```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using Firebase;

using Firebase.Firestore;

using Firebase.Extensions;

using UnityEngine.UI;

using System.Threading.Tasks;


public class ListaPacientesManager : MonoBehaviour

{

    FirebaseFirestore db;

    public GameObject panelPacientes;

    public GameObject buttonPrefab;

    public GameObject panelDetallePaciente;

    public Text nombreText;

    public Text cedulaText;

    public Text terapiaText;
```

```
    public InputField searchField; // Campo de entrada para el criterio de búsqueda

    public Button searchButton;    // Botón para iniciar la búsqueda

    public Button loadPatientsButton; // Botón para cargar los pacientes

    private string currentPatientId; // Guarda el ID del paciente seleccionado

    public GameObject Seguroeliminar; // Referencia al Canvas de la ventana emergente

    public Text textoVentana; // Referencia al texto dinámico de la ventana

    public Button botonSi; // Botón para confirmar eliminación

    public Button botonNo; // Botón para cancelar

    public Button eliminar;

    private string nombrePacienteSeleccionado; // Guarda el nombre del paciente seleccionado

    public GameObject basededatosgeneral;



    // Nuevo panel para las terapias y su Scroll View

    public GameObject panelFechasTerapias;

    public GameObject fechaButtonPrefab;

    public basededatosinterc canvasManager; // Referencia al script de cambio de canvas
```

```
    // Nuevo Text para mostrar la nota de la sesión (única a la sesión)

    public Text textoNotaSesion;



    [SerializeField] private panelesbasededatos panelesScript; // Asignar el script de paneles desde el inspector



    void Start()

    {

        // Inicialización de Firestore

        db = FirebaseFirestore.DefaultInstance;



        // Agregar listeners para botones

        loadPatientsButton.onClick.AddListener(CargarListaPacientes);

        searchButton.onClick.AddListener(BuscarPaciente);

        eliminar.onClick.AddListener(MostrarVentanaEmergente);

    }



    // Método para cargar la lista completa de pacientes

    public void CargarListaPacientes()

    {
```

```
        LoadingScreen.Instance.ShowLoading("Cargando lista de pacientes...");



        // Limpiar la lista de pacientes mostrada previamente

        foreach (Transform child in panelPacientes.transform)

        {

            Destroy(child.gameObject);

        }


        // Obtener la lista de pacientes desde Firestore

db.Collection("pacientes").GetSnapshotAsync().ContinueWithOnMainThread(task =>

        {

            QuerySnapshot snapshot = task.Result;

            foreach (DocumentSnapshot document in snapshot.Documents)

            {

                CrearBotonPaciente(document);

            }

            LoadingScreen.Instance.HideLoading();

        });
```

```
}

// Método para buscar pacientes por nombre o cédula
public void BuscarPaciente()
{
    string criterio = searchField.text.ToLower();
    if (string.IsNullOrEmpty(criterio))
    {
        CargarListaPacientes();
        return;
    }

    // Limpiar la lista de pacientes mostrada previamente
    foreach (Transform child in panelPacientes.transform)
    {
        Destroy(child.gameObject);
    }

    // Buscar en la colección "pacientes" en Firestore
```

```
db.Collection("pacientes").GetSnapshotAsync().ContinueWithOnMainThread(task =>
        {
            QuerySnapshot snapshot = task.Result;
            foreach (DocumentSnapshot document in snapshot.Documents)
            {
                Dictionary<string, object> pacienteData = document.ToDictionary();
                string nombrePaciente = pacienteData.ContainsKey("Nombre") ? pacienteData["Nombre"].ToString().ToLower() : "";
                string cedulaPaciente = document.Id.ToLower();

                // Verificar si el nombre o cédula coinciden con el criterio de búsqueda
                if (nombrePaciente.Contains(criterio) || cedulaPaciente.Contains(criterio))
                {
                    CrearBotonPaciente(document);
                }
            }
        });
    }
```

```
    // Método para crear botones dinámicos de pacientes

    private void CrearBotonPaciente(DocumentSnapshot document)

    {

        Dictionary<string, object> pacienteData = document.ToDictionary();

        string nombrePaciente = pacienteData.ContainsKey("Nombre") ?
pacienteData["Nombre"].ToString() : "Sin nombre";



        // Crear un nuevo botón para el paciente

        GameObject newButton = Instantiate(buttonPrefab, panelPacientes.transform);

        newButton.GetComponentInChildren<Text>().text = nombrePaciente;



        // Asignar la acción para ver los detalles del paciente

        newButton.GetComponent<Button>().onClick.AddListener(() =>

        {

            VerDetallesPaciente(document.Id);

        });

    }
```

```
    // Método para ver los detalles del paciente seleccionado
    public void VerDetallesPaciente(string cedula)
    {
        LoadingScreen.Instance.ShowLoading("Cargando datos del paciente...");
        panelDetallePaciente.SetActive(true);
        currentPatientId = cedula;
        Debug.Log("Paciente seleccionado con ID: " + currentPatientId);

        // Obtener los datos del paciente desde Firestore
        DocumentReference docRef = db.Collection("pacientes").Document(cedula);
        docRef.GetSnapshotAsync().ContinueWithOnMainThread(task =>
        {
            if (task.Result.Exists)
            {
                Dictionary<string, object> pacienteData = task.Result.ToDictionary();
                nombrePacienteSeleccionado = pacienteData.ContainsKey("Nombre") ? pacienteData["Nombre"].ToString() : "Sin nombre";
                string cedulaPaciente = pacienteData.ContainsKey("Cedula") ? pacienteData["Cedula"].ToString() : "Sin cédula";
```

```
            // Mostrar los datos del paciente en la UI
            nombreText.text = "Nombre: " + nombrePacienteSeleccionado;
            cedulaText.text = "Cédula: " + cedulaPaciente;
            terapiaText.text = "Sesiones de terapias:";
            CargarFechasTerapias(cedula);
            LoadingScreen.Instance.HideLoading();
        }
        else
        {
            Debug.LogWarning("No se encontró el paciente con la cédula proporcionada.");
            LoadingScreen.Instance.HideLoading();
        }
    });
}

// Método para cargar las fechas de las terapias del paciente
private void CargarFechasTerapias(string cedula)
```

```
{
    foreach (Transform child in panelFechasTerapias.transform)
    {
        Destroy(child.gameObject);
    }

    // Obtener las fechas de sesiones desde Firestore

db.Collection("pacientes").Document(cedula).Collection("Sesiones").GetSnapshotAs
ync().ContinueWithOnMainThread(task =>
    {
        if (task.IsFaulted)
        {
            Debug.LogError("Error al obtener las sesiones: " + task.Exception);
            return;
        }

        QuerySnapshot snapshot = task.Result;
        foreach (DocumentSnapshot document in snapshot.Documents)
        {
```

```
            string docID = document.Id;

            Dictionary<string, object> sessionData = document.ToDictionary();

            string fecha = sessionData.ContainsKey("fecha") ? sessionData["fecha"].ToString() : "Sin fecha";

            string tipoJuego = sessionData.ContainsKey("tipoJuego") ? sessionData["tipoJuego"].ToString() : "Desconocido";

            string sessionNumber = docID.Replace("sesion", "").Trim();

            string buttonText = $"Sesión {sessionNumber} - {tipoJuego} - {fecha}";



            // Crear un botón para cada sesión de terapia

            GameObject newButton = Instantiate(fechaButtonPrefab, panelFechasTerapias.transform);

            newButton.GetComponentInChildren<Text>().text = buttonText;



            // Asignar la acción para cargar los datos de la sesión seleccionada

            newButton.GetComponent<Button>().onClick.AddListener(() =>

            {

                Debug.Log("Botón de sesión seleccionado: " + buttonText);

                CargarDatosTerapia(currentPatientId, docID);

                if (canvasManager != null)
```

```
            {
                canvasManager.MostrarMpuEbol();
            }
        });
    }
});
}

// Método para cargar los datos de la sesión seleccionada
private void CargarDatosTerapia(string cedula, string sesionId)
{
    LoadingScreen.Instance.ShowLoading("Cargando datos de la sesión...");

    // Obtener los datos de la sesión desde Firestore
    DocumentReference docRef = db.Collection("pacientes").Document(cedula)
                                 .Collection("Sesiones").Document(sesionId);
    docRef.GetSnapshotAsync().ContinueWithOnMainThread(task =>
    {
```

```
        if (task.IsFaulted)
        {
            Debug.LogError("Error al obtener la sesión: " + task.Exception);
            LoadingScreen.Instance.HideLoading();
            return;
        }

        DocumentSnapshot snapshot = task.Result;
        if (!snapshot.Exists)
        {
            Debug.LogWarning("No se encontró la sesión solicitada en Firebase.");
            LoadingScreen.Instance.HideLoading();
            return;
        }

        Dictionary<string, object> sessionData = snapshot.ToDictionary();
        string fechaRegistro = sessionData.ContainsKey("fecha") ? sessionData["fecha"].ToString() : "Sin fecha";
        string tipoJuego = sessionData.ContainsKey("tipoJuego") ? sessionData["tipoJuego"].ToString() : "Desconocido";
```

```
        string nota = sessionData.ContainsKey("nota") ? sessionData["nota"].ToString() : "Sin nota";


        // Actualizar el texto de las notas de la sesión
        if (textoNotaSesion != null)
        {
            textoNotaSesion.text = nota;
        }


        // Procesar los ejercicios y repeticiones
        Dictionary<string, Dictionary<string, List<List<float>>>> datosEjercicios =
            new Dictionary<string, Dictionary<string, List<List<float>>>>();
        Dictionary<string, Dictionary<string, int>> repeticionesEjercicios =
            new Dictionary<string, Dictionary<string, int>>();


        if (sessionData.ContainsKey("ejercicios"))
        {
            var ejerciciosDict = sessionData["ejercicios"] as Dictionary<string, object>;
            if (ejerciciosDict != null)
            {
```

```
                foreach (var kv in ejerciciosDict)
                {
                    string nombreEjercicio = kv.Key;
                    var ejercicioData = kv.Value as Dictionary<string, object>;
                    if (ejercicioData == null || !ejercicioData.ContainsKey("series"))
                        continue;
                    var seriesDict = ejercicioData["series"] as Dictionary<string, object>;
                    if (seriesDict == null)
                        continue;
                    datosEjercicios[nombreEjercicio] = new Dictionary<string,
List<List<float>>>();
                    repeticionesEjercicios[nombreEjercicio] = new Dictionary<string,
int>();
                    foreach (var serieKv in seriesDict)
                    {
                        string serieKey = serieKv.Key; // Ahora se almacena como "Serie 1",
"Serie 2", etc.
                        var serieData = serieKv.Value as Dictionary<string, object>;
                        if (serieData == null || !serieData.ContainsKey("datos"))
                            continue;
```

```
                    string datosJson = serieData["datos"].ToString();

                    List<List<float>> rawData =
Newtonsoft.Json.JsonConvert.DeserializeObject<List<List<float>>>(datosJson);

                    datosEjercicios[nombreEjercicio][serieKey] = rawData;



                    int repeticiones = 0;

                    if (serieData.ContainsKey("repeticiones"))

                        repeticiones =
System.Convert.ToInt32(serieData["repeticiones"]);

                    repeticionesEjercicios[nombreEjercicio][serieKey] = repeticiones;

                }

            }

        }

        LoadingScreen.Instance.HideLoading();

    });

}



// Método para mostrar ventana emergente de eliminación

private void MostrarVentanaEmergente()
```

```
    {
        Seguroeliminar.SetActive(true);
        textoVentana.text = "¿Estás seguro de que deseas eliminar al paciente " + nombrePacienteSeleccionado + "?";
    }

    // Método para confirmar eliminación
    public void ConfirmarEliminacion()
    {
        // Llamar a Firestore para eliminar el paciente

db.Collection("pacientes").Document(currentPatientId).DeleteAsync().ContinueWithOnMainThread(task =>
        {
            if (task.IsCompleted)
            {
                Debug.Log("Paciente eliminado exitosamente.");
                // Recargar la lista de pacientes después de la eliminación
                CargarListaPacientes();
                Seguroeliminar.SetActive(false);
```

```
            }
            else
            {
                Debug.LogError("Error al eliminar el paciente.");
            }
        });
    }

    // Método para cancelar la eliminación
    public void CancelarEliminacion()
    {
        Seguroeliminar.SetActive(false);
    }
}
```