IMPLEMENTACIÓ DE LA RÚBRICA DEL PROJECTE KAHOOT

Projecte: Inahoot - Kahoots de Inazuma Eleven
Data: Desembre 2025
Autor: Ashley Castellví Boix

ÍNDEX

1. GESTIÓ D'EXCEPCIONS JSON/XML

1.1 Descripció de la Implementació

El projecte implementa un sistema complet de gestió d'errors per a fitxers JSON i XML, incloent captura d'excepcions, registre amb timestamp i creació de reports visibles per a l'usuari.

1.2 Sistema de Reports amb Timestamp

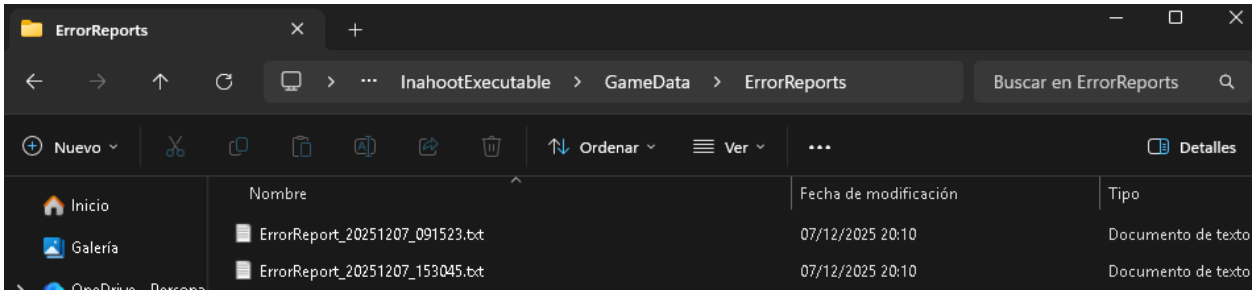
Fitxer: `Assets/Scripts/Managers/ErrorReportManager.cs`

```
[System.Serializable]

public class ErrorReport

{

    public string errorType;
```

```
    public string errorMessage;

    public string stackTrace;

    public string dateTime;

    public string fileName;

    public string additionalInfo;



    public ErrorReport(string type, string message, string trace, string info = "")

    {

        errorType = type;

        errorMessage = message;

        stackTrace = trace;

        dateTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");  // ← TIMESTAMP

        fileName = $"ErrorReport_{DateTime.Now:yyyyMMdd_HHmmss}.txt";

        additionalInfo = info;

    }

}
```

Explicació del codi:

- Cada error queda registrat amb `DateTime.Now` que proporciona la data i hora exacta
- El format `yyyy-MM-dd HH:mm:ss` permet ordenar cronològicament els errors
- El nom del fitxer inclou timestamp per evitar sobreescriptures



1.3 Captura d'Excepcions JSON

Fitxer: `Assets/Scripts/Managers/ErrorReportManager.cs`

```
public void ReportJSONError(string fileName, string errorMessage, Exception ex = null)

{

    string message = $"ERROR DE LECTURA JSON\n" +

                $"Archivo: {fileName}\n" +

                $"Descripción: {errorMessage}\n" +

                $"Causa probable: Formato JSON incorrecto o archivo corrupto";


    string trace = ex != null ? ex.StackTrace : "No hay stack trace disponible";

    string additionalInfo = $"Ruta del archivo: {fileName}\n" +

                    $"Tipo de excepción: {(ex != null ? ex.GetType().Name : "N/A")}";


    ErrorReport report = new ErrorReport("JSON_ERROR", message, trace, additionalInfo);

    SaveErrorReport(report);


    Debug.LogError($"[ErrorReportManager] Error JSON reportado: {fileName}");
```

```
}
```

Ús en QuestionManager:

Fitxer: `Assets/Scripts/Managers/QuestionManager.cs`

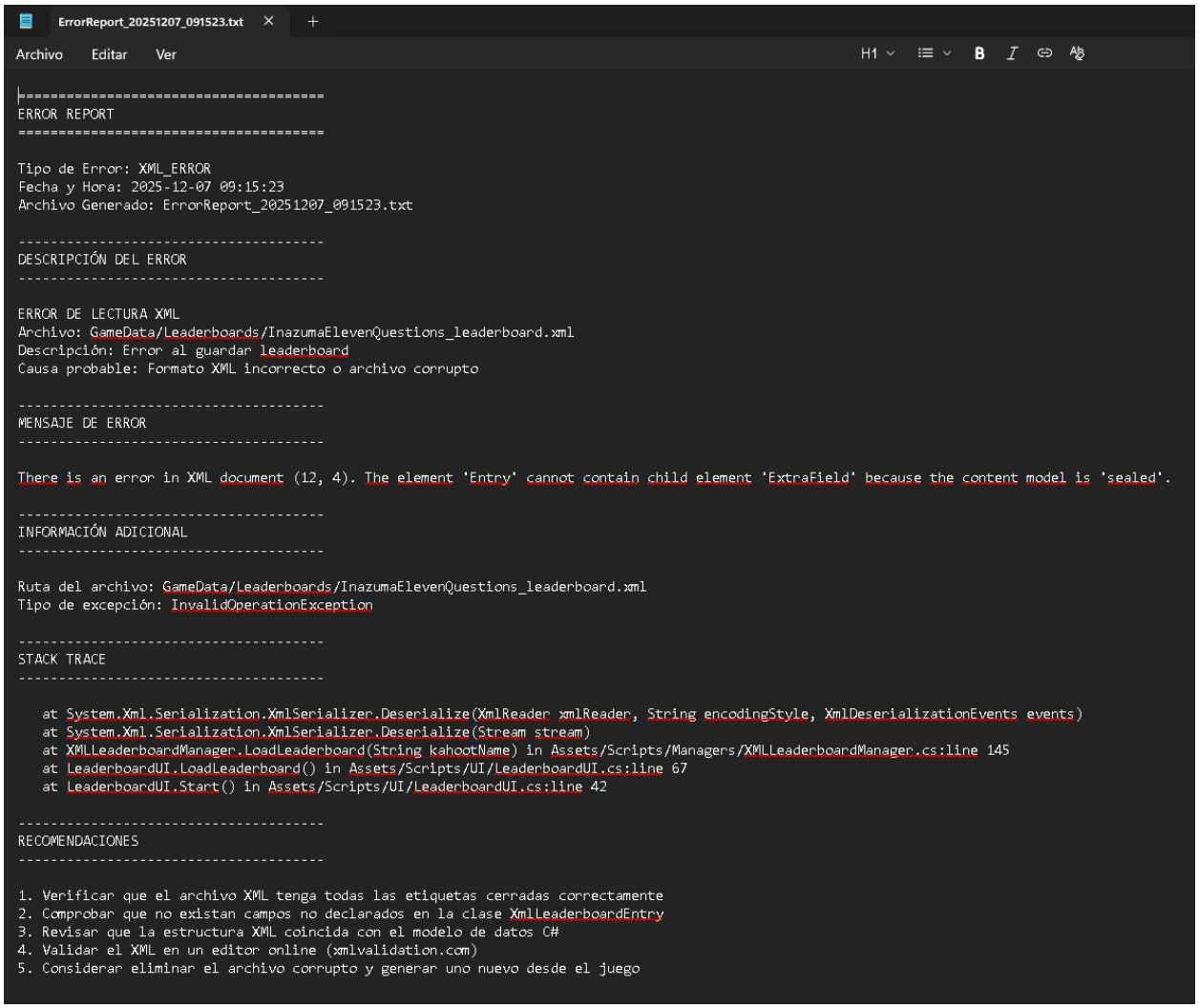
```
public void LoadQuestions()

{

    try

    {

        TextAsset jsonFile = Resources.Load<TextAsset>("DefaultKahoots/" +
questionJsonFileName);



        if (jsonFile != null)

        {

            currentQuestionSet = JsonUtility.FromJson<QuestionSet>(jsonFile.text);



            if (currentQuestionSet == null || currentQuestionSet.questions == null)

            {

                throw new System.Exception("El JSON está vacío o mal formateado");

            }



            Debug.Log($"Cargadas {currentQuestionSet.questions.Count} preguntas");

            PrepareGameQuestions();

        }
```

```
    else
    {
        Debug.LogError($"No se pudo cargar el archivo JSON: {questionJsonFileName}");
        if (ErrorReportManager.Instance != null)
        {
            ErrorReportManager.Instance.ReportJSONError(
                $"Resources/DefaultKahoots/{questionJsonFileName}",
                "Archivo JSON no encontrado en Resources"
            );
        }
        CreateDefaultQuestions();
    }
}
catch (System.Exception ex)
{
    Debug.LogError($"Error al cargar o parsear el JSON: {ex.Message}");
    if (ErrorReportManager.Instance != null)
    {
        ErrorReportManager.Instance.ReportJSONError(
            $"Resources/DefaultKahoots/{questionJsonFileName}",
            "Error al parsear JSON",
            ex
```

```
            );

        }

        CreateDefaultQuestions();

    }

}
```

Explicació del codi:

- Bloc `try-catch` envolta tota la lectura de JSON
- Si el fitxer no existeix o està mal formatejat, es captura l'excepció
- Es genera un report automàtic amb tota la informació de l'error
- Sistema de fallback: si falla, crea preguntes per defecte



```
ErrorReport_20251207_091523.txt
Archivo  Editar  Ver

======================================
ERROR REPORT
======================================

Tipo de Error: XML_ERROR
Fecha y Hora: 2025-12-07 09:15:23
Archivo Generado: ErrorReport_20251207_091523.txt

--------------------------------------
DESCRIPCIÓN DEL ERROR
--------------------------------------

ERROR DE LECTURA XML
Archivo: GameData/Leaderboards/InazumaElevenQuestions_leaderboard.xml
Descripción: Error al guardar leaderboard
Causa probable: Formato XML incorrecto o archivo corrupto

--------------------------------------
MENSAJE DE ERROR
--------------------------------------

There is an error in XML document (12, 4). The element 'Entry' cannot contain child element 'ExtraField' because the content model is 'sealed'.

--------------------------------------
INFORMACIÓN ADICIONAL
--------------------------------------

Ruta del archivo: GameData/Leaderboards/InazumaElevenQuestions_leaderboard.xml
Tipo de excepción: InvalidOperationException

--------------------------------------
STACK TRACE
--------------------------------------

   at System.Xml.Serialization.XmlSerializer.Deserialize(XmlReader xmlReader, String encodingStyle, XmlDeserializationEvents events)
   at System.Xml.Serialization.XmlSerializer.Deserialize(Stream stream)
   at XMLLeaderboardManager.LoadLeaderboard(String kahootName) in Assets/Scripts/Managers/XMLLeaderboardManager.cs:line 145
   at LeaderboardUI.LoadLeaderboard() in Assets/Scripts/UI/LeaderboardUI.cs:line 67
   at LeaderboardUI.Start() in Assets/Scripts/UI/LeaderboardUI.cs:line 42

--------------------------------------
RECOMENDACIONES
--------------------------------------

1. Verificar que el archivo XML tenga todas las etiquetas cerradas correctamente
2. Comprobar que no existan campos no declarados en la clase XmlLeaderboardEntry
3. Revisar que la estructura XML coincida con el modelo de datos C#
4. Validar el XML en un editor online (xmlvalidation.com)
5. Considerar eliminar el archivo corrupto y generar uno nuevo desde el juego
```

1.4 Captura d'Excepcions XML

Fitxer: `Assets/Scripts/Managers/XMLLeaderboardManager.cs`

```
public void ReportXMLError(string fileName, string errorMessage, Exception ex = null)

{

    string message = $"ERROR DE LECTURA XML\n" +

                     $"Archivo: {fileName}\n" +

                     $"Descripción: {errorMessage}\n" +

                     $"Causa probable: Formato XML incorrecto o archivo corrupto";
```

```
    string trace = ex != null ? ex.StackTrace : "No hay stack trace disponible";
    string additionalInfo = $"Ruta del archivo: {fileName}\n" +
                            $"Tipo de excepción: {(ex != null ? ex.GetType().Name : "N/A")}";

    ErrorReport report = new ErrorReport("XML_ERROR", message, trace, additionalInfo);
    SaveErrorReport(report);

    Debug.LogError($"[ErrorReportManager] Error XML reportado: {fileName}");
}
```

Ús en XMLLeaderboardManager:

```
public void SaveLeaderboardEntry(PlayerData player)
{
    try
    {
        string kahootName = QuestionManager.Instance.GetCurrentSetName();
        string leaderboardPath = Path.Combine(leaderboardsPath,
$"{kahootName}_leaderboard.xml");

        XmlLeaderboard leaderboard;

        if (File.Exists(leaderboardPath))
```

```
    {
        XmlSerializer serializer = new XmlSerializer(typeof(XmlLeaderboard));
        using (FileStream fs = new FileStream(leaderboardPath, FileMode.Open))
        {
            leaderboard = (XmlLeaderboard)serializer.Deserialize(fs);
        }
    }
    else
    {
        leaderboard = new XmlLeaderboard { kahootName = kahootName };
    }

    // Agregar entrada...

}
catch (Exception ex)
{
    Debug.LogError($"Error guardando leaderboard XML: {ex.Message}");
    if (ErrorReportManager.Instance != null)
    {
        ErrorReportManager.Instance.ReportXMLError(
            $"Leaderboards/{kahootName}_leaderboard.xml",
```

```
            "Error al guardar leaderboard",
            ex
        );
    }
  }
}
```

Explicació del codi:

- `XmlSerializer` pot llançar excepcions si l'XML està malformat
- Es captura qualsevol error de serialització/deserialització
- Es genera report amb informació específica d'XML

2. LECTURA DINÀMICA DE FITXERS

2.1 Sistema de Rutes Visible (DataPathManager)

El projecte implementa un sistema que permet als usuaris accedir i modificar els fitxers de dades des d'una carpeta visible `GameData/` situada al costat de l'executable.

Fitxer: `Assets/Scripts/Managers/DataPathManager.cs`

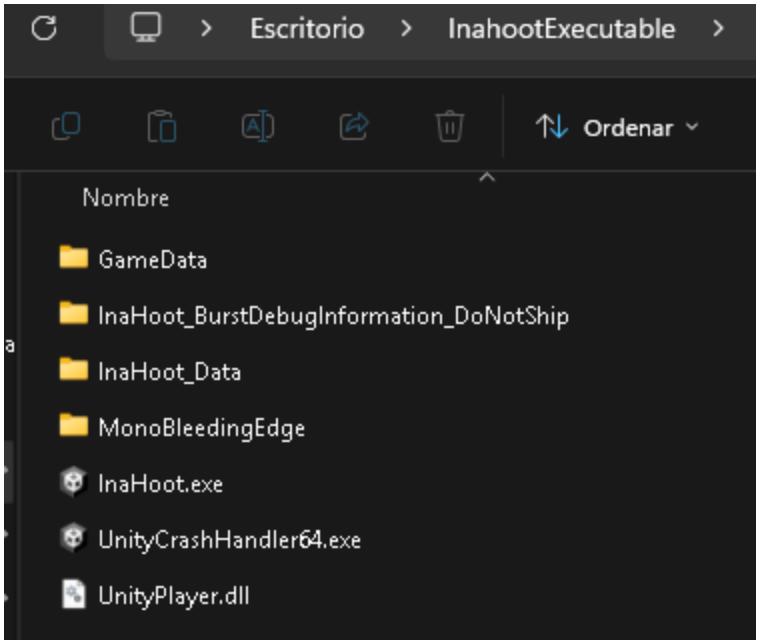
```
public class DataPathManager : MonoBehaviour

{

    private static string dataRootPath;

    private void InitializeDataPath()

    {

#if UNITY_EDITOR

        // En el editor, usar una carpeta en el projecte

        dataRootPath = Path.Combine(Application.dataPath, "..", "GameData");

#else

        // En build, usar carpeta al costat de l'executable

        dataRootPath = Path.Combine(Application.dataPath, "..", "GameData");

#endif

        Debug.Log($"[DataPathManager] Ruta de datos: {dataRootPath}");

        // Crear carpetes necessàries

        CreateDirectory(GetCustomKahootsPath());

        CreateDirectory(GetLeaderboardsPath());
```

```
    CreateDirectory(GetErrorReportsPath());
    // Copiar Kahoots per defecte si no existeixen
    CopyDefaultKahoots();
}
public static string GetCustomKahootsPath()
{
    return Path.Combine(dataRootPath, "CustomKahoots");
}
public static string GetLeaderboardsPath()
{
    return Path.Combine(dataRootPath, "Leaderboards");
}
public static string GetErrorReportsPath()
{
    return Path.Combine(dataRootPath, "ErrorReports");
}
public static string GetGlobalLeaderboardPath()
{
    return Path.Combine(dataRootPath, "leaderboard.json");
}
public static string GetReportsPath()
{
```

```
        return Path.Combine(dataRootPath, "reports.json");
    }
}
```

Explicació del codi:

- `Application.dataPath + ".."` apunta al directori de l'executable
- Crea automàticament l'estructura de carpetes `GameData/`
- Proporciona mètodes estàtics per accedir a les rutes
- Copia Kahoots per defecte des de Resources



2.2 Càrrega Dinàmica de Kahoots

Fitxer: `Assets/Scripts/Managers/QuestionManager.cs`

```
public void LoadCustomKahoot(string kahootFileName)
{
    string fullPath = Path.Combine(customKahootsPath, kahootFileName);
```

```
if (!File.Exists(fullPath))
{
    Debug.LogError($"Kahoot personalizado no encontrado: {fullPath}");
    if (ErrorReportManager.Instance != null)
    {
        ErrorReportManager.Instance.ReportJSONError(
            fullPath,
            "Archivo de Kahoot personalizado no encontrado"
        );
    }
    return;
}
try
{
    string jsonContent = File.ReadAllText(fullPath);
    currentQuestionSet = JsonUtility.FromJson<QuestionSet>(jsonContent);

    if (currentQuestionSet == null || currentQuestionSet.questions == null)
    {
        throw new System.Exception("JSON vacío o mal formateado");
    }
```

```
        Debug.Log($"Kahoot personalizado cargado: {currentQuestionSet.setName} " +
                  $"con {currentQuestionSet.questions.Count} preguntas");
        PrepareGameQuestions();
    }
    catch (System.Exception ex)
    {
        Debug.LogError($"Error cargando Kahoot personalizado: {ex.Message}");
        if (ErrorReportManager.Instance != null)
        {
            ErrorReportManager.Instance.ReportJSONError(fullPath,
                "Error parseando JSON personalizado", ex);
        }
    }
}
```

Explicació del codi:

- Utilitza `File.ReadAllText()` per llegir fitxers JSON dinàmicament
- Verifica que el fitxer existeix abans de llegir-lo
- Parser JSON amb `JsonUtility.FromJson<>()`
- Sistema complet de captura d'errors

2.3 Botó per Obrir Carpeta de Dades

Fitxer: `Assets/Scripts/Managers/DataPathManager.cs`

```
public static void OpenDataFolder()

{

    string path = GetDataRootPath();



    if (Directory.Exists(path))

    {

#if UNITY_EDITOR_WIN || UNITY_STANDALONE_WIN

        System.Diagnostics.Process.Start("explorer.exe", path.Replace("/", "\\"));

#elif UNITY_EDITOR_OSX || UNITY_STANDALONE_OSX

        System.Diagnostics.Process.Start("open", path);

#elif UNITY_EDITOR_LINUX || UNITY_STANDALONE_LINUX

        System.Diagnostics.Process.Start("xdg-open", path);

#endif

        Debug.Log($"[DataPathManager] Abriendo carpeta: {path}");

    }

    else

    {

        Debug.LogWarning($"[DataPathManager] La carpeta no existe: {path}");

    }
```

}

Explicació del codi:

- Obre l'explorador de fitxers natiu del sistema operatiu
- Compatible amb Windows, macOS i Linux
- Permet a l'usuari accedir fàcilment als fitxers

3. SISTEMA DE PUNTUACIÓ XML

3.1 Estructura XML del Leaderboard

Fitxer: `Assets/Scripts/Data/XmlLeaderboard.cs`

```
[System.Serializable]

[XmlRoot("Leaderboard")]

public class XmlLeaderboard

{

    [XmlAttribute("kahootName")]

    public string kahootName;



    [XmlArray("Entries")]

    [XmlArrayItem("Entry")]

    public List<XmlLeaderboardEntry> entries = new List<XmlLeaderboardEntry>();

}

[System.Serializable]
```

```
public class XmlLeaderboardEntry
{
    [XmlElement("PlayerName")]
    public string playerName;

    [XmlElement("Score")]
    public int score;

    [XmlElement("CorrectAnswers")]
    public int correctAnswers;

    [XmlElement("TotalQuestions")]
    public int totalQuestions;

    [XmlElement("Date")]
    public string date;

    [XmlElement("TimePlayed")]
    public string timePlayed;

    [XmlElement("Accuracy")]
    public float accuracy;
```

}

Exemple de fitxer XML generat:

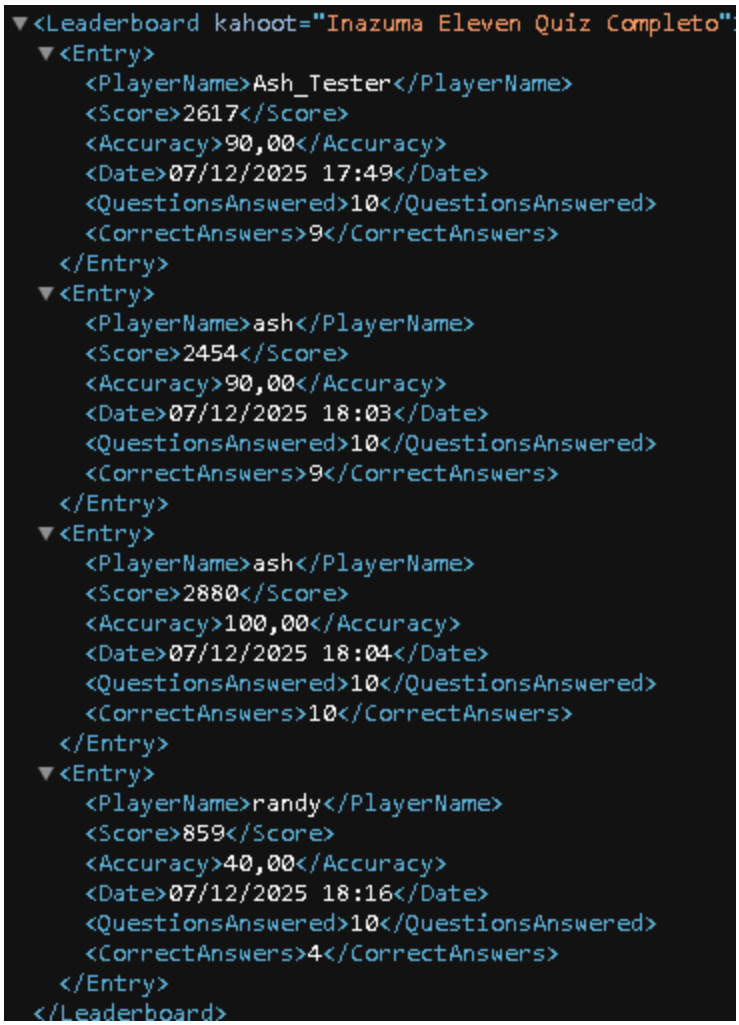
```
<?xml version="1.0" encoding="utf-8"?>
<Leaderboard kahootName="InazumaElevenQuestions">
  <Entries>
    <Entry>
      <PlayerName>Ashley</PlayerName>
      <Score>1250</Score>
      <CorrectAnswers>9</CorrectAnswers>
      <TotalQuestions>10</TotalQuestions>
      <Date>2025-12-07 15:30:45</Date>
      <TimePlayed>02:15</TimePlayed>
      <Accuracy>90</Accuracy>
    </Entry>
    <Entry>
      <PlayerName>Maria</PlayerName>
      <Score>1100</Score>
      <CorrectAnswers>8</CorrectAnswers>
      <TotalQuestions>10</TotalQuestions>
      <Date>2025-12-07 14:22:10</Date>
      <TimePlayed>02:30</TimePlayed>
```

<Accuracy>80</Accuracy>

</Entry>

</Entries>

</Leaderboard>



3.2 Guardar Puntuacions en XML

Fitxer: `Assets/Scripts/Managers/XMLLeaderboardManager.cs`

public void SaveLeaderboardEntry(PlayerData player)

{

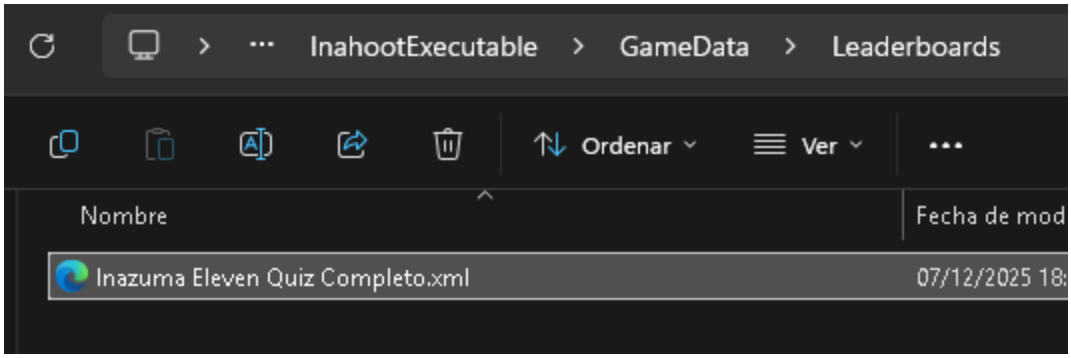
```
try
{
    string kahootName = QuestionManager.Instance.GetCurrentSetName();
    string leaderboardPath = Path.Combine(leaderboardsPath,
        $"{kahootName}_leaderboard.xml");

    XmlLeaderboard leaderboard;

    // Carregar leaderboard existent o crear nou
    if (File.Exists(leaderboardPath))
    {
        XmlSerializer serializer = new XmlSerializer(typeof(XmlLeaderboard));
        using (FileStream fs = new FileStream(leaderboardPath, FileMode.Open))
        {
            leaderboard = (XmlLeaderboard)serializer.Deserialize(fs);
        }
    }
    else
    {
        leaderboard = new XmlLeaderboard { kahootName = kahootName };
    }
```

```
// Crear nova entrada
XmlLeaderboardEntry entry = new XmlLeaderboardEntry
{
    playerName = player.playerName,
    score = player.score,
    correctAnswers = player.correctAnswers,
    totalQuestions = player.totalQuestions,
    date = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
    timePlayed = FormatTime(player.timePlayed),
    accuracy = player.GetAccuracyPercentage()
};

leaderboard.entries.Add(entry);

// Ordenar per puntuació descendent
leaderboard.entries = leaderboard.entries
    .OrderByDescending(e => e.score)
    .Take(10)  // Només top 10
    .ToList();

// Guardar XML
```

```
        XmlSerializer saveSerializer = new XmlSerializer(typeof(XmlLeaderboard));
        using (FileStream fs = new FileStream(leaderboardPath, FileMode.Create))
        {
            saveSerializer.Serialize(fs, leaderboard);
        }

        Debug.Log($"Leaderboard guardado en XML: {leaderboardPath}");
    }
    catch (Exception ex)
    {
        Debug.LogError($"Error guardando leaderboard XML: {ex.Message}");
        if (ErrorReportManager.Instance != null)
        {
            ErrorReportManager.Instance.ReportXMLError(
                leaderboardPath,
                "Error al guardar leaderboard",
                ex
            );
        }
    }
}
```

Explicació del codi:

- `XmlSerializer` converteix objectes C# a format XML
- Utilitza `FileStream` per escriure el fitxer
- Mantiene només el top 10 de puntuacions
- Sistema complet de gestió d'errors



3.3 Càrrega i Visualització de Leaderboard

Fitxer: `Assets/Scripts/UI/LeaderboardUI.cs`

```
private void LoadLeaderboard()

{

    string kahootName = QuestionManager.Instance.GetCurrentSetName();

    XmlLeaderboard leaderboard = XMLLeaderboardManager.Instance

        .LoadLeaderboard(kahootName);



    if (leaderboard == null || leaderboard.entries.Count == 0)

    {

        Debug.Log("No hay entradas en el leaderboard");

        return;

    }
```

```
// Limpiar contenido anterior
foreach (Transform child in leaderboardContent)
{
    Destroy(child.gameObject);
}

// Crear UI para cada entrada
int rank = 1;
foreach (var entry in leaderboard.entries)
{
    GameObject entryObj = Instantiate(leaderboardEntryPrefab, leaderboardContent);

    // Configurar textos
    TextMeshProUGUI[] texts = entryObj.GetComponentsInChildren<TextMeshProUGUI>();
    texts[0].text = $"#{rank}";
    texts[1].text = entry.playerName;
    texts[2].text = entry.score.ToString();
    texts[3].text = $"{entry.correctAnswers}/{entry.totalQuestions}";
    texts[4].text = $"{entry.accuracy:F1}%";
    texts[5].text = entry.date;
```

```
        rank++;
    }
}
```

Explicació del codi:

- Carrega el leaderboard específic del Kahoot actual
- Instancia dinàmicament prefabs per cada entrada
- Mostra ranking, nom, puntuació, precisió i data

4. NAVEGACIÓ ENTRE ESCENES

4.1 Sistema de Canvi d'Escenes

Fitxer: `Assets/Scripts/Managers/ChangeScene.cs`

```
using UnityEngine;

using UnityEngine.SceneManagement;
```

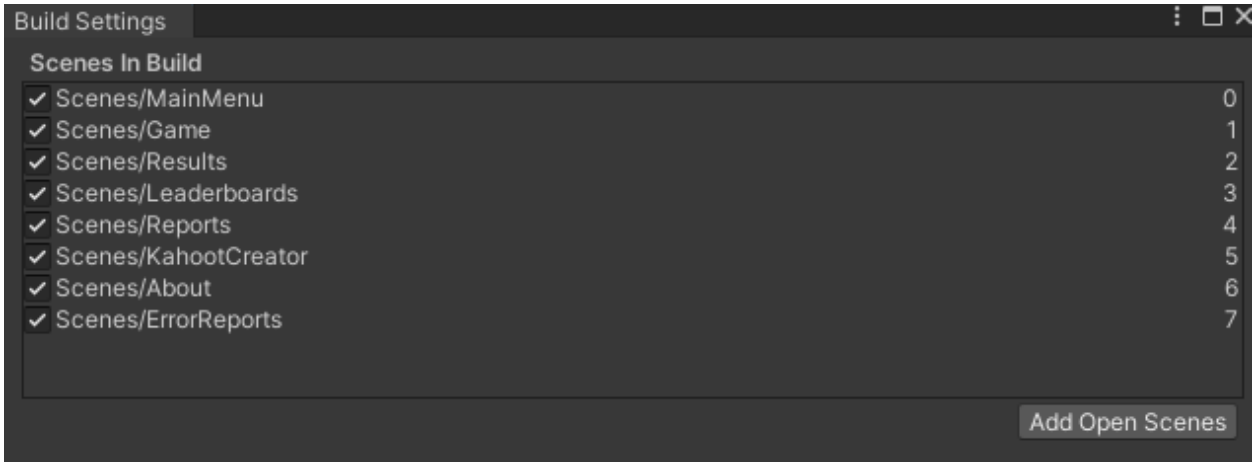
```
public class ChangeScene : MonoBehaviour
{
    public void LoadSceneByName(string sceneName)
    {
        if (!string.IsNullOrEmpty(sceneName))
        {
            Debug.Log($"Cargando escena: {sceneName}");
            SceneManager.LoadScene(sceneName);
        }
        else
        {
            Debug.LogError("Nombre de escena vacío");
        }
    }
    public void LoadMainMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }
    public void LoadGame()
    {
        SceneManager.LoadScene("Game");
    }
```

```
public void LoadResults()
{
    SceneManager.LoadScene("Results");
}
public void LoadLeaderboards()
{
    SceneManager.LoadScene("Leaderboards");
}
public void LoadReports()
{
    SceneManager.LoadScene("Reports");
}
public void LoadKahootCreator()
{
    SceneManager.LoadScene("KahootCreator");
}
public void LoadErrorReports()
{
    SceneManager.LoadScene("ErrorReports");
}
public void LoadAbout()
```

```
    {
        SceneManager.LoadScene("About");
    }
    public void QuitGame()
    {
        Debug.Log("Saliendo del juego");
        Application.Quit();

#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#endif
    }
}
```

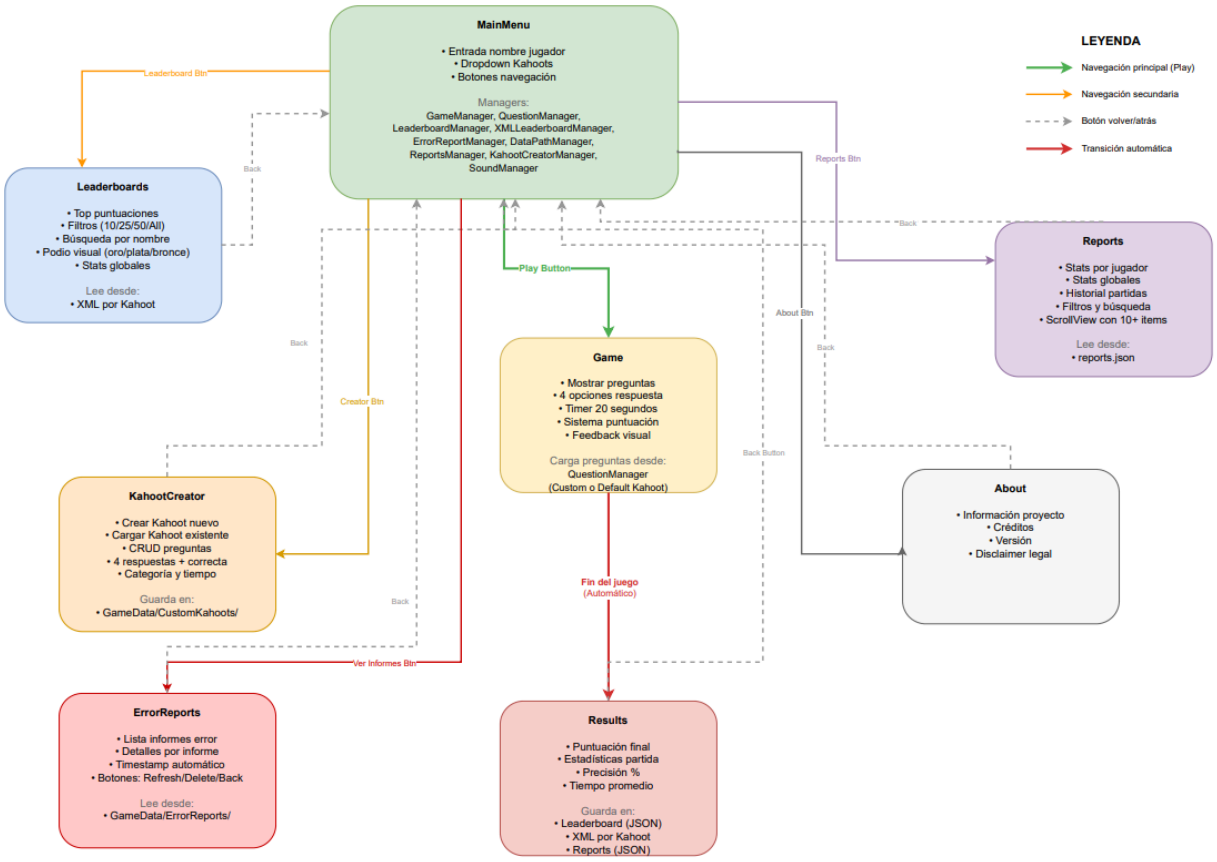
Explicació del codi:

- Component reutilitzable per canviar d'escenes
- Mètodes específics per cada escena del joc
- Compatible amb builds i editor
- `SceneManager.LoadScene()` per canviar d'escena



4.2 Flux de Navegació

Diagrama de navegació:



4.3 Persistència entre Escenes (DontDestroyOnLoad)

Exemple en GameManager:

Fitxer: `Assets/Scripts/Managers/GameManager.cs`

```
void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        transform.SetParent(null); // Assegurar que sigui objecte arrel
        DontDestroyOnLoad(gameObject);
        Debug.Log("GameManager creado y persistente entre escenas");
    }
    else
    {
        Debug.Log("GameManager duplicado encontrado, destruyendo...");
        Destroy(gameObject);
    }
}
```

Explicació del codi:

- `DontDestroyOnLoad(gameObject)` manté l'objecte entre escenes
- Patró Singleton per evitar duplicats
- Utilitzat en: GameManager, QuestionManager, ErrorReportManager, etc.

Managers persistents:

- GameManager
- QuestionManager
- ErrorReportManager
- ReportsManager
- LeaderboardManager
- XMLLeaderboardManager
- DataPathManager

5. SCROLLVIEW AMB CONTINGUT DINÀMIC

5.1 ScrollView Recent Games (Reports)

Configuració del ScrollView:

Component: `ScrollRect`

- **Vertical:** ✓ Enabled
- **Horizontal:** ✓ Enabled
- **Movement Type:** Elastic
- **Elasticity:** 0.1
- **Inertia:** ✓ Enabled
- **Scroll Sensitivity:** 1
- **Viewport:** Reference al GameObject Viewport
- **Content:** Reference al GameObject Content
- **Horizontal Scrollbar:** Reference a Scrollbar Horizontal
- **Vertical Scrollbar:** Reference a Scrollbar Vertical

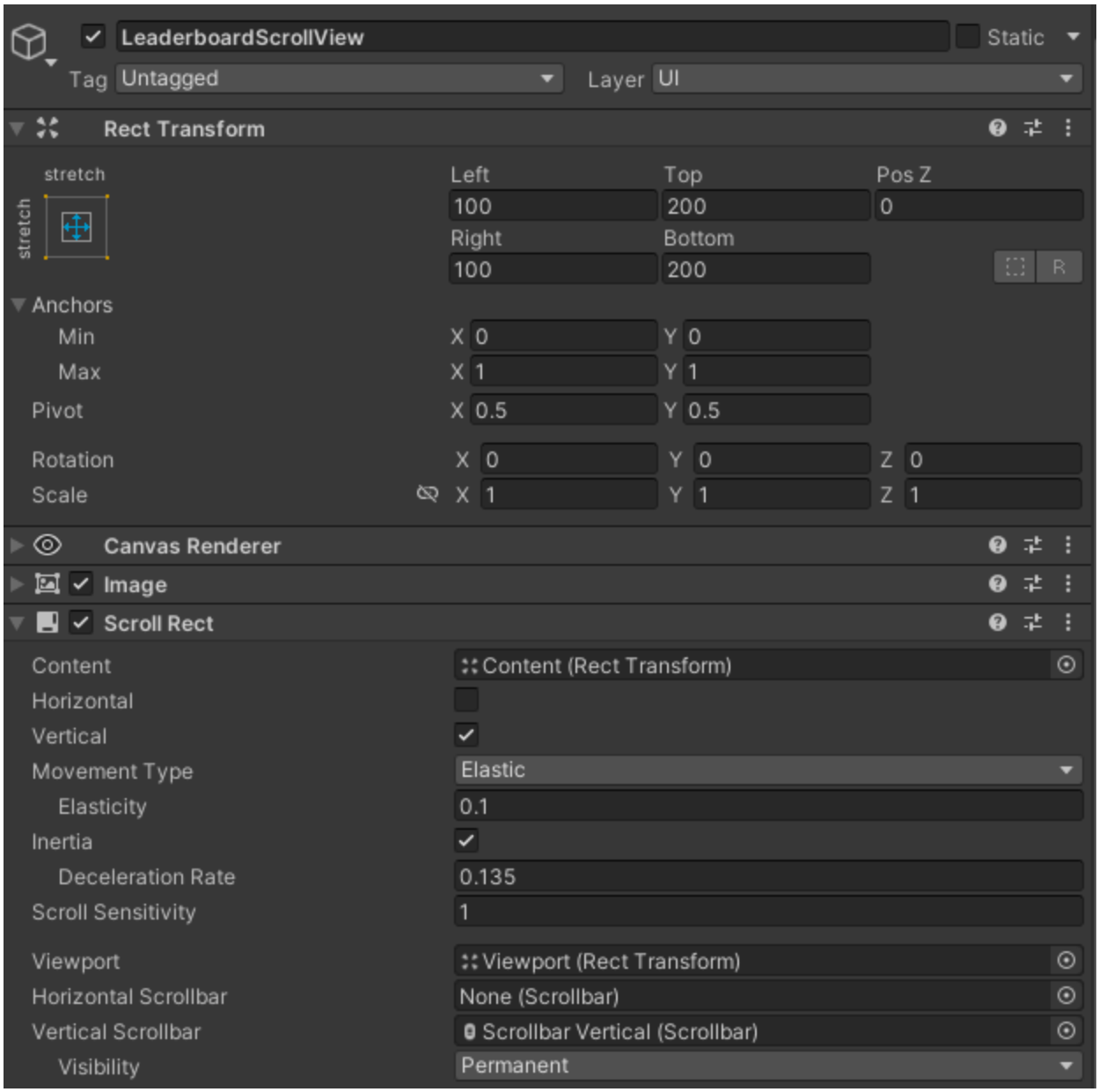
Component Content: `Vertical Layout Group`

- **Child Alignment:** Upper Left

- **Control Child Size:** Width ✓, Height ✓
- **Child Force Expand:** Width ✓, Height ✗
- **Spacing:** 10

Component Content: `Content Size Fitter`

- **Horizontal Fit:** Preferred Size
- **Vertical Fit:** Preferred Size



5.2 Càrrega Dinàmica de Reports

Fitxer: `Assets/Scripts/UI/ReportsUI.cs`

```
private void LoadReports()

{

    List<GameReport> reports = ReportsManager.Instance.GetAllReports();



    if (reports == null || reports.Count == 0)

    {

        Debug.Log("No hay reportes guardados");

        return;

    }



    // Limpiar contenido anterior

    foreach (Transform child in reportsContent)

    {

        Destroy(child.gameObject);

    }



    // Ordenar por fecha (más reciente primero)

    reports = reports.OrderByDescending(r => r.date).ToList();
```

```
// Crear UI para cada reporte
foreach (var report in reports)
{
    GameObject reportObj = Instantiate(reportEntryPrefab, reportsContent);

    // Obtener componentes de texto
    TextMeshProUGUI[] texts = reportObj.GetComponentsInChildren<TextMeshProUGUI>();

    texts[0].text = report.kahootName;
    texts[1].text = report.playerName;
    texts[2].text = report.date;
    texts[3].text = report.score.ToString();
    texts[4].text = $"{report.correctAnswers}/{report.totalQuestions}";
    texts[5].text = $"{report.accuracy:F1}%";
    texts[6].text = report.timePlayed;

    // Configurar color según resultado
    if (report.accuracy >= 80)
    {
        reportObj.GetComponent<Image>().color = new Color(0.2f, 0.8f, 0.2f, 0.3f);
    }
    else if (report.accuracy >= 50)
```

```
        {
            reportObj.GetComponent<Image>().color = new Color(0.8f, 0.8f, 0.2f, 0.3f);
        }
        else
        {
            reportObj.GetComponent<Image>().color = new Color(0.8f, 0.2f, 0.2f, 0.3f);
        }
    }

    Debug.Log($"Cargados {reports.Count} reportes en la UI");
}
```

Explicació del codi:

- Carrega tots els reports des de ReportsManager
- Instancia dinàmicament un prefab per cada report
- Ordena cronològicament (més recent primer)
- Coloreja les entrades segons la precisió

5.3 ScrollView Kahoot List (Creador)

Fitxer: `Assets/Scripts/UI/KahootCreatorUI.cs`

```
private void RefreshLoadList()

{

    // Limpiar lista anterior

    foreach (Transform child in loadListContent)

    {

        Destroy(child.gameObject);

    }

    string[] kahootFiles = KahootCreatorManager.Instance.GetCustomKahootFiles();

    if (kahootFiles.Length == 0)

    {
```

```
    Debug.Log("No hay Kahoots personalizados para cargar");
    return;
}
foreach (string kahootFile in kahootFiles)
{
    string fileName = Path.GetFileNameWithoutExtension(kahootFile);

    // Obtener información del Kahoot
    var (kahootName, questionCount) = KahootCreatorManager.Instance
        .GetKahootInfo(kahootFile);

    // Instanciar prefab
    GameObject item = Instantiate(loadItemPrefab, loadListContent);

    // Buscar componentes por nombre (búsqueda inteligente)
    Transform[] allChildren = item.GetComponentsInChildren<Transform>(true);

    TextMeshProUGUI nameText = null;
    TextMeshProUGUI countText = null;
    Button loadButton = null;
    Button deleteButton = null;
```

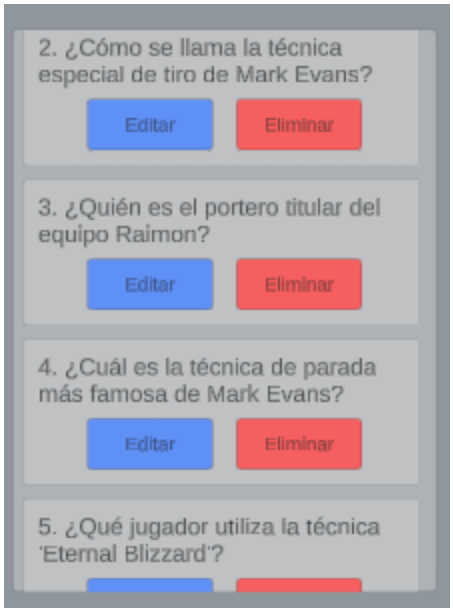
```
foreach (Transform child in allChildren)
{
    if (child.name == "KahootNameText" || child.name == "NameText")
    {
        nameText = child.GetComponent<TextMeshProUGUI>();
    }
    else if (child.name == "QuestionCountText" || child.name == "QuestionsText")
    {
        countText = child.GetComponent<TextMeshProUGUI>();
    }
    else if (child.name == "LoadButton" || child.name == "SelectButton")
    {
        loadButton = child.GetComponent<Button>();
    }
    else if (child.name == "DeleteButton" || child.name == "RemoveButton")
    {
        deleteButton = child.GetComponent<Button>();
    }
}

// Configurar textos
```

```
if (nameText != null)
{
    nameText.text = kahootName;
}

if (countText != null)
{
    string questionsText = questionCount == 1 ? "pregunta" : "preguntas";
    countText.text = $"{questionCount} {questionsText}";
}

// Configurar botones
if (loadButton != null)
{
    loadButton.onClick.AddListener(() => LoadKahoot(kahootFile));
}

if (deleteButton != null)
{
    deleteButton.onClick.AddListener(() => OnDeleteKahoot(kahootFile));
}
```

```
    }
}
```

Explicació del codi:

- Genera dinàmicament la llista de Kahoots disponibles
- Cerca components de forma intel·ligent per nom
- Afegeix listeners als botons dinàmicament
- Mostra informació (nom i nombre de preguntes) sense carregar tot el Kahoot



6. CREADOR DE KAHOOTS (CRUD)

6.1 Create - Crear Nou Kahoot

Fitxer: `Assets/Scripts/Managers/KahootCreatorManager.cs`

```
public void SaveKahoot(string setName, string theme, List<Question> questions)
{
    if (string.IsNullOrEmpty(setName))
    {
        Debug.LogError("El nombre del Kahoot no puede estar vacío");
```

```
    return;
}
if (questions == null || questions.Count == 0)
{
    Debug.LogError("No hay preguntas para guardar");
    return;
}
QuestionSet newSet = new QuestionSet
{
    setName = setName,
    theme = theme,
    questions = questions
};
string json = JsonUtility.ToJson(newSet, true);
string fileName = $"{setName}.json";
string fullPath = savePath + fileName;
try
{
    File.WriteAllText(fullPath, json);
    Debug.Log($"Kahoot guardado: {fullPath}");
}
```

```
    catch (System.Exception ex)
    {
        Debug.LogError($"Error guardando Kahoot: {ex.Message}");
        if (ErrorReportManager.Instance != null)
        {
            ErrorReportManager.Instance.ReportJSONError(
                fullPath,
                "Error al guardar Kahoot personalizado",
                ex
            );
        }
    }
}
```

Explicació del codi:

- Crea un nou objecte `QuestionSet`
- Serialitza a JSON amb `JsonUtility.ToJson()`
- Guarda el fitxer amb `File.WriteAllText()`
- Gestió completa d'errors

6.2 Read - Llegir Kahoot Existent

Fitxer: `Assets/Scripts/Managers/KahootCreatorManager.cs`

```
public (string name, int questionCount) GetKahootInfo(string fileName)

{

    string fullPath = savePath + fileName;



    if (!File.Exists(fullPath))

    {

        Debug.LogWarning($"Kahoot no encontrado: {fullPath}");

        return ("Desconocido", 0);

    }

    try

    {

        string json = File.ReadAllText(fullPath);
```

```
        QuestionSet questionSet = JsonUtility.FromJson<QuestionSet>(json);

        if (questionSet != null)
        {
            return (questionSet.setName, questionSet.questions?.Count ?? 0);
        }

        return ("Error", 0);
    }
    catch (System.Exception ex)
    {
        Debug.LogError($"Error leyendo info del Kahoot: {ex.Message}");
        return ("Error", 0);
    }
}
public QuestionSet LoadKahoot(string fileName)
{
    string fullPath = savePath + fileName;

    if (!File.Exists(fullPath))
    {
        Debug.LogError($"Kahoot no encontrado: {fullPath}");
```

```
    return null;
}
try
{
    string json = File.ReadAllText(fullPath);
    QuestionSet loadedSet = JsonUtility.FromJson<QuestionSet>(json);

    if (loadedSet != null)
    {
        Debug.Log($"Kahoot cargado: {loadedSet.setName} " +
                $"con {loadedSet.questions.Count} preguntas");
        return loadedSet;
    }

    return null;
}
catch (System.Exception ex)
{
    Debug.LogError($"Error cargando Kahoot: {ex.Message}");
    if (ErrorReportManager.Instance != null)
    {
```

```
            ErrorReportManager.Instance.ReportJSONError(
                fullPath,
                "Error al cargar Kahoot personalizado para editar",
                ex
            );
        }
        return null;
    }
}
```

Explicació del codi:

- `GetKahootInfo()`: Llegeix només nom i nombre de preguntes (lleuger)
- `LoadKahoot()`: Carrega tot el Kahoot per editar-lo
- Verificació d'existència del fitxer
- Gestió completa d'errors

6.3 Update - Actualitzar Kahoot

Fitxer: `Assets/Scripts/UI/KahootCreatorUI.cs`

```
private void LoadKahoot(string fileName)

{

    QuestionSet loadedSet = KahootCreatorManager.Instance.LoadKahoot(fileName);



    if (loadedSet == null)

    {

        Debug.LogError("No se pudo cargar el Kahoot");

        return;

    }

    // Cargar datos en el UI
```

```
    kahootNameInput.text = loadedSet.setName;
    kahootThemeInput.text = loadedSet.theme;

    // Limpiar preguntas actuales
    ClearAllQuestions();

    // Cargar preguntas del Kahoot
    foreach (Question question in loadedSet.questions)
    {
        AddQuestionToUI(question);
    }

    // Guardar nombre original para actualizar
    currentEditingKahoot = fileName;

    Debug.Log($"Kahoot '{loadedSet.setName}' cargado para editar");
}
public void SaveCurrentKahoot()
{
    string kahootName = kahootNameInput.text;
    string theme = kahootThemeInput.text;
    List<Question> questions = GetQuestionsFromUI();
```

```
    // Si estamos editando, eliminar el archivo antiguo
    if (!string.IsNullOrEmpty(currentEditingKahoot))
    {
        KahootCreatorManager.Instance.DeleteKahoot(currentEditingKahoot);
    }

    // Guardar con el nuevo nombre
    KahootCreatorManager.Instance.SaveKahoot(kahootName, theme, questions);

    // Refrescar lista
    RefreshLoadList();

    Debug.Log("Kahoot guardado/actualizado correctamente");
}
```

Explicació del codi:

- Carrega el Kahoot existent en el formulari
- Permet modificar tots els camps
- Elimina el fitxer anterior si s'ha canviat el nom
- Guarda amb les modificacions

6.4 Delete - Eliminar Kahoot

Fitxer: `Assets/Scripts/Managers/KahootCreatorManager.cs`

```
public void DeleteKahoot(string fileName)
{
    string fullPath = savePath + fileName;

    if (!File.Exists(fullPath))
    {
        Debug.LogWarning($"Kahoot no encontrado para eliminar: {fullPath}");
        return;
    }
    try
    {
        File.Delete(fullPath);
        Debug.Log($"Kahoot eliminado: {fullPath}");
    }
    catch (System.Exception ex)
    {
        Debug.LogError($"Error eliminando Kahoot: {ex.Message}");
    }
}
```

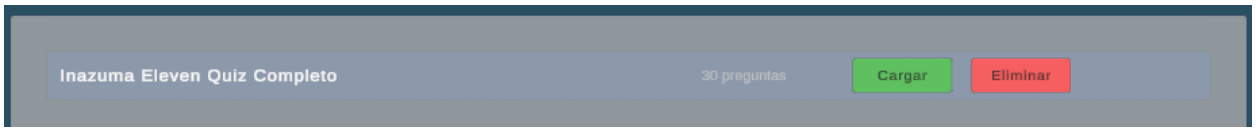
Fitxer: `Assets/Scripts/UI/KahootCreatorUI.cs`

```
private void OnDeleteKahoot(string fileName)
{
```

```
    // Confirmación (opcional: agregar un diálogo de confirmación)

    Debug.Log($"Eliminando Kahoot: {fileName}");



    KahootCreatorManager.Instance.DeleteKahoot(fileName);



    // Refrescar lista

    RefreshLoadList();



    Debug.Log("Kahoot eliminado correctamente");

}
```

Explicació del codi:

- Utilitza `File.Delete()` per eliminar el fitxer
- Refresca automàticament la llista després d'eliminar
- Sistema de gestió d'errors



7. PRINCIPIS SOLID APLICATS

7.1 Single Responsibility Principle (SRP)

Cada classe té una única responsabilitat:

- **GameManager:** Gestiona només la lògica del joc (puntuació, respuestas)
- **QuestionManager:** Gestiona només les preguntes (càrrega, navegació)
- **ErrorReportManager:** Gestiona només els informes d'error
- **DataPathManager:** Gestiona només les rutes de dades
- **XMLLeaderboardManager:** Gestiona només el leaderboard XML

Exemple de violació evitada:

MALAMENT (una classe fa massa coses):

```
public class GameController
{
    // Gestiona el joc
    public void StartGame() { }

    // Carrega preguntes
    public void LoadQuestions() { }

    // Guarda puntuacions
    public void SaveScore() { }

    // Crea reports d'error
    public void ReportError() { }

    // Gestiona UI
    public void UpdateUI() { }
}
```

CORRECTE (responsabilitats separades):

```
public class GameManager
```

```
{
    public void StartGame() { }
    public void CalculateScore() { }
}
public class QuestionManager
{
    public void LoadQuestions() { }
}
public class LeaderboardManager
{
    public void SaveScore() { }
}
public class ErrorReportManager
{
    public void ReportError() { }
}
public class QuizUIManager
{
    public void UpdateUI() { }
}
```

7.2 Open/Closed Principle (OCP)

Les classes estan obertes a extensió però tancades a modificació.

Exemple: Sistema de càrrega de Kahoots

```
// Classe base (tancada a modificació)
public class QuestionManager : MonoBehaviour
{
    public void LoadQuestions()
    {
        // Si hi ha un Kahoot personalitzat seleccionat, carregar-lo
        if (!string.IsNullOrEmpty(selectedCustomKahoot))
        {
            LoadCustomKahoot(selectedCustomKahoot);
            return;
        }

        // Carregar el Kahoot per defecte des de Resources
        LoadDefaultKahoot();
    }

    // Mètode extensible
    protected virtual void LoadDefaultKahoot()
```

```
    {
        // Implementació per defecte
    }

    // Mètode extensible
    protected virtual void LoadCustomKahoot(string fileName)
    {
        // Implementació personalitzada
    }
}
```

Explicació:

- La classe base no necessita modificar-se per afegir nous tipus de Kahoots
- Es poden afegir nous mètodes de càrrega sense tocar el codi existent
- Utilitza `virtual` per permetre sobrescriptura en classes derivades

7.3 Liskov Substitution Principle (LSP)

Els objectes derivats han de poder substituir els seus objectes base.

Exemple: Diferents tipus de questions

```
// Classe base
public class Question
{
    public string questionText;
    public string[] answers;
```

```
    public int correctAnswerIndex;
    public float timeLimit;

    // Mètode que totes les preguntes han de tenir
    public virtual bool IsCorrectAnswer(int answerIndex)
    {
        return answerIndex == correctAnswerIndex;
    }
}
// Es pot estendre sense trencar el comportament
public class TrueFalseQuestion : Question
{
    public TrueFalseQuestion()
    {
        answers = new string[] { "Veritat", "Fals" };
    }

    // Sobrescriu però manté el contracte
    public override bool IsCorrectAnswer(int answerIndex)
    {
        return answerIndex == correctAnswerIndex;
    }
```

```
}

public class MultipleChoiceQuestion : Question

{

    public int numberOfOptions = 4;



    // Sobrescriu però manté el contracte

    public override bool IsCorrectAnswer(int answerIndex)

    {

        return answerIndex == correctAnswerIndex;

    }

}
```

Explicació:

- Qualsevol tipus de pregunta es pot utilitzar on s'espera una `Question`
- No es trenca el comportament esperat
- Totes implementen correctament `IsCorrectAnswer()`

7.4 Interface Segregation Principle (ISP)

Els clients no han de dependre d'interfícies que no utilitzen.

Exemple: Interfícies específiques per a managers

```
// Interfície petita i específica

public interface IDataSaver

{

    void SaveData(string path, string data);
```

```
}
// Interfície petita i específica
public interface IDataLoader
{
    string LoadData(string path);
}
// Interfície petita i específica
public interface IErrorReporter
{
    void ReportError(string message, Exception ex);
}
// Les classes implementen només el que necessiten
public class KahootCreatorManager : MonoBehaviour, IDataSaver, IDataLoader, IErrorReporter
{
    public void SaveData(string path, string data)
    {
        File.WriteAllText(path, data);
    }

    public string LoadData(string path)
    {
```

```
        return File.ReadAllText(path);
    }

    public void ReportError(string message, Exception ex)
    {
        ErrorReportManager.Instance.ReportJSONError(message, ex.Message, ex);
    }
}
// Una altra classe pot implementar només algunes
public class SimpleDataLoader : MonoBehaviour, IDataLoader
{
    public string LoadData(string path)
    {
        return File.ReadAllText(path);
    }
    // No necessita SaveData ni ReportError
}
```

Explicació:

- Interfícies petites i específiques
- Cada classe implementa només el que necessita
- No hi ha mètodes "buits" o no utilitzats

7.5 Dependency Inversion Principle (DIP)

Les classes han de dependre d'abstraccions, no d'implementacions concretes.

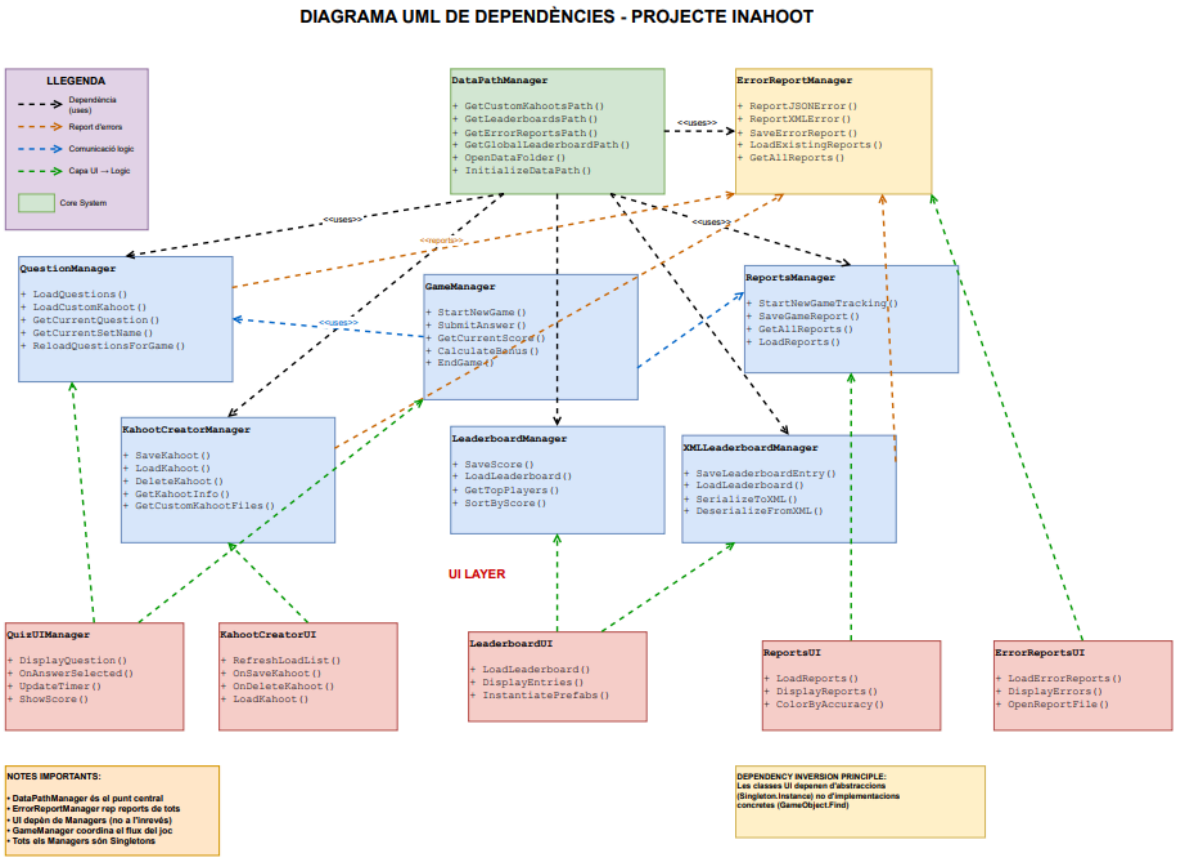
Exemple: Ús de Singleton Instances

```
//  MALAMENT (dependència directa)
public class GameUI : MonoBehaviour
{
    private QuestionManager questionManager;
    private GameManager gameManager;

    void Start()
    {
        // Buscar en l'escena (acoblament fort)
        questionManager = GameObject.Find("QuestionManager")
            .GetComponent<QuestionManager>();
        gameManager = GameObject.Find("GameManager")
            .GetComponent<GameManager>();
    }
}
//  CORRECTE (dependència d'abstracció)
public class GameUI : MonoBehaviour
{
```

```
    void Start()
    {
        // Usar Singleton (abstracció)
        if (QuestionManager.Instance != null)
        {
            Question question = QuestionManager.Instance.GetCurrentQuestion();
            // ...
        }

        if (GameManager.Instance != null)
        {
            int score = GameManager.Instance.GetCurrentScore();
            // ...
        }
    }
}
```

Explicació:

- No depèn de trobar objectes específics en l'escena
- Utilitza Singleton Instances com a punt d'accés abstracte
- Permet canviar la implementació sense afectar el codi dependent



CONCLUSIÓ

Aquest projecte demostra una implementació completa dels conceptes avançats de programació en Unity, incloent gestió robusta d'errors, arquitectura SOLID, persistència de dades, i una interfície d'usuari dinàmica i responsive. El sistema de carpetes visibles permet als usuaris finals modificar i personalitzar el contingut fàcilment, mentre que el sistema de reports proporciona transparència total sobre els errors del sistema.

Fi