1.1 ./csharp/Platform.Timestamps/Timestamp.cs

```
using System;
using System.Runtime.CompilerServices;

namespace Platform.Timestamps
{
    /// <summary>
    /// <para>Represents a timestamp.</para>
    /// <para>Представляет метку времени.</para>
    /// </summary>
    /// <remarks>
    /// <para>To make this timestamp truly unique, it is recommended to use <see cref="UniqueTimestampFactory"/>.</para>
    /// <para>Чтобы эта метка времени была действительно уникальна рекомендуется использовать <see cref="UniqueTimestampFactory"/>.</para>
    /// </remarks>
    public struct Timestamp : IEquatable<Timestamp>
    {
        /// <summary>
        /// <para>Returns a string containing the default DateTime format for Timestamp.</para>
        /// <para>Возвращает строку, содержащую формат даты и времени по умолчанию для метки времени.</para>
        /// </summary>
        public static readonly string DefaultFormat = "yyyy.MM.dd hh:mm:ss.fffffff";

        /// <summary>
        /// <para>Gets or sets the number of ticks that represent the date and time in UTC.</para>
        /// <para>Возвращает или устанавливает количество тиков, которые представляют дату и время в UTC.</para>
        /// </summary>
        public readonly ulong Ticks;

        /// <summary>
        /// <para>Creates a timestamp.</para>
        /// <para>Создаёт метку времени.</para>
        /// </summary>
        /// <param name="ticks"><para>A number representing the number of ticks.</para><para>Число представляющие количество тиков.</para></param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Timestamp(ulong ticks) => Ticks = ticks;

        /// <summary>
        /// <para>Defines an implicit conversion of a DateTime to a Timestamp.</para>
        /// <para>Определяет неявное преобразование DateTime в метку времени.</para>
        /// </summary>
        /// <param name="dateTime"><para>The DateTime struct.</para><para>Структура DateTime.</para></param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public static implicit operator Timestamp(DateTime dateTime) => new Timestamp((ulong)dateTime.ToUniversalTime().Ticks);

        /// <summary>
        /// <para>Defines an implicit conversion of a Timestamp to a DateTime.</para>
        /// <para>Определяет неявное преобразование метки времени в DateTime.</para>
        /// </summary>
        /// <param name="timestamp"><para>The Timestamp.</para><para>Отметка времени.</para></param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public static implicit operator DateTime(Timestamp timestamp) => new DateTime((long)timestamp.Ticks, DateTimeKind.Utc);

        /// <summary>
        /// <para>Defines an implicit conversion of a 64-bit unsigned integer to a Timestamp.</para>
        /// <para>Определяет неявное преобразование 64-разрядного целого числа без знака в метку времени.</para>
        /// </summary>
        /// <param name="ticks"><para>The number of ticks represented as a 64-bit integer.</para><para>Количество тиков представленное в виде 64-разрядного целого числа.</para></param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public static implicit operator Timestamp(ulong ticks) => new Timestamp(ticks);

        /// <summary>
```

```
61          /// <para>Defines an implicit conversion of a Timestamp to a 64-bit unsigned
            ↪  integer.</para>
62          /// <para>Определяет неявное преобразование метки времени в 64-разрядное целое число без
            ↪  знака.</para>
63          /// </summary>
64          /// <param name="timestamp"><para>The Timestamp.</para><para>Отметка
            ↪  времени.</para></param>
65          [MethodImpl(MethodImplOptions.AggressiveInlining)]
66          public static implicit operator ulong(Timestamp timestamp) => timestamp.Ticks;
67
68          /// <summary>
69          /// <para>Returns a string that represents the current Timestamp.</para>
70          /// <para>Возвращает строку, которая представляет текущую метку времени.</para>
71          /// </summary>
72          /// <returns><para>A string that represents the current Timestamp.</para><para>Строка,
            ↪  представляющая текущую метку времени.</para></returns>
73          [MethodImpl(MethodImplOptions.AggressiveInlining)]
74          public override string ToString() => ((DateTime)this).ToString(DefaultFormat);
75
76          /// <summary>
77          /// <para>Определяет, равна ли текущая отметка времени другой отметке времени.</para>
78          /// <para>Indicates whether the current Timestamp is equal to another Timestamp.</para>
79          /// </summary>
80          /// <param name="other"><para>Other Timestamp.</para><para>Другая отметка
            ↪  времени.</para></param>
81          /// <returns><para>True if the current Timestamp is equal to the other Timestamp;
            ↪  otherwise, false.</para><para>Истину, если текущая отметка времени равна другой
            ↪  отметке времени; иначе ложь.</para></returns>
82          [MethodImpl(MethodImplOptions.AggressiveInlining)]
83          public bool Equals(Timestamp other) => Ticks == other.Ticks;
84
85          /// <summary>
86          /// <para>Determines whether the specified object is equal to the current object.</para>
87          /// <para>Определяет, равен ли указанный объект текущему объекту.</para>
88          /// </summary>
89          /// <param name="obj"><para>The object to compare with the current
            ↪  object.</para><para>Объект для сравнения с текущим объектом.</para></param>
90          /// <returns><para>True if the specified object is equal to the current object;
            ↪  otherwise, false.</para><para>Истину, если указанный объект равен текущему объекту;
            ↪  иначе ложь.</para></returns>
91          [MethodImpl(MethodImplOptions.AggressiveInlining)]
92          public override bool Equals(object obj) => obj is Timestamp timestamp ?
            ↪  Equals(timestamp) : false;
93
94          /// <summary>
95          /// <para>Serves as the default hash function.</para>
96          /// <para>Служит в качестве хэш-функции по умолчанию.</para>
97          /// </summary>
98          /// <returns><para>A hash code for the current object.</para><para>Хеш-код для текущего
            ↪  объекта.</para></returns>
99          [MethodImpl(MethodImplOptions.AggressiveInlining)]
100         public override int GetHashCode() => Ticks.GetHashCode();
101
102         /// <summary>
103         /// <para>Determines if the specified timestamp is equal to the current timestamp.</para>
104         /// <para>Определяет, равна ли указанная метка времени текущей метке времени.</para>
105         /// </summary>
106         /// <param name="left"><para>The current timestamp.</para><para>Текущая метка
            ↪  времени.</para></param>
107         /// <param name="right"><para>A timestamp to compare with this
            ↪  timestamp.</para><para>Метка времени для сравнения с этой меткой
            ↪  времени.</para></param>
108         /// <returns><para>True if the current timestamp is equal to the other timestamp;
            ↪  otherwise, false.</para><para>True, если текущий метка времени равна другой метке
            ↪  времени; иначе false.</para></returns>
109         [MethodImpl(MethodImplOptions.AggressiveInlining)]
110         public static bool operator ==(Timestamp left, Timestamp right) => left.Equals(right);
111
112         /// <summary>
113         /// <para>Determines if the specified timestamp is not equal to the current
            ↪  timestamp.</para>
114         /// <para>Определяет, не равна ли указанная метка времени текущей метке времени.</para>
115         /// </summary>
116         /// <param name="left"><para>The current timestamp.</para><para>Текущая метка
            ↪  времени.</para></param>
```

```
        /// <param name="right"><para>A timestamp to compare with this timestamp.</para><para>Метка времени для сравнения с этой меткой времени.</para></param>
        /// <returns><para>True if the current timestamp is not equal to the other timestamp; otherwise, false.</para><para>True, если текущий метка времени не равна другой метке времени; иначе false.</para></returns>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public static bool operator !=(Timestamp left, Timestamp right) => !(left == right);
    }
}
```

1.2 ./csharp/Platform.Timestamps/UniqueTimestampFactory.cs

```
using System;
using System.Runtime.CompilerServices;
using Platform.Interfaces;

namespace Platform.Timestamps
{
    /// <summary>
    /// <para>Represents a factory for creating unique timestamps.</para>
    /// <para>Представляет фабрику по созданию уникальных отметок времени.</para>
    /// </summary>
    public class UniqueTimestampFactory : IFactory<Timestamp>
    {
        private ulong _lastTicks;

        /// <summary>
        /// <para>Creates a timestamp corresponding to the current UTC date and time or next unique timestamp.</para>
        /// <para>Создаёт отметку времени соответствующую текущей дате и времени по UTC или следующую уникальную отметку времени.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Timestamp Create()
        {
            var utcTicks = (ulong)DateTime.UtcNow.Ticks;
            _lastTicks = utcTicks > _lastTicks ? utcTicks : _lastTicks + 1;
            return new Timestamp(_lastTicks);
        }
    }
}
```

1.3 ./csharp/Platform.Timestamps.Tests/UniqueTimestampFactoryTests.cs

```
using Xunit;

namespace Platform.Timestamps.Tests
{
    public class UniqueTimestampFactoryTests
    {
        [Fact]
        public void UniqueTimestampTest()
        {
            var factory = new UniqueTimestampFactory();
            var timestamp1 = factory.Create();
            var timestamp2 = factory.Create();
            Assert.NotEqual(timestamp1, timestamp2);
        }
    }
}
```

Index