using System.Collections.Generic; // Importa la biblioteca para manejar colecciones genéricas como listas y diccionarios.

using UnityEngine; // Importa la biblioteca de Unity para la manipulación de componentes y objetos del juego.

using Firebase.Firestore; // Importa la biblioteca de Firestore de Firebase, que permite interactuar con la base de datos en la nube.

using Firebase.Extensions; // Importa las extensiones necesarias para trabajar con Firebase y Unity.

using UnityEngine.UI; // Importa componentes de UI de Unity, como InputFields y Texts.

using System; // Importa las funciones básicas del sistema, como la fecha y hora.

using TMPro; // Importa la biblioteca de TextMeshPro, que proporciona más funcionalidades para trabajar con texto en Unity.

public class FirebaseManager : MonoBehaviour // Declara la clase FirebaseManager, que hereda de MonoBehaviour para interactuar con Unity.

{

    FirebaseFirestore db; // Declara la instancia de la base de datos Firestore.

    public secuenciaManager secuencia; // Declara una referencia a un objeto 'secuenciaManager' (que probablemente gestiona las secuencias de ejercicios).

    public InputField inputNombre; // Referencia al InputField donde se ingresa el nombre del paciente.

    public InputField inputCedula; // Referencia al InputField donde se ingresa la cédula del paciente.

public InputField inputNota; // Referencia al InputField donde se ingresa la nota del fisioterapeuta (nuevo campo).

public Text mensajeOutput; // Referencia a un Text para mostrar mensajes de salida (como advertencias).

public GameObject Registroexitoso; // Referencia a un GameObject que representa el panel de registro exitoso.

public GameObject Registroexistente; // Referencia a un GameObject que representa el panel cuando el paciente ya está registrado.

public Text textoRegistroExitoso; // Referencia al texto que se muestra en el panel de registro exitoso.

public Text textoRegistroExistente; // Referencia al texto que se muestra en el panel cuando el paciente ya existe.

private string pacienteActivoCedula; // Almacena la cédula del paciente actualmente activo.

public GameObject panelError; // Referencia a un GameObject que muestra un panel de error.

public TextMeshProUGUI textoError; // Referencia al texto de error mostrado en el panel de error.

public Button botonReintentar; // Referencia al botón que permite reintentar una acción (por ejemplo, guardar la sesión).

public GameObject panelExito; // Referencia a un GameObject que muestra un panel de éxito.

public TextMeshProUGUI textoExito; // Referencia al texto de éxito mostrado en el panel de éxito.

public GameObject canvasInicio; // Referencia al Canvas principal del menú de inicio.

public GameObject canvasPaneles; // Referencia al Canvas que contiene los paneles de éxito y error.

public GameObject canvasemg; // Referencia a un Canvas relacionado con los paneles de EMG.

void Start() // Método que se ejecuta al inicio de la escena.

{

db = FirebaseFirestore.DefaultInstance; // Inicializa la base de datos Firestore.

}

public void RegistrarPaciente() // Método que registra a un paciente en la base de datos.

{

string nombreIngresado = inputNombre.text; // Obtiene el nombre ingresado por el usuario.

string cedula = inputCedula.text; // Obtiene la cédula ingresada por el usuario.

if (string.IsNullOrEmpty(nombreIngresado) || string.IsNullOrEmpty(cedula)) // Si alguno de los campos está vacío.

```
{

    mensajeOutput.text = "Por favor, complete todos los campos antes de registrar.";  // Muestra un mensaje de error.

    return;  // Sale del método si los campos no están completos.

}


DocumentReference docRef = db.Collection("pacientes").Document(cedula);  // Crea una referencia al documento del paciente con la cédula.

docRef.GetSnapshotAsync().ContinueWithOnMainThread(task =>  // Obtiene los datos del documento de Firestore de forma asincrónica.

{

    if (task.Result.Exists)  // Si el paciente ya existe en la base de datos.

    {

        Dictionary<string, object> pacienteData = task.Result.ToDictionary();  // Convierte los datos del paciente en un diccionario.

        string nombreRegistrado = pacienteData.ContainsKey("Nombre") ? pacienteData["Nombre"].ToString() : "Desconocido";  // Obtiene el nombre registrado.


        pacienteActivoCedula = cedula;  // Almacena la cédula del paciente activo.


        textoRegistroExistente.text = $"El paciente ha sido registrado anteriormente: \"{nombreRegistrado}\", ¿Desea usar este perfil?";  // Muestra un mensaje indicando que el paciente ya está registrado.
```

```
            Registroexistente.SetActive(true);  // Muestra el panel de registro existente.

        }

        else  // Si el paciente no está registrado en la base de datos.

        {

            Dictionary<string, object> pacienteData = new Dictionary<string, object>  // Crea un diccionario con los datos del paciente.

            {

                { "Nombre", nombreIngresado },

                { "Cedula", cedula },

                { "CreadoPor", "montesdiazdayana@gmail.com" }  // Agrega el campo 'CreadoPor' para identificar al creador.

            };



            docRef.SetAsync(pacienteData).ContinueWithOnMainThread(task =>  // Guarda los datos del paciente en Firestore de manera asincrónica.

            {

                pacienteActivoCedula = cedula;  // Almacena la cédula del paciente activo.



                textoRegistroExitoso.text = $"El paciente \"{nombreIngresado}\" con número de cédula \"{cedula}\" ha sido registrado con éxito.";  // Muestra un mensaje de éxito.
```

```
                Registroexitoso.SetActive(true);  // Muestra el panel de registro exitoso.

            });

        }

    });

}


public void AlmacenarDatosTerapia(Dictionary<string, object> datosSensores)  // Método que almacena los datos de terapia del paciente.

{

    if (string.IsNullOrEmpty(pacienteActivoCedula))  // Si no hay un paciente activo registrado.

    {

        Debug.LogError("No hay un paciente activo para almacenar la terapia.");  // Muestra un mensaje de error.

        return;  // Sale del método.

    }


    string fechaActual = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");  // Obtiene la fecha y hora actuales.

    string idFinal = fechaActual + "  " + secuencia.tipoEjercicio + "  " + secuencia.ejercicios[secuencia.ejercicioActualIndex] + "  " + secuencia.sesionActual.ToString();  // Crea un identificador único para la sesión.
```

```
DocumentReference docRef = db.Collection("pacientes")  // Obtiene una referencia a la colección de pacientes.

    .Document(pacienteActivoCedula)  // Utiliza la cédula del paciente activo.

    .Collection("Terapias")  // Accede a la subcolección de Terapias del paciente.

    .Document(idFinal);  // Crea un documento con el identificador generado.



docRef.SetAsync(datosSensores).ContinueWithOnMainThread(task =>  // Guarda los datos de los sensores en Firestore.

{

    if (task.IsCompleted && !task.IsFaulted)  // Si la tarea se completó correctamente.

    {

        Debug.Log($"Datos almacenados correctamente en Firebase para el paciente {pacienteActivoCedula}");  // Muestra un mensaje de éxito.

    }

    else  // Si ocurrió un error.

    {

        Debug.LogError($"Error al almacenar datos: {task.Exception}");  // Muestra el error en la consola.

    }

});
```

```
}

public void AlmacenarSesionFirebase(Dictionary<string, object> datosRecoleccion) // Método que almacena la sesión del paciente.

{

if (string.IsNullOrEmpty(pacienteActivoCedula)) // Si no hay un paciente activo registrado.

{

Debug.LogError("No hay un paciente activo para almacenar la sesión."); // Muestra un mensaje de error.

return; // Sale del método.

}

CollectionReference sesionesRef = db.Collection("pacientes") // Obtiene la referencia a la colección de sesiones del paciente.

.Document(pacienteActivoCedula) // Utiliza la cédula del paciente activo.

.Collection("Sesiones"); // Accede a la subcolección de Sesiones del paciente.

sesionesRef.GetSnapshotAsync().ContinueWithOnMainThread(task => // Obtiene un snapshot de las sesiones actuales del paciente.

{
```

```
        if (task.IsFaulted)  // Si ocurrió un error.

        {

            Debug.LogError("Error al obtener las sesiones: " + task.Exception);  // Muestra el error.

            return;  // Sale del método.

        }



        int numeroDeSesiones = task.Result.Count;  // Obtiene el número actual de sesiones almacenadas.

        int nuevaSesionNumero = numeroDeSesiones + 1;  // Calcula el número de la nueva sesión.



        DocumentReference sesionDocRef = sesionesRef.Document("sesion " + nuevaSesionNumero);  // Crea una referencia al documento de la nueva sesión.



sesionDocRef.SetAsync(datosRecoleccion).ContinueWithOnMainThread(setTask => // Guarda los datos de la sesión en Firestore.

        {

            if (setTask.IsCompleted && !setTask.IsFaulted)  // Si la tarea se completó correctamente.

            {

                Debug.Log("Sesión almacenada exitosamente en Firebase para el paciente " + pacienteActivoCedula);  // Muestra un mensaje de éxito.
```

```
            if (panelExito != null)  // Si el panel de éxito está configurado.

            {

                textoExito.text = "Sesión almacenada exitosamente en Firebase para el paciente " + pacienteActivoCedula;  // Muestra un mensaje de éxito.

                panelExito.SetActive(true);  // Muestra el panel de éxito.

            }

        }

        else  // Si ocurrió un error.

        {

            Debug.LogError("Error al almacenar la sesión: " + setTask.Exception);  // Muestra el error.



            if (panelError != null)  // Si el panel de error está configurado.

            {

                textoError.text = "Error al almacenar la sesión: " + setTask.Exception;  // Muestra el mensaje de error.

                panelError.SetActive(true);  // Muestra el panel de error.



                botonReintentar.onClick.RemoveAllListeners();  // Elimina los listeners previos del botón de reintentar.
```

```
                botonReintentar.onClick.AddListener(() =>  // Agrega un listener para reintentar guardar la sesión.

                {

                    secuencia.GuardarSesionCompletaEnFirebase();  // Llama a la función para guardar la sesión nuevamente.

                    panelError.SetActive(false);  // Oculta el panel de error.

                });

            }

        }

    });

});

}
```