APÉNDICE A. ESTRUCTURA PARA CREAR NUEVOS PLUGINS

Para apoyar el desarrollo de nuevos plugins, a continuación se detallara paso a paso la creación de un plugin de ejemplo en NetBeans.

A.1 CREACIÓN DEL PROYECTO Y DEPENDENCIAS

Para este tutorial se asumirá el desarrollo en Netbeans 8.1. Se crea un nuevo proyecto que por simplicidad será de tipo aplicación de Java, como muestra la Figura A.1.

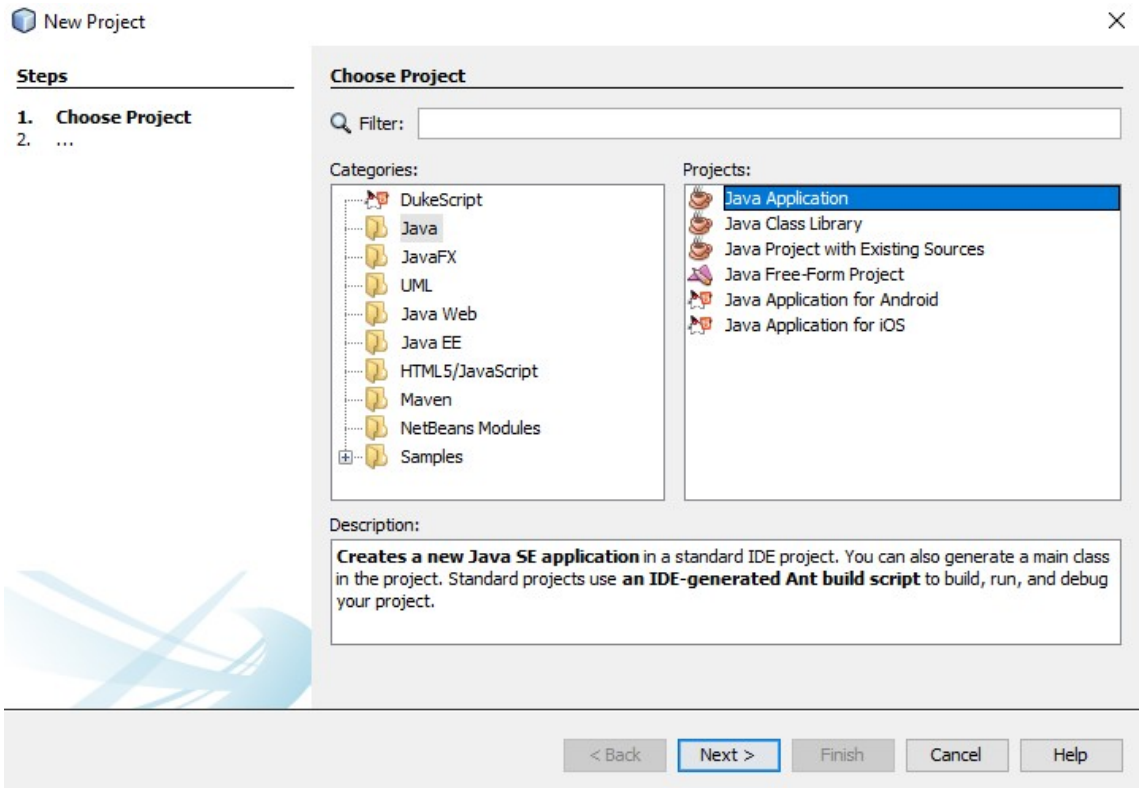

Figura A.1: Creación proyecto NetBeans.
Fuente: Elaboración propia, 2019.

Para agregar la dependencia de BG se accede al menú desplegable como se muestra en la Figura A.2 y se agrega el archiv jar.

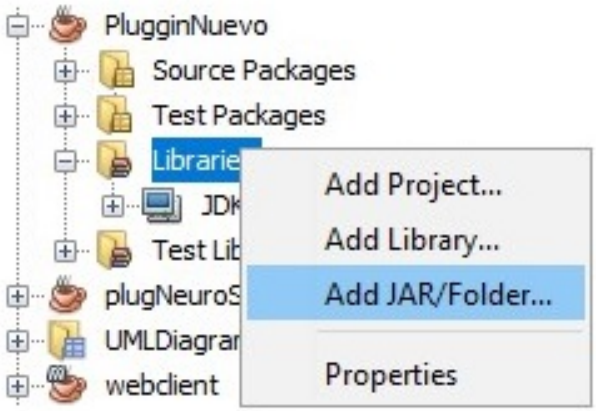

Figura A.2: Incluir dependencias.
Fuente: Elaboración propia, 2019.

A.2 IMPLEMENTACIÓN BÁSICA DE PLUGIN Y WEBSOCKET

Luego de incluir la biblioteca, es necesario implementar las interfaces indicadas anteriormente junto con sus respectivos métodos. Se utilizan las interfaces de sensorPlug y webSocketIOc, además, es necesario agregar ciertas necesidades mínimas para que la aplicación se ejecute correctamente, estás se muestran en el Código A.1.

```
1  package pluginnuevo;
2
3  //Imports necesarios
4  import BG.objects.AttributePlayer;
5  import BG.objects.SensorNeed;
6  import BG.Observador;
7  import BG.sensorPlug;
8  import BG.webSocketIOc;
9  import io.socket.client.IO;
10 import io.socket.client.Socket;
11 import io.socket.emitter.Emitter;
12
13
14 public class pluginNuevo implements sensorPlug,webSocketIOc{
15     public static void main(String[] args) {
16         // TODO Prueba previamente la obtención de datos de forma independiente
               en este lugar
17     }
18     //Cualquier método debe poder usar el WebSocket.
19     private final Socket socket;
20     //Idéntico para el SensorNeed
21     private SensorNeed SNeed;
22
23     //Puedes Agregar cualquier método según lo necesites
24     //Los métodos que no se encuentren en "initializeValues()" o en "run()" no
           serán ejecutados.
25     public String MetodoInterno(){
26     }
27
```

```
28      //Sensor Plug implements
29      @Override
30      public void initializeValues() {
31          //Se recomienda inicializar las siguientes variables
32          String Version= "1.0";
33          String Categoria = "Físico";
34          String Descripcion = "Plugin dedicado a la extracción de información
                del sensor Nuevo";
35          String NamePlug = "Nombre del plugin";
36          ArrayList<AttributePlayer> ListaAtributos = new ArrayList();
37          int PlayerID = 1;
38          String Host = "8080";
39          ArrayList<Observador> Obs = new ArrayList();
40          this.SNeed = new SensorNeed(NamePlug,Version,Categoria,Descripcion,
                ListaAtributos,PlayerID,Host,Obs);
41
42          //Para inicializar el ClienteWebSocket
43          final Socket socket = IO.socket("http://localhost:"+Host);
44          socket.on(Socket.EVENT_CONNECT, new Emitter.Listener() {//Evento de
                conectar al servidor WebSocket
45          @Override
46          public void call(Object... os) {
47              System.out.println("Se conecto al WebSocket Server"); //Acción
                    realizada cuando se conecta al servidor
48          }
49
50          }).on("join_sensor", new Emitter.Listener() {//Aquí se puede obtener
                quien entro al Room
51          @Override
52          public void call(Object... args) {
53              System.out.println("Entraron al room: "+args[0]);
54          }
55
56          }).on("AllSensors", new Emitter.Listener() { //Para obtener la lista de
                 sensores que poseen su propia room en el WebSocket Server
57          @Override
58          public void call(Object... args) {
59              try {
60                  System.out.println("Argumentos de entrada: "+args[0]);
61                  JSONObject obj = (JSONObject)args[0];
62                  System.out.println("Sensores activos: "+obj.getString("
                        sensoresActivos"));
63              } catch (JSONException ex) {
64                  Logger.getLogger(MainApp.class.getName()).log(Level.SEVERE,
                        null, ex);
65              }
66          }
67
68          }).on("message", new Emitter.Listener() {//Si es un sensor que se
                conecta a otros sensores, es neceario utilizar este evento
69
70          @Override
71          public void call(Object... args) {
72              try {
73                  JSONObject obj = (JSONObject)args[0];
74                  System.out.println("Objeto nombre: "+obj.getString("name"));
75                  System.out.println("Objeto message: "+obj.getString("message"))
                        ;
76              } catch (JSONException ex) {
```

```
77                  System.out.println("Error de obtener el objeto o nombre");
78                  Logger.getLogger(MainApp.class.getName()).log(Level.SEVERE,
                        null, ex);
79              }
80          }
81
82          }).on("Smessage", new Emitter.Listener() {//Si es un sensor que se
                conecta a otros sensores, es neceario utilizar este evento
83
84          @Override
85          public void call(Object... args) {
86              try {
87                  JSONObject obj = (JSONObject)args[0];
88                  System.out.println("Objeto nombre: "+obj.getString("name"));
89                  System.out.println("Objeto message: "+obj.getString("message"))
                        ;
90              } catch (JSONException ex) {
91                  System.out.println("Error de obtener el objeto o nombre");
92                  Logger.getLogger(MainApp.class.getName()).log(Level.SEVERE,
                        null, ex);
93              }
94          }
95
96          }).on(Socket.EVENT_DISCONNECT, new Emitter.Listener() { // evento de
                desconexión del servidor WebSocket
97
98          @Override
99          public void call(Object... args) {}
100             System.out.println("Se desconecto del WebSocket Server");//
101         });
102
103         //Por último nos conectamos al servidor
104         socket.connect();
105
106         ...
107         MetodoInterno();
108         ...
109     }
110
111     /**
112      * Se debe poder entregar el Objeto Sensorneed, que se encuentra en la
             biblioteca "objects". No olvidar inicializar este objeto en "
             initializeValues()";
113      * El código de retorno debe ser el siguiente: return this.SNeed;
114      */
115     @Override
116     public SensorNeed getSensorNeed() {
117         return this.SNeed;
118     }
119
120     /**
121      * @param Obs Son los Observadores
122      * A continuación se establecen los observadores que puede tener esta
             plugin, inicialmente el código mínimo debe ser el siguiente: this.
             SNeed.Observadores = (ArrayList)Obs;
123      */
124     @Override
125     public void setSensorNeedObservers(ArrayList<?> Obs) {
126         this.SNeed.Observadores = (ArrayList)Obs;
```

```
127     }
128
129     @Override
130     public void run() {
131         /**
132         * Para poder enviar mensajes luego por el socket el código es socket.
                emit("message",obj)
133         *obj debe poseer el nombre del socket en el que se comunica, el
                atributo y el nombre del sensor, todo lo anterior en un JSON como a
                 continuación: {room,message,name}
134         * message debe ser idéntico al Atributo de la biblioteca "objects"
135         *
136         */
137         ...
138         socket.emit("message",obj); //Ejemplo básico de envío de datos directo
                al WebScoket
139         ...
140         /**
141         * Adicionalmente si se desea enviar algún dato procesado para ser
                almacenado es necesario utiliar el siguiente metodo para que el
                sensor envie la información al modulo Sensors SNeed.notifica(
                attributePlayer)
142         *Donde attributePlayer es el objeto encontrado en objects.
                AttributePlayer
143         */
144         ...
145         SNeed.notifica(attributePlayer); //método para almacenar dato procesado
146         ...
147     }
148
149     //WebSocket implements
150     @Override
151     public void iniciarConexion(String Host) {
152         socket = IO.socket("http://localhost:"+Host);
153         socket.on(Socket.EVENT_CONNECT, new Emitter.Listener() {
154             @Override
155             public void call(Object... os) {
156                 //Evento ejecutado al lograr conectarse al websocket
157                 ...
158             }
159
160           }).on("Imessage", new Emitter.Listener() {
161             @Override
162             public void call(Object... args) {
163                 //Evento ejecutado al momento en que otro cliente se conecte a
                        la misma room
164                 ...
165             }
166
167           }).on("AllSensors", new Emitter.Listener() {
168             @Override
169             public void call(Object... args) {
170                 //Evento ejecutado cuando se consulta por todos los sensores
                        que se encuentran en el servidor
171                 ...
172             }
173
174           }).on("Smessage", new Emitter.Listener() {
175             @Override
```

```
176             public void call(Object... args) {
177                 //Evento ejecutado al momento en que llega un mensaje con un
                        JSON que contiene un atributo
178                 // solo se utiliza si este sensor se retroalimenta luego de
                        enviar información, esto solo sucedera si desde algún otro
                        cliente se envia un mensaje
179                 ...
180             }
181
182         }).on(Socket.EVENT_DISCONNECT, new Emitter.Listener() {
183             @Override
184             public void call(Object... args) {}
185                 //Evento ejecutado cuando se desconecta el cliente del servidor
186                 ...
187         });
188       ...
189     }
190 }
```

Índice de Códigos A.1: Ejemplo de plugin nuevo.
Fuente: Elaboración propia, 2019.

A.3 COMPILAR JAR

Finalmente en Netbeans se da clic al símbolo del martillo (o clic derecho en el proyecto y ”Clean and Build”) para hacer build del proyecto, esto generará un archivo .jar que se deposita en la carpeta plugins dentro de BG.

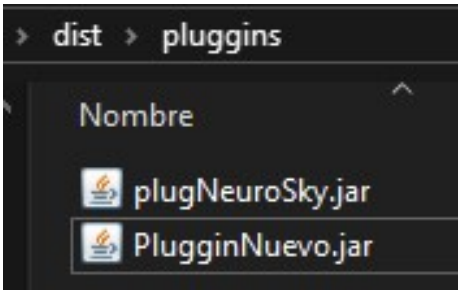

Figura A.3: Agregar jar.
Fuente: Elaboración propia, 2019.

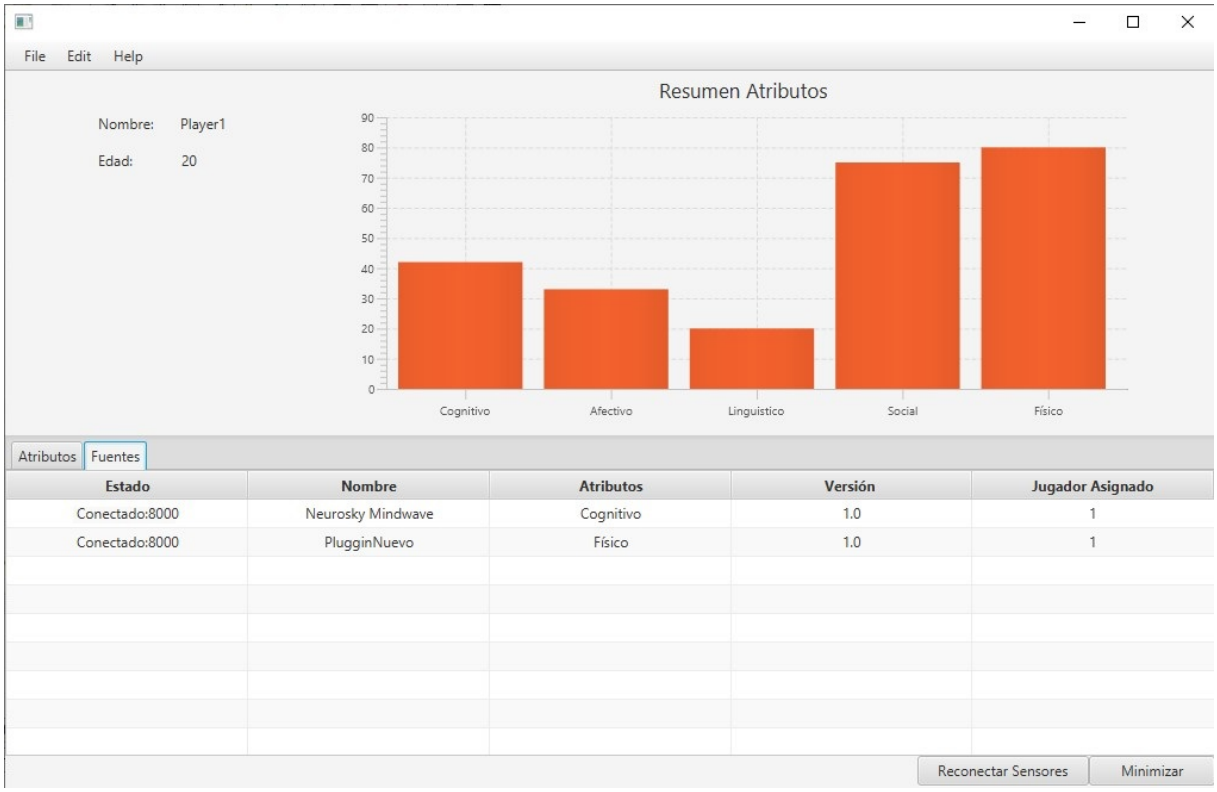

Figura A.4: Agregar jar - Interafaz de usuario.
Fuente: Elaboración propia, 2019.

APÉNDICE B. TUTORIAL PARA DESARROLLO DE VIDEOJUEGOS CON B-GAMES EN UNITY

Al diseñar nuevos videojuegos se puede utilizar esta herramienta para administrar nuevos datos de sensores de forma sencilla basándose en la comunicación por WebSocket. A continuación, se describen los pasos para poder utilizar el Asset de Unity.

Los requerimientos iniciales antes de utilizar el asset son:

- Instalar Aplicación de escritorio.
- Descargar plugin que se desea utilizar y ubicar en carpeta plugins.
- Para trabajar con fuentes de datos externas es necesario desplegar los microservicios.

B.1 AGREGANDO ASSET: SCRIPT Y PREFAB

Para comenzar a usar b-Games Framework en Unity inicialmente hay que descargar el Asset desde el repositorio de git: `https://github.com/Grybyus/BGAssetUnity`. Luego de descargar hay que ubicar la carpeta BG en la carpeta Assets dentro del proyecto de Unity en el que se esta trabajando:

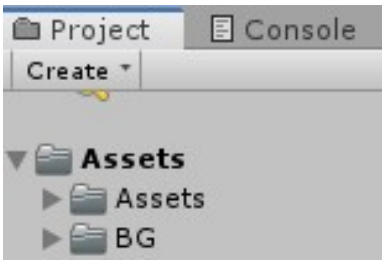

Figura B.1: Agregar Asset BG.
Fuente: Elaboración propia, 2019.

Ahora con los archivos dispuestos dentro del proyecto, se accede a la carpeta BG/Data donde se encuentra el script BGWebSocket, el cual debe ser arrastrado hacia un Scene para que se ejecute, al mismo tiempo hay que arrastrar el Prefab que se encuentra en la carpeta BG/SocketIO/Prefabs hasta los GameObjects del proyecto.

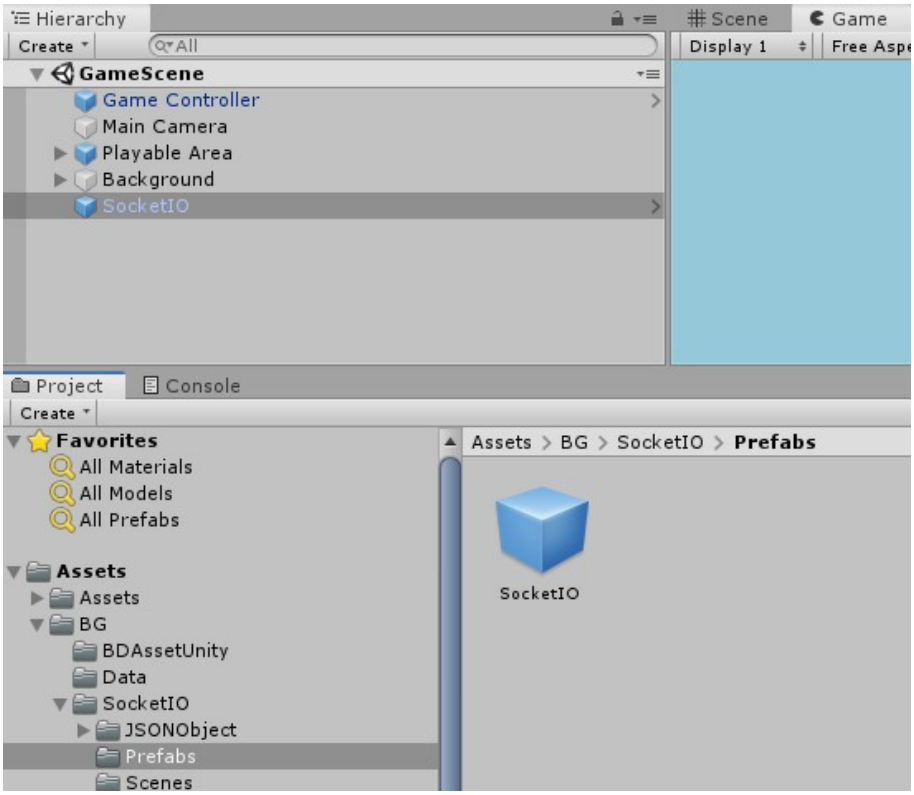

Figura B.2: Agregar Prefab.
Fuente: Elaboración propia, 2019.

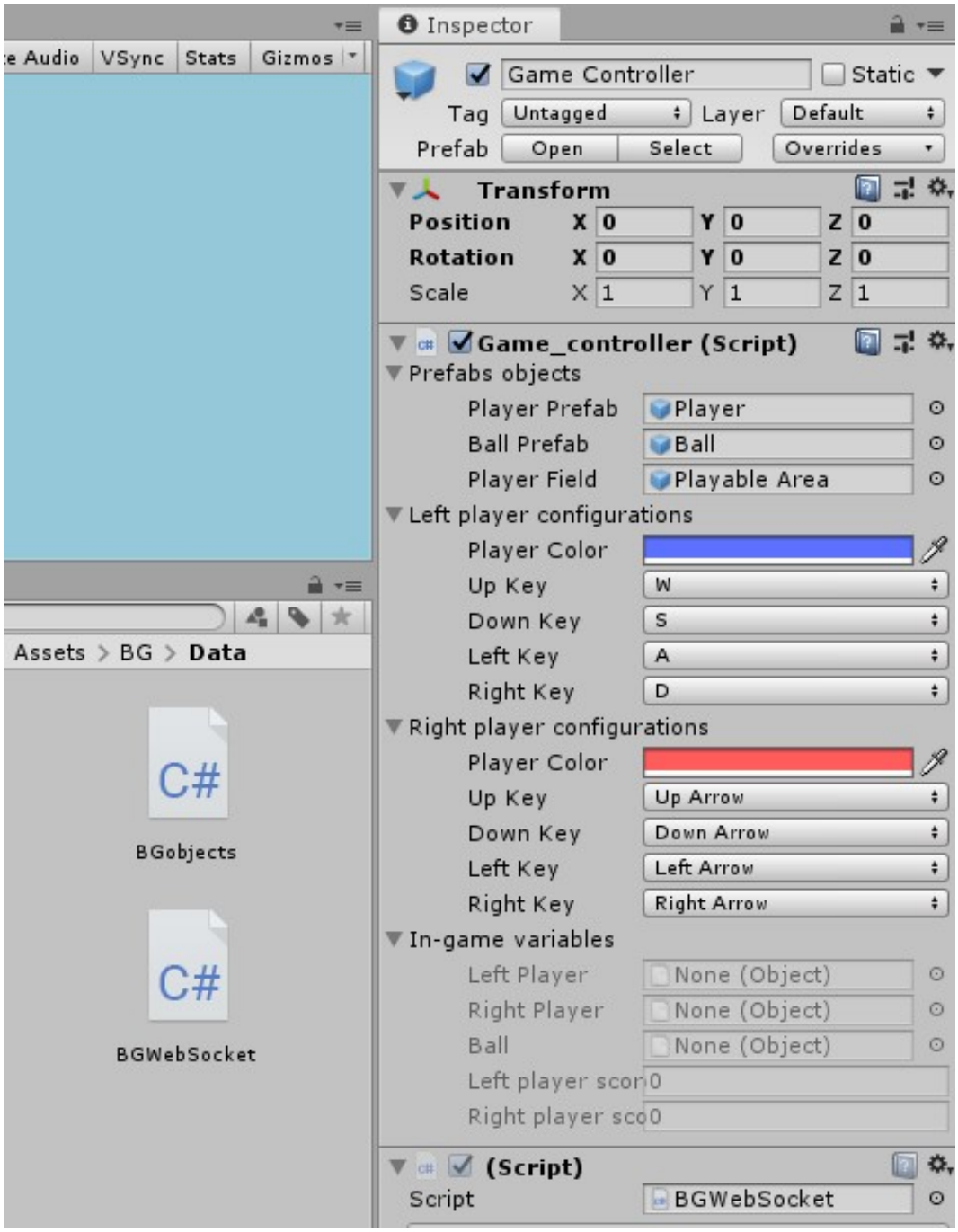

Figura B.3: Agregar Script.
Fuente: Elaboración propia, 2019.

Por último hay que modificar el Url del servidor donde tenemos nuestro WebSocket en el Prefab SocketIO agregado anteriormente.

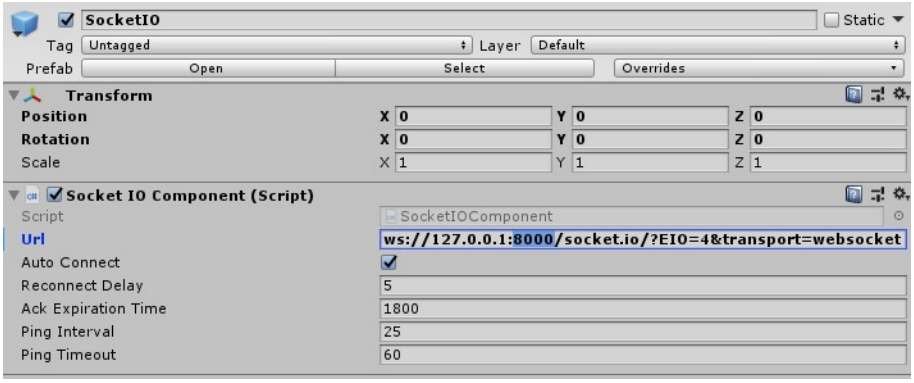

Figura B.4: Cambiar Url.
Fuente: Elaboración propia, 2019.

Ahora esta todo listo para utilizar los métodos necesarios y los objetos de atributos para obtener y manejar datos de diversas fuentes de información.

B.2 IMPLEMENTACIÓN BÁSICA DE USO

Habiendo completado los pasos anteriores, nos dirigirnos al Script que deseemos para modificar los eventos del WebSocket ante la entrada de Atributos, tanto como de Resumen de atributos (Categorías), como de atributos simples (Procesados o en Streaming).

```
1  //Agregar biblioteca SocketIO
2  using SocketIO;
3  //Declarar una variable para el uso del atributo y del atributo categoría o
       resumen
4  private BGobjects.AttributePlayer attAux;
5  private BGobjects.AttributeResPlayer attAuxRes;
6  //Agregar al inicio del script
7  void Start()
8  {
9      //Para obtener los sensores que están disponibles en el socket se utiliza :
10     BDWebSocket.instance.GetAllSensors()
11     //Revisando la lista de Strings en APIRestClient.instance.AllSensor o
           viendo el mensaje impreso en pantalla, se obtienen los sensores
           disponibles.
12     //Ahora nos conectamos a alguno de los sensores disponibles, de la
           siguiente manera:
13     BDWebSocket.instance.ConnectToSensor("NombreObtenidoEnAllsensors", "
           NombreDelJuegoNuevo");
14     // Se específica las funciones OnSmessage y OnRmessage para manejar el
           evento de la llegada de un atributo, para facilitar el uso de los datos
            apenas estén disponibles.
15     BDWebSocket.instance.socket.On("Smessage",OnSmessage);
16     BDWebSocket.instance.socket.On("Rmessage",OnRmessage);
17
18     // Para desconectarse de un sensor o fuente de información en específico es
            idéntico al ConnectToSensor
19     BDWebSocket.instance.DisconnectToSensor("NombreObtenidoEnAllsensors", "
           NombreDelJuegoNuevo");
```

```
20  }
21  //El siguiente código solo se ejecutara cuando el WebSocket se comunique
        enviándonos los atributos el resumen de una categoría de atributo.
22  private void OnSmessage(SocketIOEvent socketIOevent)
23  {
24  //Guardamos el atributo que nos llega en el objeto que posee los campos
        correspondientes al Atributo esperado, y luego podemos usar el dato.
25      attAux = BDWebSocket.instance.JSONstrToAttribute(socketIOevent.data);
26      string data = attAux.Dato.ToString();
27  }
28
29  private void OnRmessage(SocketIOEvent socketIOevent)
30  {
31  //De la misma manera para el resumen de una categoría de atributo se transforma
        en su objeto correspondiente
32      attAuxRes = BDWebSocket.instance.JSONstrToAttribute(socketIOevent.data);
33      string data = attAuxRes.Dato.ToString();
34  }
```

Índice de Códigos B.1: Ejemplo de Uso de b-Games Framework.
Fuente: Elaboración propia, 2019.