

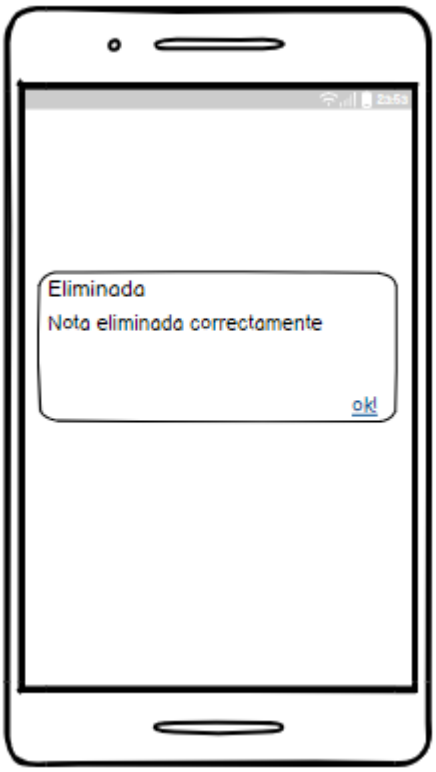


Pantalla insertar nota

Alertas


Alerta
Antes de eliminar, seleccione una _____.
ok!


Eliminar
Desea eliminar la _____.
Si
No

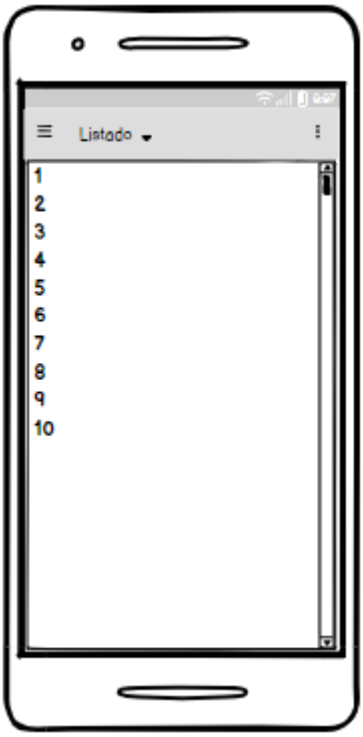

Eliminada
Nota eliminada correctamente
ok!


Campos vacios
Llene todos los campos
ok!

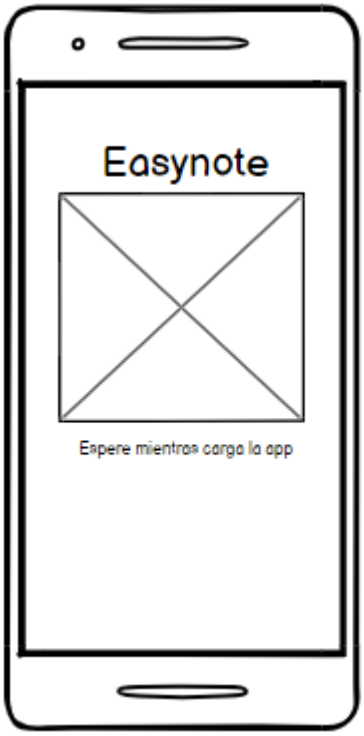
Captura de imagen

Nota de audio

Listado

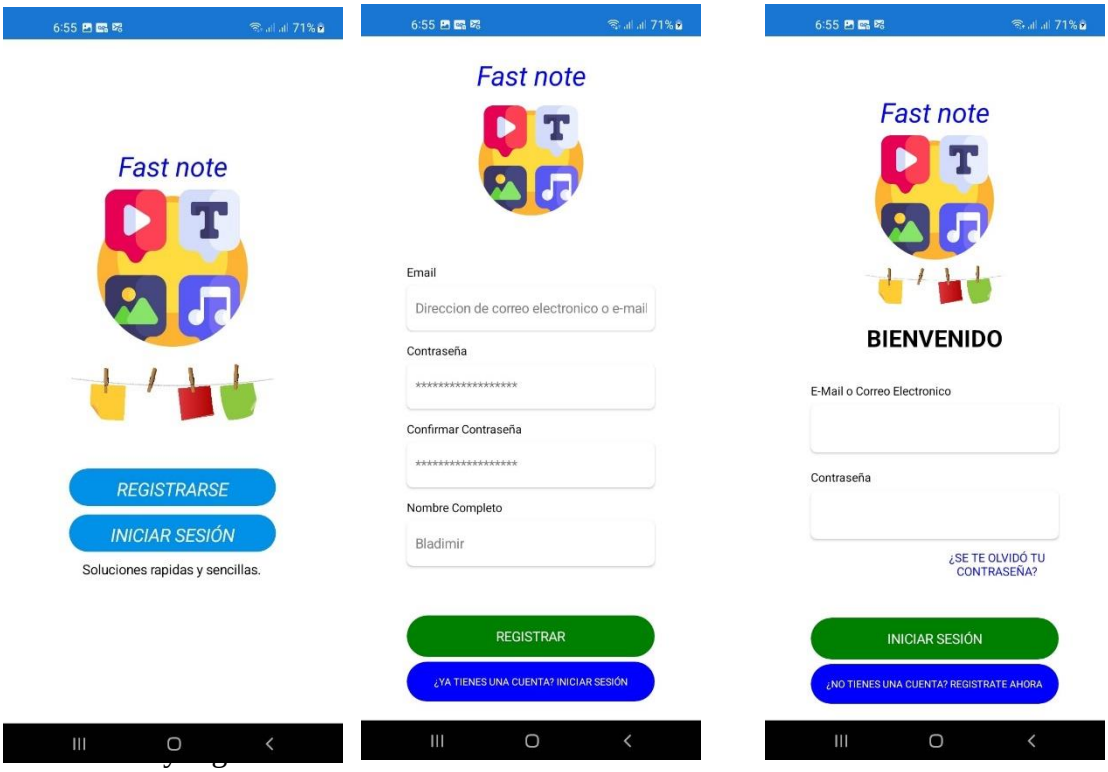


Espera

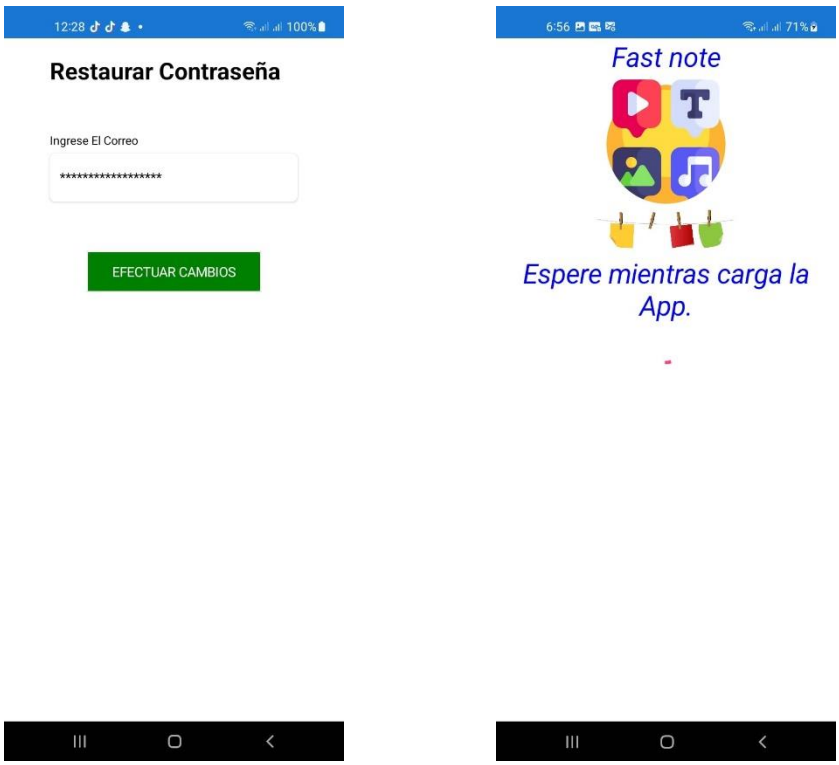


Cerrar sesión

PROTOTIPO DE ALTA FIDELIDAD Y OPERACIONAL



Recuperación de cuenta y sala de espera

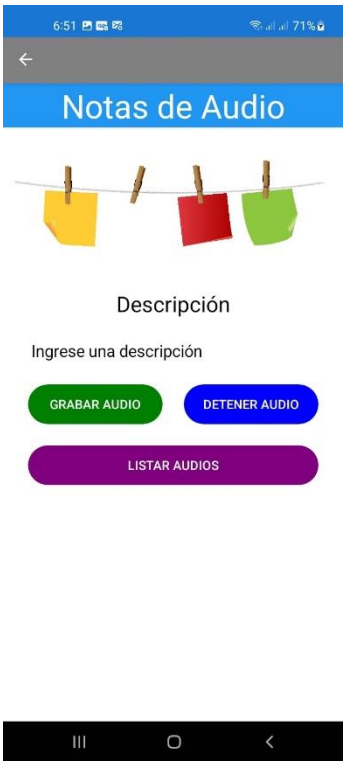


Pantalla principal y menu

Llenados


6:50
71%
Foto Notas
Titulo de Notas
Foto nota Prueba
Descripcion de nota
HOLIS


6:50
71%
Titulo de Notas
Prueba 2
Descripcion de nota
HOLA PRUEBA 2


6:51
71%
Notas de Audio
Descripción
Ingrese una descripción
GRABAR AUDIO
DETENER AUDIO
LISTAR AUDIOS

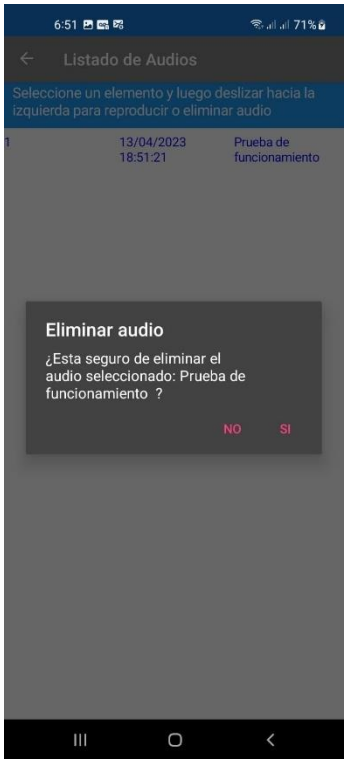
Listas


6:50
71%
Lista de Notas Guardadas
Buscar
PRUEBA
PRUEBA 2


6:51
71%
Listado de Audios
Seleccione un elemento y luego deslizar hacia la izquierda para reproducir o eliminar audio
1
13/04/2023 18:51:21
Prueba de funcionamiento


6:50
71%
Foto Notas
AGREGAR FOTO NOTA
FOTO NOTA PRUEBA
HOLIS
uth@gmail.com

Alertas



Cerrar sesion

CODIGO DOCUMENTACIÓN

Controller

AudioModel.cs

```
namespace EasyNote.Models
{
    4 referencias
    public class AudioModelDB
    {
        readonly SQLiteAsyncConnection db;

        1 referencia
        public AudioModelDB(string pathdb)
        {
            db = new SQLiteAsyncConnection(pathdb);
            db.CreateTableAsync<AudioModel>().Wait();
        }

        1 referencia
        public Task<List<AudioModel>> ObtenerListaAudios()
        {
            return db.Table<AudioModel>().ToListAsync();
        }

        1 referencia
        public Task<AudioModel> ObtenerAudio(int pcodigo)
        {
            return db.Table<AudioModel>()
                .Where(i => i.Id == pcodigo)
                .FirstOrDefaultAsync();
        }

        1 referencia
        public Task<int> GrabarAudio(AudioModel audio)
        {
            if (audio.Id != 0)
```

```
        1 referencia
        public Task<AudioModel> ObtenerAudio(int pcodigo)
        {
            return db.Table<AudioModel>()
                .Where(i => i.Id == pcodigo)
                .FirstOrDefaultAsync();
        }

        1 referencia
        public Task<int> GrabarAudio(AudioModel audio)
        {
            if (audio.Id != 0)
            {
                return db.UpdateAsync(audio);
            }
            else
            {
                return db.InsertAsync(audio);
            }
        }

        1 referencia
        public Task<int> EliminarAudio(AudioModel audio)
        {
            return db.DeleteAsync(audio);
        }


    }
}
```

No se encontraron problemas.

FirebaseHelper.cs

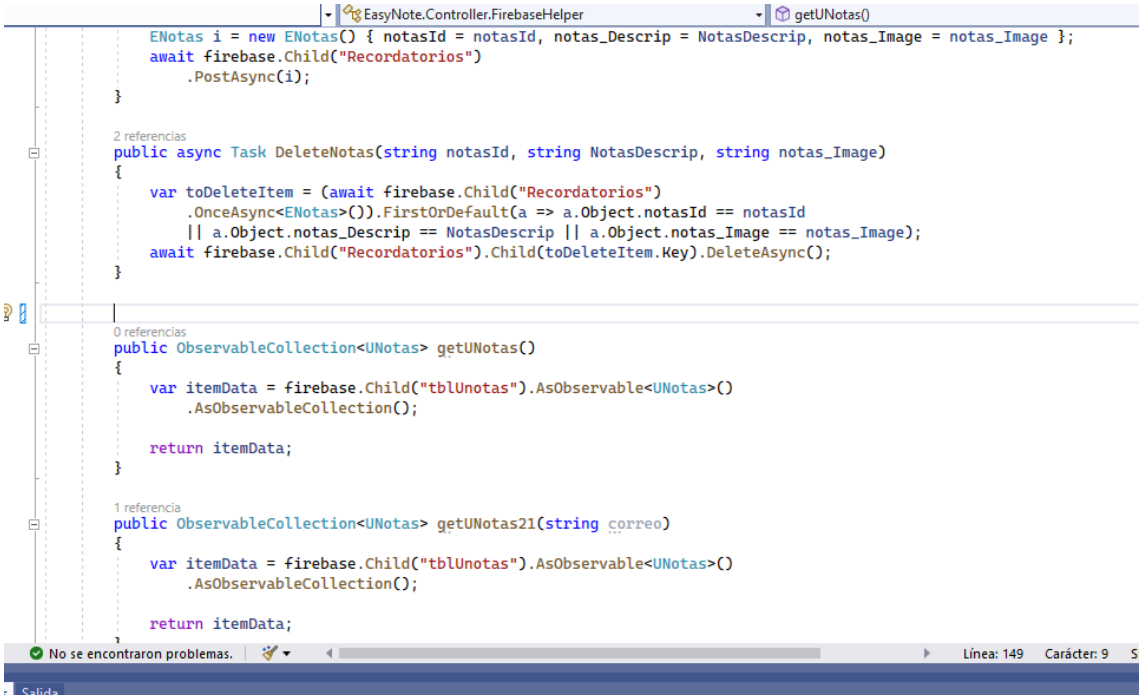
```
namespace EasyNote.Controller
{
    14 referencias
    public class FirebaseHelper
    {
        FirebaseClient firebase = new FirebaseClient("https://easynote-7c1a5-default-rtdb.firebaseio.com/");
        static FirebaseStorage stroageImage = new FirebaseStorage("easynote-7c1a5.appspot.com");

        1 referencia
        public async Task<List<Recordatorio>> GetAllPersons()
        {

            return (await firebase
              .Child("Notas")
              .OnceAsync<Recordatorio>()).Select(item => new Recordatorio
              {
                  NotasDescrip = item.Object.NotasDescrip,
                  notasId = item.Object.notasId
              }).ToList();
        }

        0 referencias
        public async Task AddPerson(string name, Byte[] imagen)
        {

            await firebase
              .Child("Notas")
              .PostAsync(new Recordatorio()
              {
                  NotasDescrip = name,
                  notas_Image = imagen
              });
        }

        0 referencias
        public async Task AddPersonAu(AudioRecorderService name)
        {

            await firebase
              .Child("Notas")
              .PostAsync(new Notas()
              {
                  recorder = name
              });
        }

        0 referencias
        public async Task<Recordatorio> GetPerson(String personId)
        {
            var allPersons = await GetAllPersons();
            await firebase
              .Child("Persons")
              .OnceAsync<Recordatorio>();
            return allPersons.Where(a => a.notasId == personId).FirstOrDefault();
        }

        0 referencias
        public async Task UpdatePerson(string personId, string name)
```

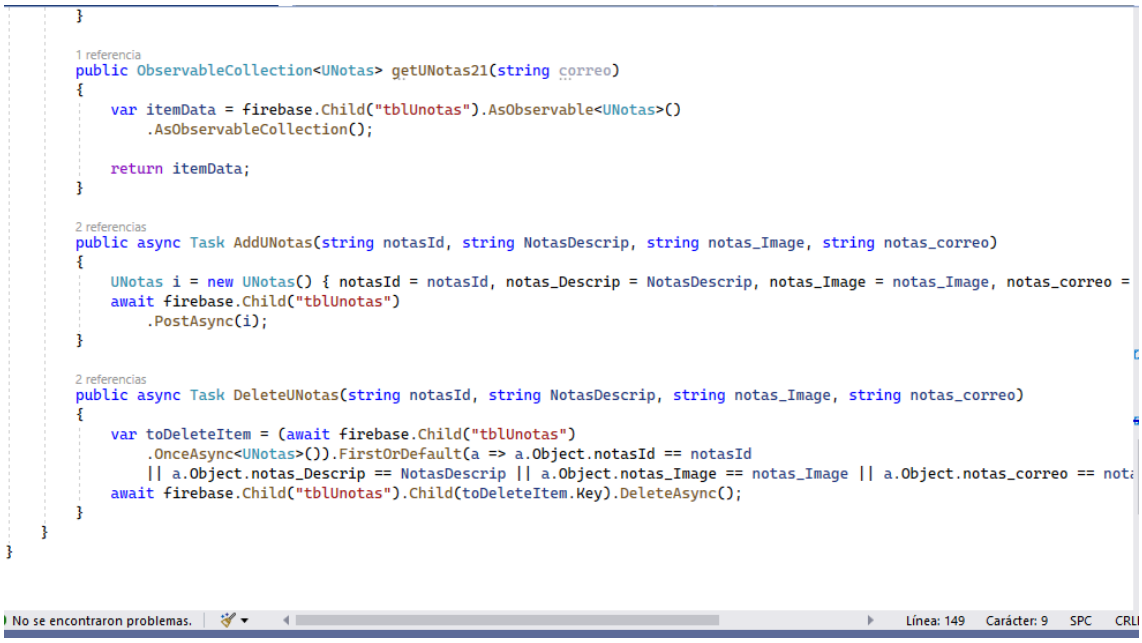
```
0 referencias
public async Task UpdatePerson(string personId, string name)
{
    var toUpdatePerson = (await firebase
      .Child("Persons")
      .OnceAsync<Recordatorio>()).Where(a => a.Object.notasId == personId).FirstOrDefault();

    await firebase
      .Child("Persons")
      .Child(toUpdatePerson.Key)
      .PutAsync(new Recordatorio() { notasId = personId, NotasDescrip = name });
}

0 referencias
public async Task DeletePerson(string personId)
{
    var toDeletePerson = (await firebase
      .Child("Persons")
      .OnceAsync<Recordatorio>()).Where(a => a.Object.notasId == personId).FirstOrDefault();
    await firebase.Child("Persons").Child(toDeletePerson.Key).DeleteAsync();

}

0 referencias
public async Task AddNotas(string notasId, string notas_Descrip, string notas_Image, string notas_Audio, string userId)
{


    //  downloadLink.Text = downloadlink;
    await firebase
```

No se encontraron problemas. Línea: 8 Carácter: 17 SPC CRLF

```
2 referencias
public async Task Update_Notas(string notasId, string notasDescrip, string notasImage)
{
    var toUpdateNotas = (await firebase.
        Child("Recordatorios").OnceAsync<ENotas>()).FirstOrDefault
        (a => a.Object.notasId == notasId
        || a.Object.notas_Descrip == notasDescrip || a.Object.notas_Image == notasImage);
    ENotas u = new ENotas() { notasId = notasId, notas_Descrip = notasDescrip, notas_Image = notasImage };
    await firebase.Child("Recordatorios").Child(toUpdateNotas.Key).PutAsync(u);
}

0 referencias
public static async Task<string> UploadFile(Stream fileStream, string fileName)
{
    var imageUrl = await stroageImage
        .Child("NotasImagenes")
        .Child(fileName)
        .PutAsync(fileStream);
    return imageUrl;
}


1 referencia
public ObservableCollection<ENotas> getNotas()
{
    var itemData = firebase.Child("Recordatorios").AsObservable<ENotas>()
        .AsObservableCollection();

    return itemData;
}
```

No se encontraron problemas. Línea: 124 Carácter: 9





FirebaseHelperUsers.cs

```
namespace EasyNote.Controller
{
    1 referencia
    public class FirebaseHelperUsers
    {
        public static FirebaseClient firebase = new FirebaseClient("https://easynote-7c1a5-default-rtdb.firebaseio.com/Users");

        //Read All
        1 referencia
        public static async Task<List<Users>> GetAllUser()
        {
            try
            {
                var userlist = (await firebase
                .Child("Users")
                .OnceAsync<Users>()).Select(item =>
                new Users
                {
                    Email = item.Object.Email,
                    Password = item.Object.Password
                }).ToList();
                return userlist;
            }
            catch (Exception e)
            {
                Debug.WriteLine($"Error:{e}");
                return null;
            }
        }
```

```
        //Read
        0 referencias
        public static async Task<Users> GetUser(string email)
        {
            try
            {
                var allUsers = await GetAllUser();
                await firebase
                .Child("Users")
                .OnceAsync<Users>();
                return allUsers.Where(a => a.Email == email).FirstOrDefault();
            }
            catch (Exception e)
            {
                Debug.WriteLine($"Error:{e}");
                return null;
            }
        }

        //Inser a user
        1 referencia
        public static async Task<bool> AddUser(string email, string password, string username)
        {
            try
            {


                await firebase
                .Child("Users")
                .PostAsync(new Users() { Email = email, Password = password, Username = username });
```

No se encontraron problemas. Línea: 14

RegistroLoginController.cs

```
namespace EasyNote.Controller
{
    3 referencias
    public class RegistroLoginController : Base
    {
        Attributes

        Properties

        Commands

        Methods

        Constructor
    }
}
```

Models

AudioModel.cs

```
namespace EasyNote.Models
{
    11 referencias
    public class AudioModel
    {

        [PrimaryKey, AutoIncrement]
        3 referencias
        public int Id { get; set; }
        3 referencias
        public string Url { get; set; }
        3 referencias
        public string Descripcion { get; set; }
        1 referencia
        public DateTime Date { get; set; }
    }
}
```

Base.cs

```
namespace EasyNote.Models
{
    2 referencias
    public class Base
    {
        public event PropertyChangedEventHandler PropertyChanged;


        1 referencia
        protected void OnPropertyChanged([CallerMemberName] string propertyName = null)

        {

            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

        }


        11 referencias
        protected void SetValue<T>(ref T backingFieled, T value, [CallerMemberName] string propertyName = null)

        {

            if (EqualityComparer<T>.Default.Equals(backingFieled, value))

            {

                return;

            }
```

```
                return;
            }

            backingFieled = value;

            OnPropertyChanged(propertyName);

        }


        0 referencias
        protected virtual void OnPropertyChangeds([CallerMemberName] string propertyName = null)

        {

            PropertyChangedEventHandler handler = PropertyChanged;

            if (handler != null)

            {

                handler(this, new PropertyChangedEventArgs(propertyName));

            }

        }
    }
}
```

Correo.cs

```
4
5
6   namespace EasyNote.Models
7   {
        2 referencias
8       public class Correo
9       {
            1 referencia
.0          public string usuario { get; set; }
.1      }
.2  }
.3
```

ENotas.cs

```
namespace EasyNote.Models
{
    16 referencias
    public class ENotas
    {
        5 referencias
        public string notasId { get; set; }

        4 referencias
        public string notas_Descrip { get; set; }

        4 referencias
        public string notas_Image { get; set; }
    }
}
```

Item.cs

```
namespace EasyNote.Models
{
    27 referencias
    public class Item
    {
        13 referencias
        public string Id { get; set; }
        8 referencias
        public string Text { get; set; }
        8 referencias
        public string Description { get; set; }
    }
}
```

Notas.cs

```
namespace EasyNote.Models
{
    1 referencia
    public class Notas
    {
        1 referencia
        public AudioRecorderService recorder { set; get; }
        0 referencias
        public AudioPlayer player { set; get; }
    }
}
```

NotasGeneral.cs

```
namespace EasyNote.Models
{
    1 referencia
    class NotasGeneral
    {
        1 referencia
        public string notasId { get; set; }

        1 referencia
        public string notas_Descrip { get; set; }

        1 referencia
        public string notas_Image { get; set; }
        1 referencia
        public string notas_Audio { get; set; }

        1 referencia
        public string userId { get; set; }
    }
}
```

Recordatorio.cs

```
namespace EasyNote.Models
{
    9 referencias
    public class Recordatorio
    {
        6 referencias
        public String notasId { set; get; }
        0 referencias
        public String notas_Fecha { set; get; }
        1 referencia
        public Byte[] notas_Image { set; get; }
        0 referencias
        public String notas_Audio { set; get; }
        4 referencias
        public String NotasDescrip { set; get; }
        0 referencias
        public String userId { set; get; }
    }
}
```

UNotas.cs

```
namespace EasyNote.Models
{
    15 referencias
    public class UNotas
    {
        3 referencias
        public string notasId { get; set; }

        2 referencias
        public string notas_Descrip { get; set; }

        2 referencias
        public string notas_Image { get; set; }

        2 referencias
        public string notas_correo{ get; set; }
    }
}
```

Users.cs

```
namespace EasyNote.Models
{
    9 referencias
    public class Users
    {
        7 referencias
        public String Email { get; set; }
        4 referencias
        public String Password { get; set; }

        1 referencia
        public String Username { get; set; }

    }
}
```

Services

IDataStore.cs

```
namespace EasyNote.Services
{
    3 referencias
    public interface IDataStore<T>
    {
        2 referencias
        Task<bool> AddItemAsync(T item);
        1 referencia
        Task<bool> UpdateItemAsync(T item);
        1 referencia
        Task<bool> DeleteItemAsync(string id);
        2 referencias
        Task<T> GetItemAsync(string id);
        2 referencias
        Task<IEnumerable<T>> GetItemsAsync(bool forceRefresh = false);
    }
}
```

MockDataStore.cs

```
namespace EasyNote.Services
{
    2 referencias
    public class MockDataStore : IDataStore<Item>
    {
        readonly List<Item> items;

        0 referencias
        public MockDataStore()
        {
            items = new List<Item>()
            {
                new Item { Id = Guid.NewGuid().ToString(), Text = "First item", Description="This is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Second item", Description="This is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Third item", Description="This is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Fourth item", Description="This is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Fifth item", Description="This is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Sixth item", Description="This is an item description." }
            };
        }

        2 referencias
        public async Task<bool> AddItemAsync(Item item)
        {
            items.Add(item);

            return await Task.FromResult(true);
        }

        1 referencia
        public async Task<bool> UpdateItemAsync(Item item)
```

```
        1 referencia
        public async Task<bool> UpdateItemAsync(Item item)
        {
            var oldItem = items.Where((Item arg) => arg.Id == item.Id).FirstOrDefault();
            items.Remove(oldItem);
            items.Add(item);

            return await Task.FromResult(true);
        }

        1 referencia
        public async Task<bool> DeleteItemAsync(string id)
        {
            var oldItem = items.Where((Item arg) => arg.Id == id).FirstOrDefault();
            items.Remove(oldItem);

            return await Task.FromResult(true);
        }

        2 referencias
        public async Task<Item> GetItemAsync(string id)
        {
            return await Task.FromResult(items.FirstOrDefault(s => s.Id == id));
        }

        2 referencias
        public async Task<IEnumerable<Item>> GetItemsAsync(bool forceRefresh = false)
        {
            return await Task.FromResult(items);
        }
    }
}
```

Login

```
15
16      namespace EasyNote
17      {
            4 referencias
18          public class Login:Base
19          {
20              Attribute
27
28              Properties
61
62              Commands
71
72              Methods
53
54              Constructor
60          }
61      }
62
```

MainActivity.cs

```
4   using Android.Runtime;
5   using Android.OS;
6   using Plugin.FirebasePushNotification;
7
8   namespace EasyNote.Droid
9   {
10      [Activity(Label = "EasyNote", Icon = "@mipmap/icon", Theme = "@style/MainTheme", MainLauncher = true, ConfigurationChang
            0 referencias
11      public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
12      {
                0 referencias
13          protected override void OnCreate(Bundle savedInstanceState)
14          {
15              base.OnCreate(savedInstanceState);
16              Rg.Plugins.Popup.Popup.Init(this);
17              Xamarin.Essentials.Platform.Init(this, savedInstanceState);
18              global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
19              LoadApplication(new App());
20
21              //NotificationCenter.NotifyNotification(Intent);
22              FirebasePushNotificationManager.ProcessIntent(this, Intent);
23          }
                0 referencias
24          public override void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Conte
25          {
26              Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);
27
28              base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
29          }
30      }
31  }
0 %   No se encontraron problemas.   Línea: 11   Carácter: 30   SPC   CRL
```

Application.cs

```
namespace EasyNote.Droid
{
    [Application]
    1 referencia
    public class MainApplication : Application
    {
        0 referencias
        public MainApplication(IntPtr handle, JniHandleOwnership transer) : base(handle, transer)
        {
        }

        0 referencias
        public override void OnCreate()
        {
            base.OnCreate();

            //Set the default notification channel for your app when running Android Oreo
            if (Build.VERSION.SdkInt >= Android.OS.BuildVersionCodes.O)
            {
                //Change for your default notification channel id here
                FirebasePushNotificationManager.DefaultNotificationChannelId = "FirebasePushNotificationChannel";

                //Change for your default notification channel name here
                FirebasePushNotificationManager.DefaultNotificationChannelName = "General";

                FirebasePushNotificationManager.DefaultNotificationChannelImportance = NotificationImportance.Max;
            }

            //If debug you should reset the token each time.
#if DEBUG
            FirebasePushNotificationManager.Initialize(this, true);
```

No se encontraron problemas. Línea: 24 Carácter: 29 SPC

```
Android  EasyNote.Droid.MainApplication  OnCreate()

            //Set the default notification channel for your app when running Android Oreo
            if (Build.VERSION.SdkInt >= Android.OS.BuildVersionCodes.O)
            {
                //Change for your default notification channel id here
                FirebasePushNotificationManager.DefaultNotificationChannelId = "FirebasePushNotificationChannel";

                //Change for your default notification channel name here
                FirebasePushNotificationManager.DefaultNotificationChannelName = "General";

                FirebasePushNotificationManager.DefaultNotificationChannelImportance = NotificationImportance.Max;
            }

            //If debug you should reset the token each time.
#if DEBUG
            FirebasePushNotificationManager.Initialize(this, true);
#else
                    FirebasePushNotificationManager.Initialize(this, false);
#endif

            //Handle notification when app is closed here
            CrossFirebasePushNotification.Current.OnNotificationReceived += (s, p) =>
            {

            };
        }
    }
}
```

View models

UNotasViewModel.cs

```
namespace EasyNote.ViewModels
{
    3 referencias
    public class UNotasViewModel
    {
        0 referencias
        public string notasId { get; set; }
        0 referencias
        public string NotasDescrip { get; set; }
        0 referencias
        public string notas_Image { get; set; }
        1 referencia
        public string notas_correo { get; set; }
        // BaseFirebase firebaseHelper = new BaseFirebase();
        FirebaseHelper services;

        //public ICommand RemoveEmployeeCommand => new Command(RemoveEmployee);
        0 referencias
        private Command AddNotasCommand { get; }
        private ObservableCollection<UNotas> _ubicaciones = new ObservableCollection<UNotas>(
        0 referencias
        public ObservableCollection<UNotas> Employees { get; set; }
        0 referencias
        public ObservableCollection<string> Employeess { get; set; }
        0 referencias
        public string SelectedEmployee { get; set; }

        3 referencias
        public ObservableCollection<UNotas> UNotas
        {
            get { return _ubicaciones; }
            set
```

EasyNote.ViewModels.UNotasViewModel | notasId

```
        0 referencias
        public ObservableCollection<string> Employeess { get; set; }
        0 referencias
        public string SelectedEmployee { get; set; }

        3 referencias
        public ObservableCollection<UNotas> UNotas
        {
            get { return _ubicaciones; }
            set
            {
                _ubicaciones = value;
                // OnPropertyChanged();
            }
        }
        //public ObservableCollection<Models.Ubicaciones>Products { get; set; }

        1 referencia
        public UNotasViewModel()
        {
            services = new FirebaseHelper();
            // UNotas = services.getUNotas();
            UNotas = services.getUNotas21(notas_correo);
            //AddNotasCommand = new Command(async () => await addUNotasAsync(notasId, NotasDescrip, notas_Image));


        }
    }
}
```

NewItemViewModel.cs

```
namespace EasyNote.ViewModels
{
    2 referencias
    public class NewItemViewModel : BaseViewModel
    {
        private string text;
        private string description;

        1 referencia
        public NewItemViewModel()
        {
            SaveCommand = new Command(OnSave, ValidateSave);
            CancelCommand = new Command(OnCancel);
            this.PropertyChanged +=
                (_, __) => SaveCommand.ChangeCanExecute();
        }

        1 referencia
        private bool ValidateSave()
        {
            return !String.IsNullOrWhiteSpace(text)
                && !String.IsNullOrWhiteSpace(description);
        }

        1 referencia
        public string Text
        {
            get => text;
            set => SetProperty(ref text, value);
        }
```

No se encontraron problemas.

```
        }

        2 referencias
        public Command SaveCommand { get; }
        1 referencia
        public Command CancelCommand { get; }

        1 referencia
        private async void OnCancel()
        {
            // This will pop the current page off the navigation stack
            await Shell.Current.GoToAsync("..");
        }

        1 referencia
        private async void OnSave()
        {
            Item newItem = new Item()
            {
                Id = Guid.NewGuid().ToString(),
                Text = Text,
                Description = Description
            };

            await DataStore.AddItemAsync(newItem);

            // This will pop the current page off the navigation stack
            await Shell.Current.GoToAsync("..");
        }
    }
}
```

LoginViewModel.cs

```
using EasyNote.Views;
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;


namespace EasyNote.ViewModels
{
    1 referencia
    public class LoginViewModel : BaseViewModel
    {
        1 referencia
        public Command LoginCommand { get; }

        0 referencias
        public LoginViewModel()
        {
            LoginCommand = new Command(OnLoginClicked);
        }

        1 referencia
        private async void OnLoginClicked(object obj)
        {
            // Prefixing with `//` switches to a different navigation stack instead of pushing to the active
            await Shell.Current.GoToAsync($"//{nameof(AboutPage)}");
        }
    }
}

```

ItemsViewModel.cs

```
namespace EasyNote.ViewModels
{
    5 referencias
    public class ItemsViewModel : BaseViewModel
    {
        private Item _selectedItem;

        3 referencias
        public ObservableCollection<Item> Items { get; }
        1 referencia
        public Command LoadItemsCommand { get; }
        1 referencia
        public Command AddItemCommand { get; }
        1 referencia
        public Command<Item> ItemTapped { get; }

        2 referencias
        public ItemsViewModel()
        {
            Title = "Browse";
            Items = new ObservableCollection<Item>();
            LoadItemsCommand = new Command(async () => await ExecuteLoadItemsCommand());

            ItemTapped = new Command<Item>(OnItemSelected);

            AddItemCommand = new Command(OnAddItem);
        }

        1 referencia
        async Task ExecuteLoadItemsCommand()
        {
```

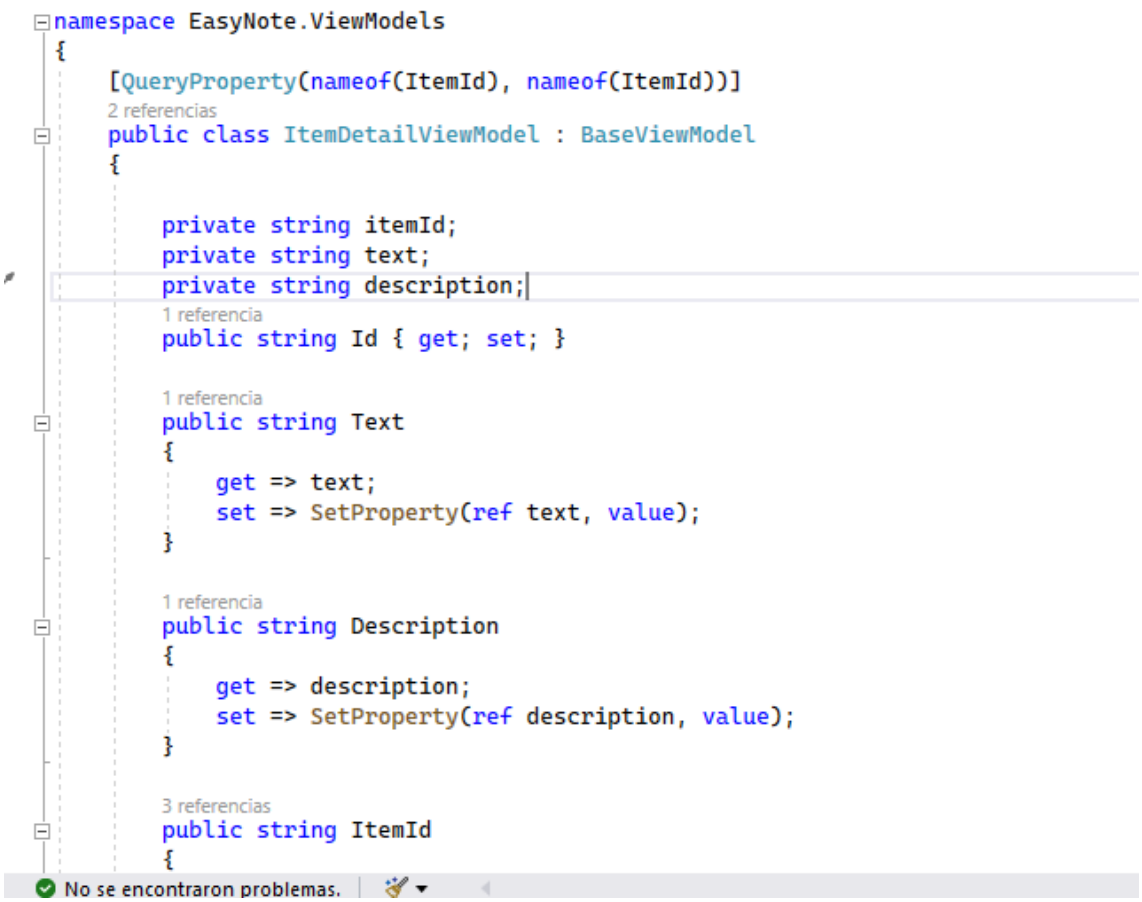
No se encontraron problemas. Línea: 19


1 referencia
async Task ExecuteLoadItemsCommand()
{
IsBusy = true;
try
{
Items.Clear();
var items = await DataStore.GetItemsAsync(true);
foreach (var item in items)
{
Items.Add(item);
}
}
catch (Exception ex)
{
Debug.WriteLine(ex);
}
finally
{
IsBusy = false;
}
}
2 referencias
public void OnAppearing()
{
IsBusy = true;
SelectedItem = null;
}
No se encontraron problemas.

```
2 referencias
public void OnAppearing()
{
    IsBusy = true;
    SelectedItem = null;
}

1 referencia
public Item SelectedItem
{
    get => _selectedItem;
    set
    {
        SetProperty(ref _selectedItem, value);
        OnItemSelected(value);
    }
}

1 referencia
private async void OnAddItem(object obj)
{
    await Shell.Current.GoToAsync(nameof(NewItemPage));
}

2 referencias
async void OnItemSelected(Item item)
{
    if (item == null)
        return;

    // This will push the ItemDetailPage onto the navigation stack
```

No se encontraron problemas

ItemDetailViewModel.cs



```
namespace EasyNote.ViewModels
{
    [QueryProperty(nameof(ItemId), nameof(ItemId))]
    2 referencias
    public class ItemDetailViewModel : BaseViewModel
    {

        private string itemId;
        private string text;
        private string description;
        1 referencia
        public string Id { get; set; }

        1 referencia
        public string Text
        {
            get => text;
            set => SetProperty(ref text, value);
        }

        1 referencia
        public string Description
        {
            get => description;
            set => SetProperty(ref description, value);
        }

        3 referencias
        public string ItemId
        {
```

```
        3 referencias
        public string ItemId
        {
            get
            {
                return itemId;
            }
            set
            {
                itemId = value;
                LoadItemId(value);
            }
        }

        1 referencia
        public async void LoadItemId(string itemId)
        {
            try
            {
                var item = await DataStore.GetItemAsync(itemId);
                Id = item.Id;
                Text = item.Text;
                Description = item.Description;
            }
            catch (Exception)
            {
                Debug.WriteLine("Failed to Load Item");
            }
        }
    }
```

No se encontraron problemas.

ENotasViewModel.cs

```
namespace EasyNote.ViewModels
{
    3 referencias
    public class ENotasViewModel
    {
        0 referencias
        public string notasId { get; set; }
        0 referencias
        public string NotasDescrip { get; set; }
        0 referencias
        public string notas_Image { get; set; }

        // BaseFirebase firebaseHelper = new BaseFirebase();
        FirebaseHelper services;

        //public ICommand RemoveEmployeeCommand => new Command(RemoveEmployee);
        0 referencias
        private Command AddNotasCommand { get; }
        private ObservableCollection<ENotas> _ubicaciones = new ObservableCollection<ENotas>();
        0 referencias
        public ObservableCollection<ENotas> Employees { get; set; }
        0 referencias
        public ObservableCollection<string> Employeess { get; set; }
        0 referencias
        public string SelectedEmployee { get; set; }

        3 referencias
        public ObservableCollection<ENotas> ENotas
        {
            get { return _ubicaciones; }
            set
            {
```

No se encontraron problemas. Línea

```
0 referencias
public ObservableCollection<string> Employeess { get; set; }
0 referencias
public string SelectedEmployee { get; set; }

3 referencias
public ObservableCollection<ENotas> ENotas
{
    get { return _ubicaciones; }
    set
    {
        _ubicaciones = value;
        // OnPropertyChanged();
    }
}
//public ObservableCollection<Models.Ubicaciones>Products { get; set; }

1 referencia
public ENotasViewModel()
{
    services = new FirebaseHelper();
    ENotas = services.getNotas();
    //AddNotasCommand = new Command(async () => await addUNotasAsync(notasId, NotasDescrip, notas_Image));


}


}
}
```

BaseViewModel.cs

```
namespace EasyNote.ViewModels
{
    5 referencias
    public class BaseViewModel : INotifyPropertyChanged
    {
        3 referencias
        public IDataStore<Item> DataStore => DependencyService.Get<IDataStore<Item>>();

        bool isBusy = false;
        3 referencias
        public bool IsBusy
        {
            get { return isBusy; }
            set { SetProperty(ref isBusy, value); }
        }

        string title = string.Empty;
        2 referencias
        public string Title
        {
            get { return title; }
            set { SetProperty(ref title, value); }
        }

        7 referencias
        protected bool SetProperty<T>(ref T backingStore, T value,
            [CallerMemberName] string propertyName = "",
            Action onChanged = null)
        {
            if (EqualityComparer<T>.Default.Equals(backingStore, value))
```

No se encontraron problemas. Línea: 24

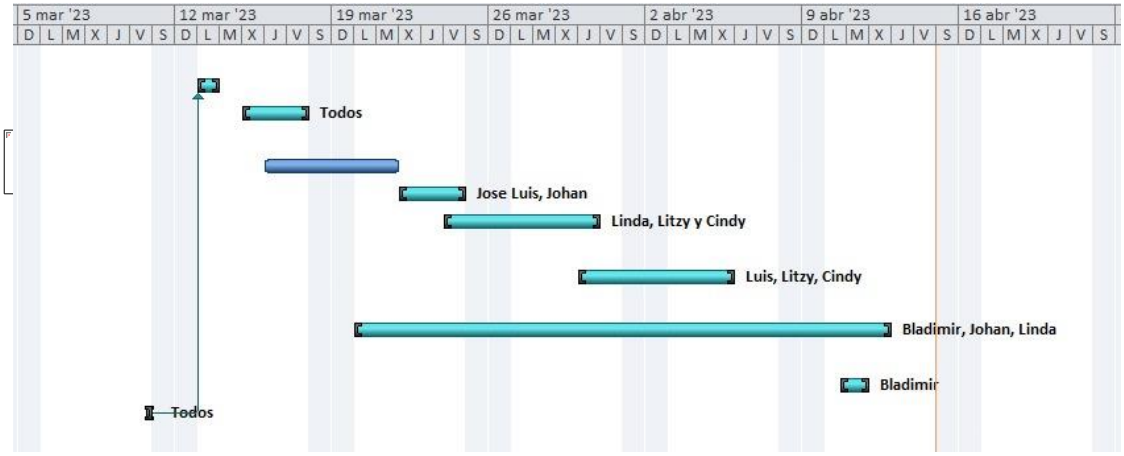
```
        string title = string.Empty;
        2 referencias
        public string Title
        {
            get { return title; }
            set { SetProperty(ref title, value); }
        }

        7 referencias
        protected bool SetProperty<T>(ref T backingStore, T value,
            [CallerMemberName] string propertyName = "",
            Action onChanged = null)
        {
            if (EqualityComparer<T>.Default.Equals(backingStore, value))
                return false;

            backingStore = value;
            onChanged?.Invoke();
            OnPropertyChanged(propertyName);
            return true;
        }

        INotifyPropertyChanged
    }
}
```

AboutViewModel.cs

```
namespace EasyNote.ViewModels
{
    1 referencia
    public class AboutViewModel : BaseViewModel
    {
        0 referencias
        public AboutViewModel()
        {
            Title = "About";
            OpenWebCommand = new Command(async () => await Browser.OpenAsync("https://aka.ms/xamarin-quickstart"));
        }

        1 referencia
        public ICommand OpenWebCommand { get; }
    }
}
```

PLANIFICACION DE ACTIVIDADES

Modo de	Nombre de tarea	Duración	Comienzo	Fin
	Proyecto de Aplicación Movil	29 días	lun 26/12/22	jue 2/2/23
	FASE DE INVESTIGACION	1 día	lun 13/3/23	lun 13/3/23
	Reunir Informacion y validarla para deinir los objetivos y demandas	3 días	mié 15/3/23	vie 17/3/23
	FASE DE DISEÑO	4 días	jue 16/3/23	mar 21/3/23
	Creacion del Wireframe	3 días	mié 22/3/23	vie 24/3/23
	Diseño de la interfaz	5 días	vie 24/3/23	jue 30/3/23
	FASE DE DESARROLLO	10 días	vie 27/1/23	jue 9/2/23
	Desarrollo de diagrama y creacion de datos	5 días	jue 30/3/23	mié 5/4/23
	Desarrollo de codigo fuente	18 días	lun 20/3/23	mié 12/4/23
	FASE DE IMPLEMENTACION	5 días	vie 3/2/23	jue 9/2/23
	Generar APK	9 horas	lun 10/4/23	mar 11/4/23
	Descargar la aplicación	1 hora	vie 10/3/23	vie 10/3/23



Nombres de los recursos
Todos
Jose Luis, Johan
Linda, Litzy y Cindy
Luis, Litzy, Cindy
Bladimir, Johan, Linda
Bladimir
Todos

Conclusiones

- A partir del estudio realizado se puede determinar que desde el punto el proyecto cuenta con un alto potencial, donde en general lo que se busca es un servicio de nota y audios y una gran complejidad en el sistema.

- Otro aspecto importante era la calidad y la rapidez del servicio que se vio como una oportunidad para nuestra aplicación Easy Note el cual se implementa el sistema de notas y en el cual también se graban notas de audio.

- La aplicación cuenta con una facilidad para poder agregar imágenes ya sea directamente de la galería o poder tomar la fotografía haciendo uso de la cámara del celular de una manera sutil.

- Dado esto nuestro proyecto busca tener un servicio para atender y satisfacer las oportunidades detectadas, mediante nuestra aplicación ¨Easy Note¨.

- Se puede concluir que nuestro proyecto busca en enfocarse en el envío de notas y audios, en el cual también se puede compartir mediante la aplicación WhatsApp la nota que se encuentra guardada y esperemos en un futuro implementar nuevos aspectos al proyecto.

BIBLIOGRAFÍA

(ForosDelWeb, 2021)

(StackExchangeInc, 2021)

(developers, 2021)