```
LinksPlatform's Platform Communication Class Library
     ./Platform.Communication/Protocol/Gexf/Edge.cs
   using System.Globalization;
   using System.Xml;
2
   using System.Xml.Serialization;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Communication.Protocol.Gexf
        public class Edge
9
10
            public static readonly string ElementName = "edge";
11
            public const string IdAttributeName = "id";
public const string SourceAttributeName = "source";
public const string TargetAttributeName = "target";
12
13
14
            public const string LabelAttributeName = "label";
16
            [XmlAttribute(AttributeName = IdAttributeName)]
17
            public long Id { get; set; }
18
            [XmlAttribute(AttributeName = SourceAttributeName)]
20
            public long Source { get; set; }
21
22
            [XmlAttribute(AttributeName = TargetAttributeName)]
23
            public long Target { get; set; }
25
            [XmlAttribute(AttributeName = LabelAttributeName)]
26
27
            public string Label { get; set; }
28
            public void WriteXml(XmlWriter writer) => WriteXml(writer, Id, Source, Target, Label);
30
            public static void WriteXml(XmlWriter writer, long id, long sourceNodeId, long
31
            targetNodeId) => WriteXml(writer, id, sourceNodeId, targetNodeId, null);
32
            public static void WriteXml(XmlWriter writer, long id, long sourceNodeId, long
33
                targetNodeId, string label)
                // <edge id="0" source="0" target="0" label="..." />
35
                writer.WriteStartElement(ElementName);
36
                writer.WriteAttributeString(IdAttributeName,

    id.ToString(CultureInfo.InvariantCulture));
                writer.WriteAttributeString(SourceAttributeName,
                 writer.WriteAttributeString(TargetAttributeName,
39
                    targetNodeId.ToString(CultureInfo.InvariantCulture));
                if (!string.IsNullOrWhiteSpace(label))
40
                {
41
                     writer.WriteAttributeString(LabelAttributeName, label);
43
                writer.WriteEndElement();
44
            }
        }
46
47
     ./Platform.Communication/Protocol/Gexf/Gexf.cs
   using System;
   using System.Xml;
2
   using System.Xml.Serialization;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Communication.Protocol.Gexf
7
8
        [XmlRoot(ElementName = ElementName, Namespace = Namespace)]
        public class Gexf
10
11
            public const string ElementName = "gexf";
12
            public const string Namespace = "http://www.gexf.net/1.2draft";
13
            public const string VersionAttributeName = "version";
public const string GraphElementName = "graph";
15
            public static readonly string CurrentVersion = "1.2";
16
17
            [XmlAttribute(AttributeName = VersionAttributeName)]
18
            public string Version { get; set; }
19
20
            [XmlElement(ElementName = GraphElementName)]
21
            public Graph Graph { get; set; }
23
            public Gexf() => (Version, Graph) = (CurrentVersion, new Graph());
```

```
public void WriteXml(XmlWriter writer) => WriteXml(writer, () => Graph.WriteXml(writer),
            → Version);
            public static void WriteXml(XmlWriter writer, Action writeGraph) => WriteXml(writer,
28

→ writeGraph, CurrentVersion);

29
            public static void WriteXml(XmlWriter writer, Action writeGraph, string version)
30
                writer.WriteStartDocument();
32
                writer.WriteStartElement(ElementName, Namespace);
33
                writer.WriteAttributeString(VersionAttributeName, version);
                writeGraph();
                writer.WriteEndElement()
36
                writer.WriteEndDocument();
37
            }
38
39
            public static void WriteXml(XmlWriter writer, Action writeNodes, Action writeEdges) =>
            WriteXml(writer, writeNodes, writeEdges, CurrentVersion, GraphMode.Static,
               GraphDefaultEdgeType.Directed);
41
            public static void WriteXml(XmlWriter writer, Action writeNodes, Action writeEdges,
               string version) => WriteXml(writer, writeNodes, writeEdges, version,
               GraphMode.Static, GraphDefaultEdgeType.Directed);
43
            public static void WriteXml(XmlWriter writer, Action writeNodes, Action writeEdges,
44
            string version, GraphMode mode) => WriteXml(writer, writeNodes, writeEdges, version,
            → mode, GraphDefaultEdgeType.Directed);
45
            public static void WriteXml(XmlWriter writer, Action writeNodes, Action writeEdges,
46
                string version, GraphMode mode, GraphDefaultEdgeType defaultEdgeType) =>
                WriteXml(writer, () => Graph.WriteXml(writer, writeNodes, writeEdges, mode,
               defaultEdgeType), version);
       }
   }
48
    ./Platform.Communication/Protocol/Gexf/Graph.cs
   using System;
   using System.Collections.Generic;
   using System.Xml;
3
   using System.Xml.Serialization;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Communication.Protocol.Gexf
Q
        public class Graph
10
11
            public static readonly string ElementName = "graph";
public const string ModeAttributeName = "mode";
12
13
            public const string DefaultEdgeTypeAttributeName = "defaultedgetype";
            public const string NodesElementName =
                                                      "nodes";
15
            public const string NodeElementName = "node";
public const string EdgesElementName = "edges";
16
17
            public const string EdgeElementName = "edge"
18
            [XmlAttribute(AttributeName = ModeAttributeName)]
20
            public GraphMode Mode { get; set; }
22
            [XmlAttribute(AttributeName = DefaultEdgeTypeAttributeName)]
23
            public GraphDefaultEdgeType DefaultEdgeType { get; set; }
25
            [XmlArray(ElementName = NodesElementName)]
            [XmlArrayItem(ElementName = NodeElementName)]
28
            public List<Node> Nodes { get; set; }
            [XmlArray(ElementName = EdgesElementName)]
30
            [XmlArrayItem(ElementName = EdgeElementName)]
31
            public List<Edge> Edges { get; set; }
33
            public Graph() => (Nodes, Edges) = (new List<Node>(), new List<Edge>());
34
35
            public void WriteXml(XmlWriter writer) => WriteXml(writer, () => WriteNodes(writer), ()
36
            → => WriteEdges(writer), Mode, DefaultEdgeType);
37
            private void WriteEdges(XmlWriter writer)
38
39
                for (var i = 0; i < Edges.Count; i++)</pre>
```

```
Edges[i] .WriteXml(writer);
42
                }
43
            }
44
            private void WriteNodes(XmlWriter writer)
46
47
                for (var i = 0; i < Nodes.Count; i++)</pre>
48
                    Nodes[i] .WriteXml(writer);
50
                }
51
            }
53
            public static void WriteXml(XmlWriter writer, Action writeNodes, Action writeEdges) =>
54
                WriteXml(writer, writeNodes, writeEdges, GraphMode.Static,
                GraphDefaultEdgeType.Directed);
            public static void WriteXml(XmlWriter writer, Action writeNodes, Action writeEdges,
56
                GraphMode mode) => WriteXml(writer, writeNodes, writeEdges, mode,
                GraphDefaultEdgeType.Directed);
57
            public static void WriteXml(XmlWriter writer, Action writeNodes, Action writeEdges,
58
                GraphMode mode, GraphDefaultEdgeType defaultEdgeType)
                writer.WriteStartElement(ElementName);
                writer.WriteAttributeString(ModeAttributeName, mode.ToString().ToLower());
                writer.WriteAttributeString(DefaultEdgeTypeAttributeName,
62

→ defaultEdgeType.ToString().ToLower());
                writer.WriteStartElement(NodesElementName);
                writeNodes():
64
                writer.WriteEndElement():
65
                writer.WriteStartElement(EdgesElementName);
66
                writeEdges()
                writer.WriteEndElement();
68
                writer.WriteEndElement();
69
            }
       }
71
72
     ./Platform.Communication/Protocol/Gexf/GraphDefaultEdgeType.cs
1.4
   using System.Xml.Serialization;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Communication.Protocol.Gexf
5
        public enum GraphDefaultEdgeType
            [XmlEnum(Name = "directed")]
9
            Directed
10
        }
11
12
    /Platform.Communication/Protocol/Gexf/GraphMode.cs
   using System.Xml.Serialization;
2
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
3
4
   namespace Platform.Communication.Protocol.Gexf
6
        public enum GraphMode
            [XmlEnum(Name = "static")]
            Static,
11
            [XmlEnum(Name = "dynamic")]
12
            Dynamic
13
        }
14
   ./Platform.Communication/Protocol/Gexf/Node.cs
   using System.Globalization;
using System.Xml;
   using System.Xml.Serialization;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Communication.Protocol.Gexf
       public class Node
```

```
10
            public static readonly string ElementName = "node";
11
            public const string IdAttributeName = "id";
public const string LabelAttributeName = "label";
13
            [XmlAttribute(AttributeName = IdAttributeName)]
15
            public long Id { get; set; }
16
17
            [XmlAttribute(AttributeName = LabelAttributeName)]
18
            public string Label { get; set; }
19
20
            public void WriteXml(XmlWriter writer) => WriteXml(writer, Id, Label);
22
            public static void WriteXml(XmlWriter writer, long id, string label)
23
24
                // <node id="0" label="...
25
                writer.WriteStartElement(ElementName);
26
                writer.WriteAttributeString(IdAttributeName,

→ id.ToString(CultureInfo.InvariantCulture));
                writer.WriteAttributeString(LabelAttributeName, label);
                writer.WriteEndElement();
29
            }
30
        }
31
   }
32
     ./Platform.Communication/Protocol/Udp/UdpClientExtensions.cs
1.7
   using System.Net;
   using System.Net.Sockets;
   using System.Runtime.CompilerServices;
using System.Text;
3
4
   using Platform.Singletons;
5
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Communication.Protocol.Udp
   {
10
        public static class UdpClientExtensions
11
12
            private static readonly Encoding _defaultEncoding = Singleton.Get(() =>
13

→ Encoding.GetEncoding(0));
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static int SendString(this UdpClient udp, IPEndPoint ipEndPoint, string message)
16
17
                var bytes = _defaultEncoding.GetBytes(message);
18
                return udp.Send(bytes, bytes.Length, ipEndPoint);
19
            }
20
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
22
            public static string ReceiveString(this UdpClient udp)
23
                IPEndPoint remoteEndPoint = default;
25
                return _defaultEncoding.GetString(udp.Receive(ref remoteEndPoint));
            }
27
        }
28
29
    ./Platform.Communication/Protocol/Udp/UdpReceiver.cs
1.8
   using System;
   using System.Net.Sockets;
2
   using System.Runtime.CompilerServices;
   using System. Threading
4
   using Platform.Disposables;
   using Platform. Exceptions;
   using Platform. Threading;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
10
   namespace Platform.Communication.Protocol.Udp
11
12
        public delegate void MessageHandlerCallback(string message);
13
14
        /// <summary>
15
        /// Represents the receiver of messages transfered via UDP protocol.
16
        /// Представляет получателя сообщений по протоколу UDP.
17
        /// </summary>
18
        public class UdpReceiver : DisposableBase //-V3073
19
20
            private const int DefaultPort = 15000;
```

```
private bool _receiverRunning;
private Thread _thread;
private readonly UdpClient _udp;
            private readonly MessageHandlerCallback _messageHandler;
            public bool Available => _udp.Available > 0;
            public UdpReceiver(int listenPort, bool autoStart, MessageHandlerCallback messageHandler)
30
                _udp = new UdpClient(listenPort);
                _messageHandler = messageHandler;
                if (autoStart)
34
35
                     Start();
36
                }
            }
            public UdpReceiver(int listenPort, MessageHandlerCallback messageHandler) :
40
                this(listenPort, true, messageHandler) { }
            public UdpReceiver(MessageHandlerCallback messageHandler) : this(DefaultPort, true,
             \rightarrow messageHandler) { }
            public UdpReceiver() : this(DefaultPort, true, message => { }) { }
            public void Start()
                if (!_receiverRunning && _thread == null)
                     _receiverRunning = true;
50
                     _thread = new Thread(Receiver);
                     _thread.Start();
            }
54
            public void Stop()
56
                if (_receiverRunning && _thread != null)
                     _receiverRunning = false;
                     _thread.Join();
                     _thread = null;
                }
            }
64
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
66
            public string Receive() => _udp.ReceiveString();
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public void ReceiveAndHandle() => _messageHandler(Receive());
            // Функция извлекающая пришедшие сообщения
            // и работающая в отдельном потоке.
            private void Receiver()
                while (_receiverRunning)
                     trv
                         if (Available)
                         {
                             ReceiveAndHandle();
                         }
                         else
                         {
                             ThreadHelpers.Sleep();
                     catch (Exception exception)
90
                         exception.Ignore();
                }
            }
            protected override void Dispose(bool manual, bool wasDisposed)
                if (!wasDisposed)
```

23 24

26 27

28 29

31

32

37

38

41

42

43

44 45

46 47

48 49

51

52 53

57

58 59

60

62

67

69

70 7.1

72

73

74 7.5

76

78 79

80

81

82

84

86 87

89

91

93

94

96 97

qq

```
Stop();
100
                     _udp.DisposeIfPossible();
                 }
102
            }
103
        }
    }
105
     ./Platform.Communication/Protocol/Udp/UdpSender.cs
    using System.Net;
    using System.Net.Sockets;
using System.Runtime.CompilerServices;
 2
    using Platform.Disposables;
    #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
 6
    namespace Platform.Communication.Protocol.Udp
    {
 9
        /// <summary>
1.0
        /// Represents the sender of messages transfered via UDP protocol.
11
        /// Представляет отправителя сообщений по протоколу UDP.
12
        /// </summary>
13
        public class UdpSender : DisposableBase //-V3073
14
            private readonly UdpClient _udp;
private readonly IPEndPoint _ipendpoint;
16
17
18
            public UdpSender(IPEndPoint ipendpoint) => (_udp, _ipendpoint) = (new UdpClient(),

→ ipendpoint);

20
            public UdpSender(IPAddress address, int port) : this(new IPEndPoint(address, port)) { }
21
22
            public UdpSender(string hostname, int port) : this(IPAddress.Parse(hostname), port) { }
23
24
            public UdpSender(int port) : this(IPAddress.Loopback, port) { }
25
26
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
27
            public int Send(string message) => _udp.SendString(_ipendpoint, message);
28
29
            protected override void Dispose(bool manual, bool wasDisposed)
30
                 if (!wasDisposed)
32
                 {
33
                     _udp.DisposeIfPossible();
                 }
35
             }
36
        }
37
38
1.10
      ./Platform.Communication/Protocol/Xml/Serializer.cs
    using System.IO;
    using System. Text
 2
    using System.Xml.Serialization;
 4
    #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
    namespace Platform.Communication.Protocol.Xml
        public static class Serializer<T>
 9
10
            public static readonly XmlSerializer Instance = new XmlSerializer(typeof(T));
11
12
             public static T FromFile(string path)
13
14
                 using (var stream = File.OpenRead(path))
15
16
                     return (T) Instance.Deservalize(stream);
17
                 }
             }
19
20
             public static T FromString(string xml)
21
22
                 using (var reader = new StringReader(xml))
23
                     return (T)Instance.Deserialize(reader);
25
26
             }
27
28
            public static void ToFile(T @object, string path)
29
```

```
using (var stream = File.OpenWrite(path))
31
33
                                               Instance.Serialize(stream, @object);
34
                           }
36
                           public static string ToString(T @object)
37
38
                                     var sb = new StringBuilder();
39
                                    using (var writer = new StringWriter(sb))
40
41
                                              Instance.Serialize(writer, @object);
42
43
                                     return sb.ToString();
44
                           }
45
                 }
46
        }
47
1.11
              ./Platform.Communication.Tests/SerializerTests.cs
       using System;
        using System.IO;
       using Xunit;
using Platform.Singletons;
 3
 4
        using Platform.Communication.Protocol.Xml;
        namespace Platform.Communication.Tests
                  public static class SerializerTests
10
                           [Fact]
11
                           public static void SerializeToFileTest()
12
13
                                     var tempFilename = Path.GetTempFileName();
14
                                     Serializer<object>.ToFile(Default<object>.Instance, tempFilename);
15
                                     Assert.Equal(File.ReadAllText(tempFilename), $\"<?xml
16
                                             version=\"1.0\"?>{Environment.NewLine}<anyType
                                      xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
                                            xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" />");
                                    File.Delete(tempFilename);
                           }
18
                           [Fact]
20
                           public static void SerializeAsXmlStringTest()
21
22
                                     var serializedObject = Serializer<object>.ToString(Default<object>.Instance);
                                     Assert.Equal(serializedObject, $\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\sq}}}}}}}}}}}eltarestrup}}}eltarestendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentendentende
24
                                              encoding=\"utf-16\"?>{Environment.NewLine}<anyType
                                              xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
                                              xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" />");
                           }
25
                  }
26
        }
             ./Platform.Communication.Tests/UdpReceiverTests.cs
1.12
       using Xunit;
using Platform.Communication.Protocol.Udp;
        namespace Platform.Communication.Tests
 4
 5
                  public class UdpReceiverTests
 6
                           |Fact|
 9
                           public static void DisposalTest()
10
                                     using (var receiver = new UdpReceiver())
11
12
13
                           }
14
                 }
15
        }
```

16

## Index

```
./Platform.Communication.Tests/SerializerTests.cs, 7
./Platform.Communication.Tests/UdpReceiverTests.cs, 7
./Platform.Communication/Protocol/Gexf/Edge.cs, 1
./Platform.Communication/Protocol/Gexf/Gexf.cs, 1
./Platform.Communication/Protocol/Gexf/Graph.cs, 2
./Platform.Communication/Protocol/Gexf/GraphDefaultEdgeType.cs, 3
./Platform.Communication/Protocol/Gexf/Node.cs, 3
./Platform.Communication/Protocol/Gexf/Node.cs, 3
./Platform.Communication/Protocol/Udp/UdpClientExtensions.cs, 4
./Platform.Communication/Protocol/Udp/UdpReceiver.cs, 4
./Platform.Communication/Protocol/Udp/UdpSender.cs, 6
./Platform.Communication/Protocol/Xml/Serializer.cs, 6
```