Tarea: Unity - Firebase

Utilización de Base de Datos Real Time con Unity

Implementa estas tres características:

1. En el inicio del juego, recoger las posiciones de varios elementos (los prefab) y posicionarlos en función de los datos de la base de datos

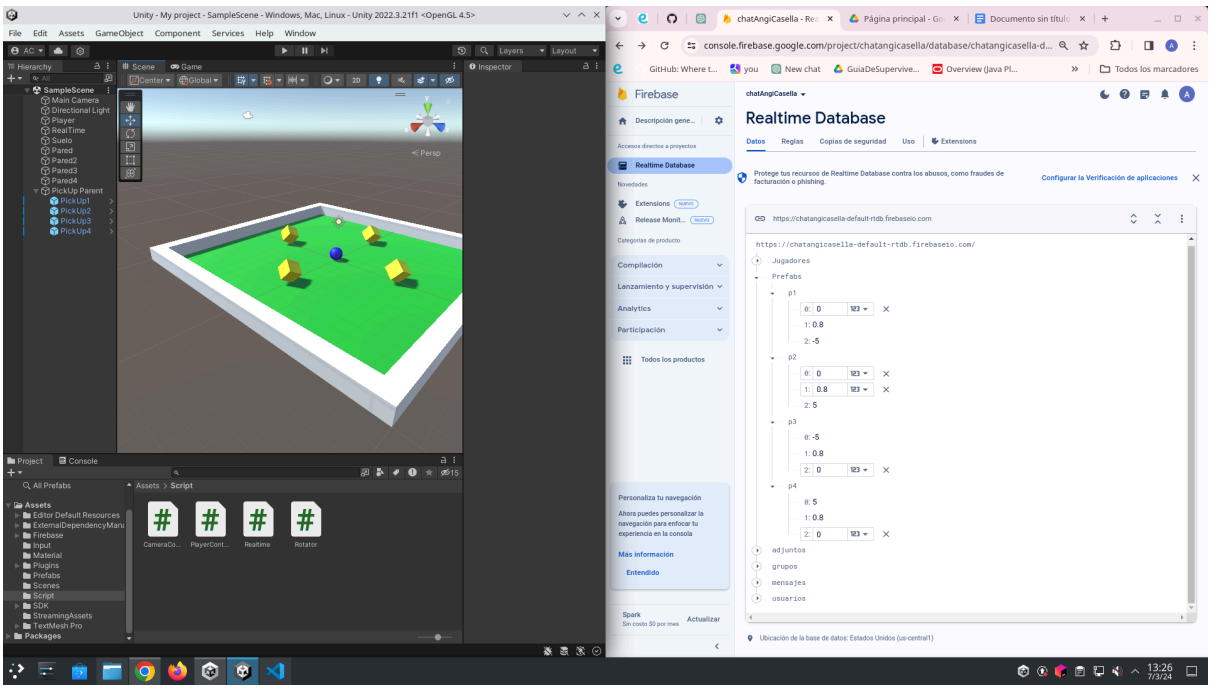
    1. Configuración de Firebase:
        - Crear un proyecto en Firebase Console.
        - Agregar la aplicación de Unity a Firebase y descargar el archivo de configuración `google-services.json`.

    2. Integración de Firebase en Unity:

        - Importar el SDK de Firebase al proyecto Unity.
        - Colocar el archivo `google-services.json` en la carpeta `Assets`

    3. Estructura de Datos en la Base de Datos:

        - En la base de datos, crear una Colección con Documentos que representen los PickUp y dentro de los mismos sus Documentos para almacenar las posiciones (x, y, z).
        - Estructura en FireBase:
            - Prefabs:
                - p1:
                    - x: 0
                    - y: 0.8
                    - z: -5
                - p2:
                    - x: 0
                    - y: 0.8
                    - z: 5
                - p3:
                    - x: -5
                    - y: 0.8
                    - z: 0
                - p4:
                    - x: 5
                    - y: 0.8
                    - z: 0

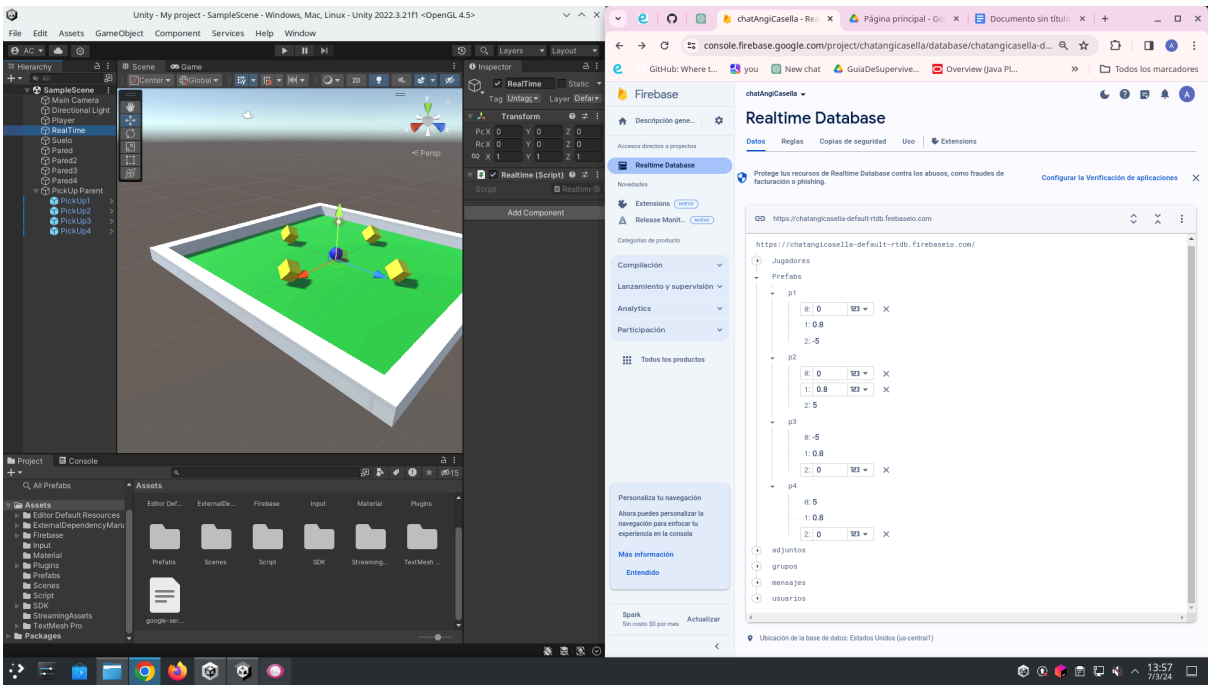


En ésta imágen se ve la estructura de la base de datos y cómo se representan los PickUp de acuerdo a los datos x, y, z establecidos en dicha base de datos.

Es importante destacar que se debe colocar el script que maneja el tiempo real del juego en un objeto vacío en la escena para que se ejecute al inicio del juego. En éste caso mi objeto vacío se llama RealTime2 y su script es el siguiente:

```
using UnityEngine;
using Firebase;
using Firebase.Database;
using Firebase.Extensions;
public class Realtime2 : MonoBehaviour
{
    DatabaseReference databaseReference;

    void Start()
{
    FirebaseApp.CheckAndFixDependenciesAsync().ContinueWithOnMainThread(task
=>
    {
        FirebaseApp app = FirebaseApp.DefaultInstance;
        databaseReference = FirebaseDatabase.DefaultInstance.RootReference;

        // Suscribe a cambios en tiempo real
        databaseReference.Child("Prefabs").ValueChanged += HandleValueChanged;

        // Llama a la función para recoger y posicionar elementos
        GetAndPositionElements();
    });
```

```
}

void HandleValueChanged(object sender, ValueChangedEventArgs args)
{
   // Maneja los cambios en la base de datos
   GetAndPositionElements();
}

   void GetAndPositionElements()
   {
       // Accede a la referencia del nodo de posiciones en la base de datos
       DatabaseReference positionsRef = databaseReference.Child("Prefabs");

       // Lee los datos de la base de datos
       positionsRef.GetValueAsync().ContinueWithOnMainThread(task =>
       {
           if (task.IsFaulted)
           {
               // Maneja el error
               Debug.LogError("Error al obtener datos de la base de datos");
           }
           else if (task.IsCompleted)
           {
               DataSnapshot snapshot = task.Result;

               // Recorre los elementos en la base de datos y posiciónalos en Unity
               foreach (DataSnapshot elementSnapshot in snapshot.Children)
               {
                   string elementName = elementSnapshot.Key;

                   // Accede a los campos "x", "y", y "z" en el documento
                   float x = float.Parse(elementSnapshot.Child("0").Value.ToString());
                   float y = float.Parse(elementSnapshot.Child("1").Value.ToString());
                   float z = float.Parse(elementSnapshot.Child("2").Value.ToString());

                   // Encuentra el objeto por el nombre y actualiza su posición
                   GameObject element = GameObject.Find("PickUp" + elementName.Substring(1));
                   if (element != null)
                   {
                       element.transform.position = new Vector3(x, y, z);
                   }
               }
           }
       });
   }
}
```



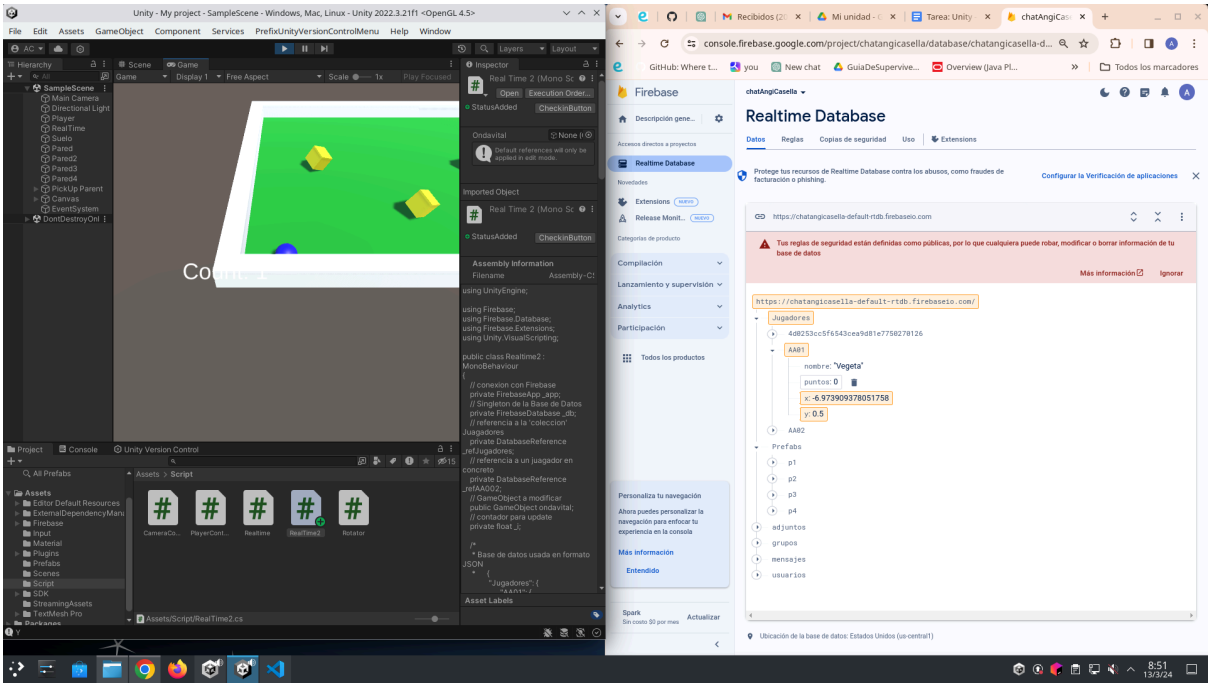
2. Actualizar datos en la base de datos según avance el juego, por ejemplo, los puntos recogidos, las vidas, etc. Esto tiene que ser particular para cada jugador. Pensar que el juego lo juegan muchos jugadores y comparten la base de datos.

   En este caso yo escojo guardar las posiciones x e y del Jugador: Para esto tengo otro script llamado RealTime que está asociado al jugador, donde en el Update se guarda en una variable la posición x y en otra variable se guarda la posición y, para referenciar luego y asignar esos valores en la base de datos.

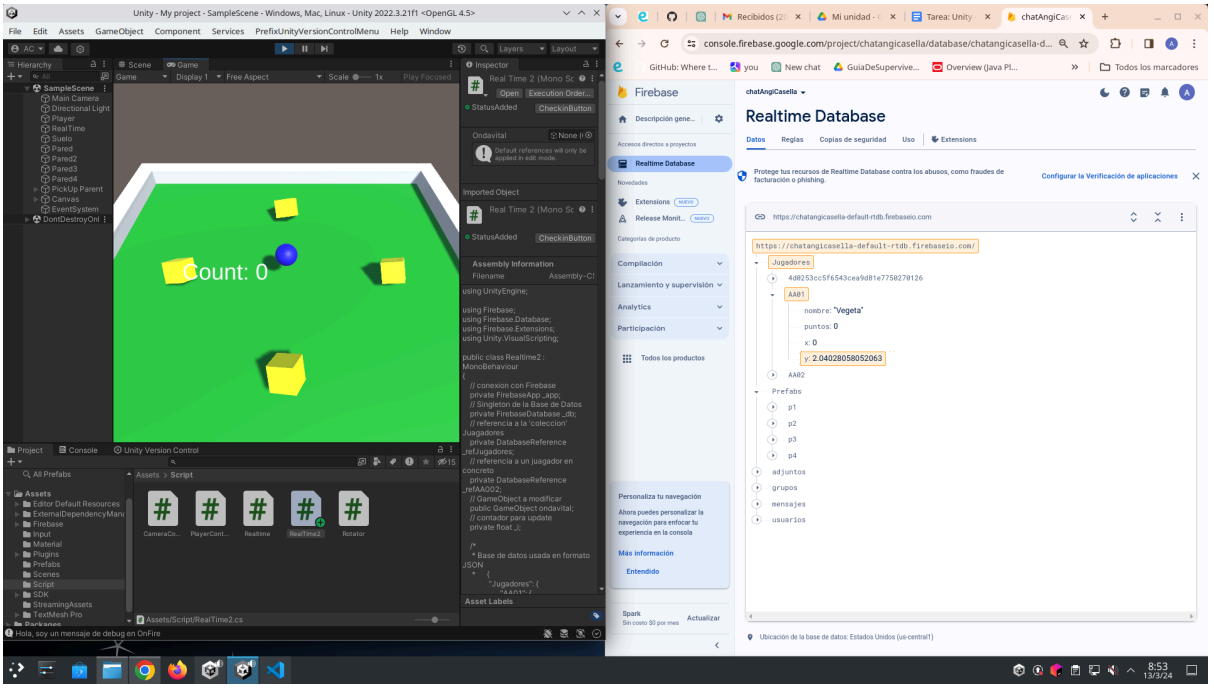
```
void Update()
    {
        double playerX = ondavital.transform.position.x;
        double playerY = ondavital.transform.position.y;

        // Actualiza la base de datos en cada frame

      _refJugadores.Child("AA01").Child("x").SetValueAsync(playerX
      );

      _refJugadores.Child("AA01").Child("y").SetValueAsync(playerY
      );
    }
```

   En firebase en el jugador AA01 agrego la x y la y para que pueda encontrar los Child.

En esta foto se ve como cambia en tiempo real la posición de x en firebase:

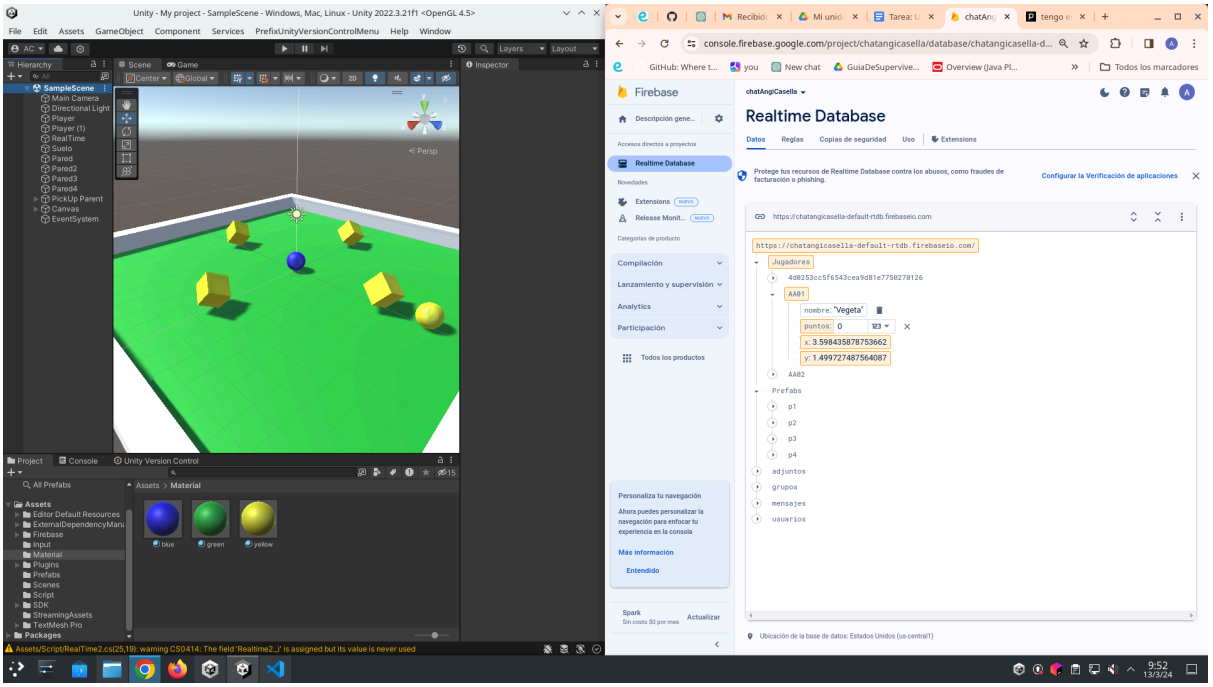


En esta foto se ve como cambia en tiempo real la posición de y en firebase:

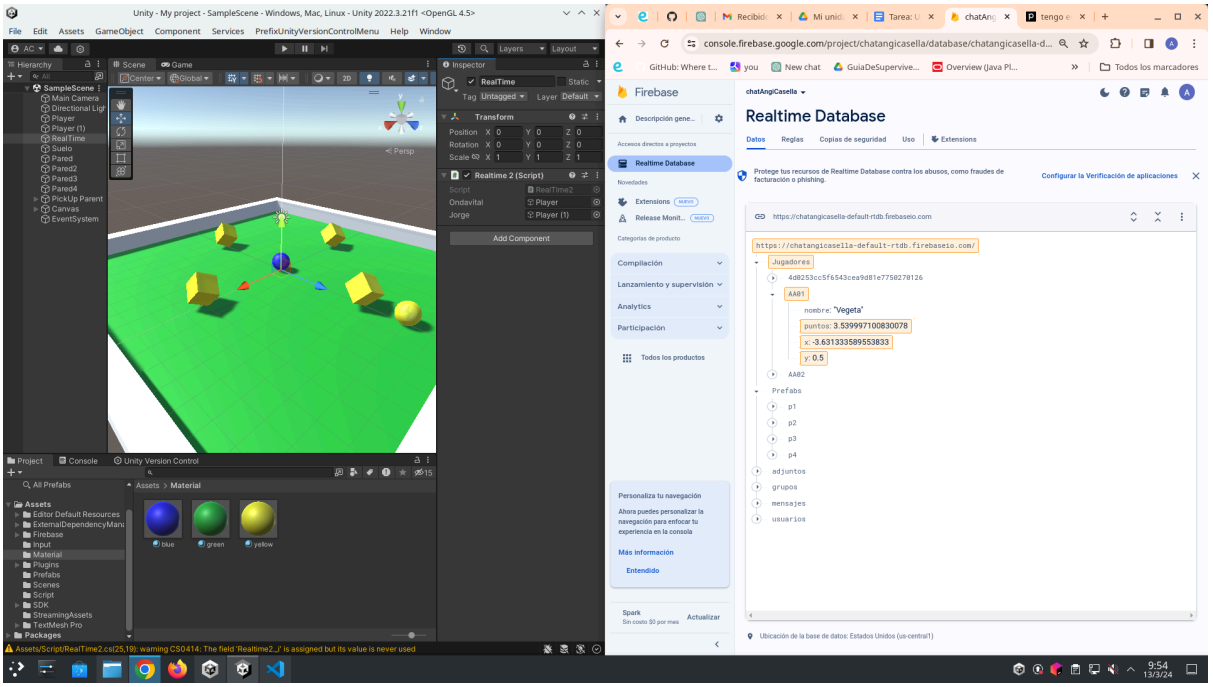


3. Actualizar en tiempo real lo que ocurre en el juego. Guardar la posición de un objeto, y usarla. Pensar en dos bolas, cada una, la mueve un jugador diferente. ¿Cómo sería moverlas tomando los valores de la base de datos?

Logré actualizar la posición x e y en tiempo real cuando mi compañero coge mi google-services.json.



En esta foto se ve como yo no tengo en play el juego y que igualmente cambia la x e y.



Y en esta foto se ve como está asociado “jorge” en el RealTime con el Player(1).