using System.Collections.Generic; // Importa el espacio de nombres para las colecciones genéricas como List y Dictionary.

using UnityEngine; // Importa el espacio de nombres de Unity para poder usar clases como MonoBehaviour, GameObject, etc.

using TMPro; // Importa el espacio de nombres de TextMeshPro para trabajar con textos en la interfaz de usuario.

public class DropdownManagerFirebase : MonoBehaviour // Declara la clase que maneja la lógica de los Dropdowns y la interacción con Firebase.

{

[Header("Referencias a Dropdowns")] // Agrega un encabezado en el Inspector de Unity para organizar las variables.

public TMP_Dropdown dropdownEjercicios; // Referencia al componente Dropdown para seleccionar ejercicios.

public TMP_Dropdown dropdownSeries; // Referencia al componente Dropdown para seleccionar series de ejercicios.

[Header("Referencia al script de gráficas o paneles")] // Encabezado para las referencias a scripts de paneles.

public panelesbasededatos panelesScript; // Referencia al script que maneja la visualización de los datos en los paneles.

[Header("Opcional: Texto de resumen de repeticiones")] // Encabezado para la variable opcional de resumen de repeticiones.

public TextMeshProUGUI textoResumenRepeticiones; // Texto que muestra el número de repeticiones.

```
    // Estructura interna para almacenar los datos que vienen de Firebase.
    private Dictionary<string, Dictionary<string, List<List<float>>>> dataFromFirebase
=
        new Dictionary<string, Dictionary<string, List<List<float>>>>();

    // Para almacenar las repeticiones por ejercicio y serie.
    private Dictionary<string, Dictionary<string, int>> repeticionesFromFirebase =
        new Dictionary<string, Dictionary<string, int>>();

    // Listas para controlar los dropdowns.
    private List<string> listaEjercicios = new List<string>(); // Lista que almacena los
nombres de los ejercicios.
    // Key: nombreEjercicio, Value: lista de keys de series (ej. "Serie 1")
    private Dictionary<string, List<string>> seriesPorEjercicio =
        new Dictionary<string, List<string>>(); // Diccionario para mapear ejercicios con
sus respectivas series.

    void Start() // Método de inicialización que se ejecuta al inicio.
    {
```

```
        dropdownEjercicios.onValueChanged.AddListener(delegate { ActualizarDropdownSeries(); }); // Agrega un listener al cambio de valor del Dropdown de ejercicios.

        dropdownSeries.onValueChanged.AddListener(delegate { GraficarDatos(); }); // Agrega un listener al cambio de valor del Dropdown de series.

    }


    /// <summary>

    /// Método principal que recibe la estructura desde Firebase.

    /// </summary>

    public void CargarDatosDesdeFirebase(

        Dictionary<string, Dictionary<string, List<List<float>>>> datosFirebase,

        Dictionary<string, Dictionary<string, int>> repeticionesFirebase)

    {

        dataFromFirebase = datosFirebase; // Asigna los datos de Firebase a la variable interna.

        repeticionesFromFirebase = repeticionesFirebase; // Asigna los datos de repeticiones de Firebase.



        dropdownEjercicios.ClearOptions(); // Limpia las opciones del dropdown de ejercicios.

        dropdownSeries.ClearOptions(); // Limpia las opciones del dropdown de series.
```

```
listaEjercicios.Clear(); // Limpia la lista de ejercicios.

seriesPorEjercicio.Clear(); // Limpia el diccionario de series por ejercicio.



// Itera sobre los ejercicios recibidos desde Firebase y los agrega a los dropdowns.

foreach (var ejercicioEntry in dataFromFirebase)

{

    string ejercicio = ejercicioEntry.Key; // Obtiene el nombre del ejercicio.

    listaEjercicios.Add(ejercicio); // Agrega el ejercicio a la lista de ejercicios.



    if (!seriesPorEjercicio.ContainsKey(ejercicio))

        seriesPorEjercicio[ejercicio] = new List<string>(); // Si no existe una entrada para el ejercicio, la crea.



    // Itera sobre las series asociadas a cada ejercicio y las agrega al diccionario.

    foreach (var serieKey in ejercicioEntry.Value.Keys)

    {

        seriesPorEjercicio[ejercicio].Add(serieKey); // Agrega la serie al diccionario de series por ejercicio.

    }

}
```

```
    dropdownEjercicios.AddOptions(listaEjercicios); // Agrega las opciones de ejercicios al dropdown.

    ActualizarDropdownSeries(); // Actualiza el dropdown de series para el primer ejercicio seleccionado.

  }

  /// <summary>

  /// Llena el dropdown de series en base al ejercicio seleccionado.

  /// </summary>

  private void ActualizarDropdownSeries()

  {

    dropdownSeries.ClearOptions(); // Limpia las opciones del dropdown de series.

    if (listaEjercicios.Count == 0) // Si no hay ejercicios disponibles, no hace nada.

      return;


    int indiceEj = dropdownEjercicios.value; // Obtiene el índice del ejercicio seleccionado.

    if (indiceEj < 0 || indiceEj >= listaEjercicios.Count) // Verifica si el índice es válido.

      return;
```

```
    string ejercicioSeleccionado = listaEjercicios[indiceEj]; // Obtiene el nombre del ejercicio seleccionado.

    if (!seriesPorEjercicio.ContainsKey(ejercicioSeleccionado)) // Si no hay series para este ejercicio, no hace nada.

        return;



    List<string> listaSeries = seriesPorEjercicio[ejercicioSeleccionado]; // Obtiene la lista de series para el ejercicio seleccionado.

    if (listaSeries == null || listaSeries.Count == 0) // Si no hay series disponibles, no hace nada.

        return;



    // Agregamos "Seleccione serie" como opción inicial.

    List<string> opcionesSeries = new List<string>();

    opcionesSeries.Add("Seleccione serie"); // Opción predeterminada de "Seleccione serie".

    opcionesSeries.AddRange(listaSeries); // Agrega las series al dropdown de series.

    dropdownSeries.AddOptions(opcionesSeries); // Actualiza el dropdown con las opciones de series.

    dropdownSeries.value = 0; // Establece el valor predeterminado en la primera opción.

  }
```

```
    /// <summary>

    /// Cuando se selecciona una serie, graficamos los datos.

    /// </summary>

    private void GraficarDatos()

    {

        if (listaEjercicios.Count == 0 || dropdownSeries.options.Count == 0) // Si no hay datos, no hace nada.

            return;



        int indiceEj = dropdownEjercicios.value; // Obtiene el índice del ejercicio seleccionado.

        if (indiceEj < 0 || indiceEj >= listaEjercicios.Count) // Verifica si el índice del ejercicio es válido.

            return;

        string ejercicioSeleccionado = listaEjercicios[indiceEj]; // Obtiene el nombre del ejercicio seleccionado.



        int indiceSerie = dropdownSeries.value; // Obtiene el índice de la serie seleccionada.

        if (indiceSerie < 0 || indiceSerie >= dropdownSeries.options.Count) // Verifica si el índice de la serie es válido.

            return;
```

```
string serieSeleccionada = dropdownSeries.options[indiceSerie].text; // Obtiene el nombre de la serie seleccionada.

if (serieSeleccionada.Equals("Seleccione serie")) // Si no se seleccionó una serie válida, no hace nada.

{

    Debug.Log("No se ha seleccionado una serie válida.");

    return;

}



if (!dataFromFirebase.ContainsKey(ejercicioSeleccionado)) // Si el ejercicio seleccionado no está en los datos de Firebase, no hace nada.

{

    Debug.LogWarning($"El ejercicio '{ejercicioSeleccionado}' no está en dataFromFirebase.");

    return;

}



if (!dataFromFirebase[ejercicioSeleccionado].ContainsKey(serieSeleccionada)) // Si la serie seleccionada no está en los datos de Firebase, no hace nada.

{

    Debug.LogWarning($"La serie '{serieSeleccionada}' no existe en '{ejercicioSeleccionado}'.");
```

```
            return;

        }

        List<List<float>> rawData = dataFromFirebase[ejercicioSeleccionado][serieSeleccionada]; // Obtiene los datos crudos de la serie seleccionada.

        List<List<float>> translatedData = DataTranslator.TranslateData(rawData, ejercicioSeleccionado); // Traduce los datos crudos.

        if (panelesScript != null) // Si el script de paneles no es null, actualiza los paneles con los datos traducidos.

        {

            panelesScript.ActualizarDatos(translatedData);

        }

        if (textoResumenRepeticiones != null) // Si el texto de resumen de repeticiones no es null, actualiza el texto con el número de repeticiones.

        {

            int rep = 0;

            if (repeticionesFromFirebase.ContainsKey(ejercicioSeleccionado) &&

repeticionesFromFirebase[ejercicioSeleccionado].ContainsKey(serieSeleccionada)) // Verifica si hay datos de repeticiones para el ejercicio y la serie seleccionados.
```

```
        {

            rep = repeticionesFromFirebase[ejercicioSeleccionado][serieSeleccionada]; // Obtiene el número de repeticiones.

        }

        textoResumenRepeticiones.text = "Repeticiones: " + rep; // Actualiza el texto con el número de repeticiones.

    }

}



// Método para limpiar los datos de los dropdowns y otras variables.

public void LimpiarDatos()

{

    dropdownEjercicios.ClearOptions(); // Limpia las opciones del dropdown de ejercicios.

    dropdownSeries.ClearOptions(); // Limpia las opciones del dropdown de series.



    listaEjercicios.Clear(); // Limpia la lista de ejercicios.

    seriesPorEjercicio.Clear(); // Limpia el diccionario de series por ejercicio.



    if (textoResumenRepeticiones != null) // Si el texto de repeticiones no es null, lo limpia.
```

```
        {
            textoResumenRepeticiones.text = "Repeticiones: ";
        }

        dataFromFirebase.Clear(); // Limpia los datos internos de Firebase.
        repeticionesFromFirebase.Clear(); // Limpia las repeticiones internas.
    }
}
```