

Как делать разработчикам больно с помощью даты и времени

Александр Кузнецов
ведущий разработчик

Контур

DOT
NEXT



kontur.ru

Сколько дней в году?

Поставщики бензина Z Energy, Allied Petroleum и Gul отключили в Новой Зеландии свои комплексы автоматизированных АЗС (работали в режиме самообслуживание) из-за проблем с некорректной обработкой ПО комплексов текущей даты 29 февраля 2024 года.

36 багов 29 февраля 2024



<https://habr.com/ru/news/797047/>

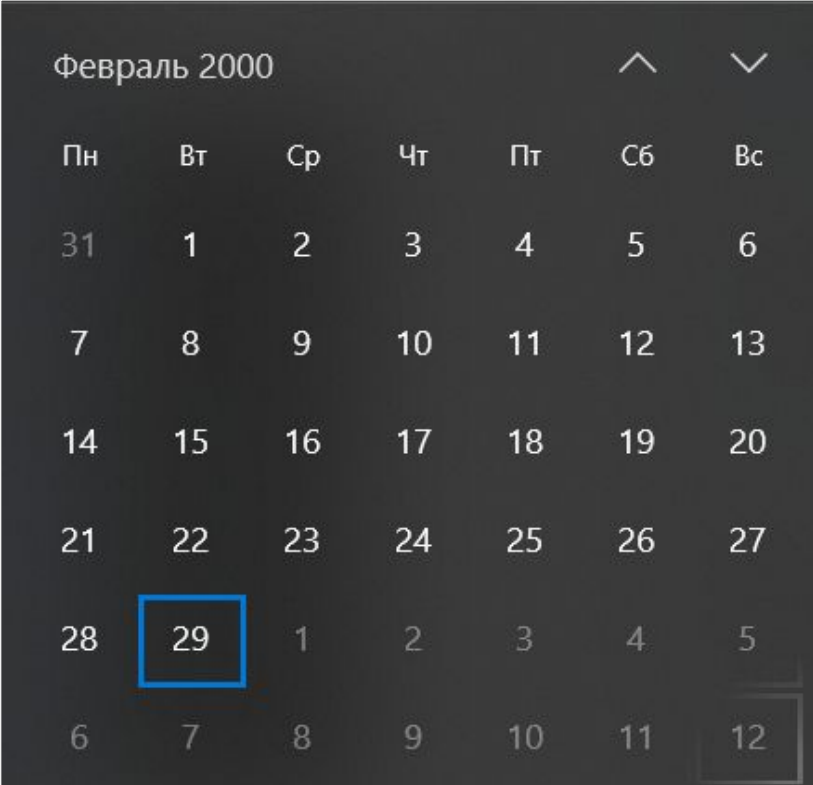


<https://habr.com/ru/articles/797569/>

Февраль 2024							^	v
Пн	Вт	Ср	Чт	Пт	Сб	Вс		
29	30	31	1	2	3	4		
5	6	7	8	9	10	11		
12	13	14	15	16	17	18		
19	20	21	22	23	24	25		
26	27	28	29	1	2	3		
4	5	6	7	8	9	10		

Високосный год

- Если номер года делится на 4, то он високосный
- Но если номер года делится на 100, то не високосный
- Но если он делится на 400, то всё-таки високосный



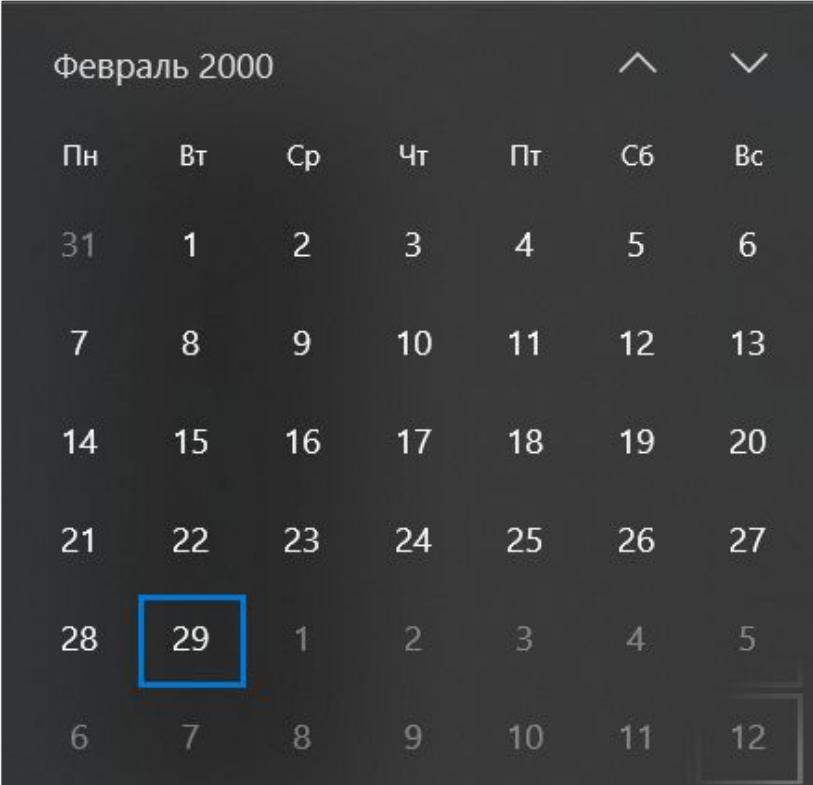
Февраль 2000

Пн	Вт	Ср	Чт	Пт	Сб	Вс
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	1	2	3	4	5
6	7	8	9	10	11	12

The image shows a calendar for February 2000. The days of the week are listed at the top: Пн (Monday), Вт (Tuesday), Ср (Wednesday), Чт (Thursday), Пт (Friday), Сб (Saturday), and Вс (Sunday). The dates are arranged in a grid. The date 29 is highlighted with a blue square, indicating that February 2000 is a leap year. The date 12 is also highlighted with a grey square.

Високосный год

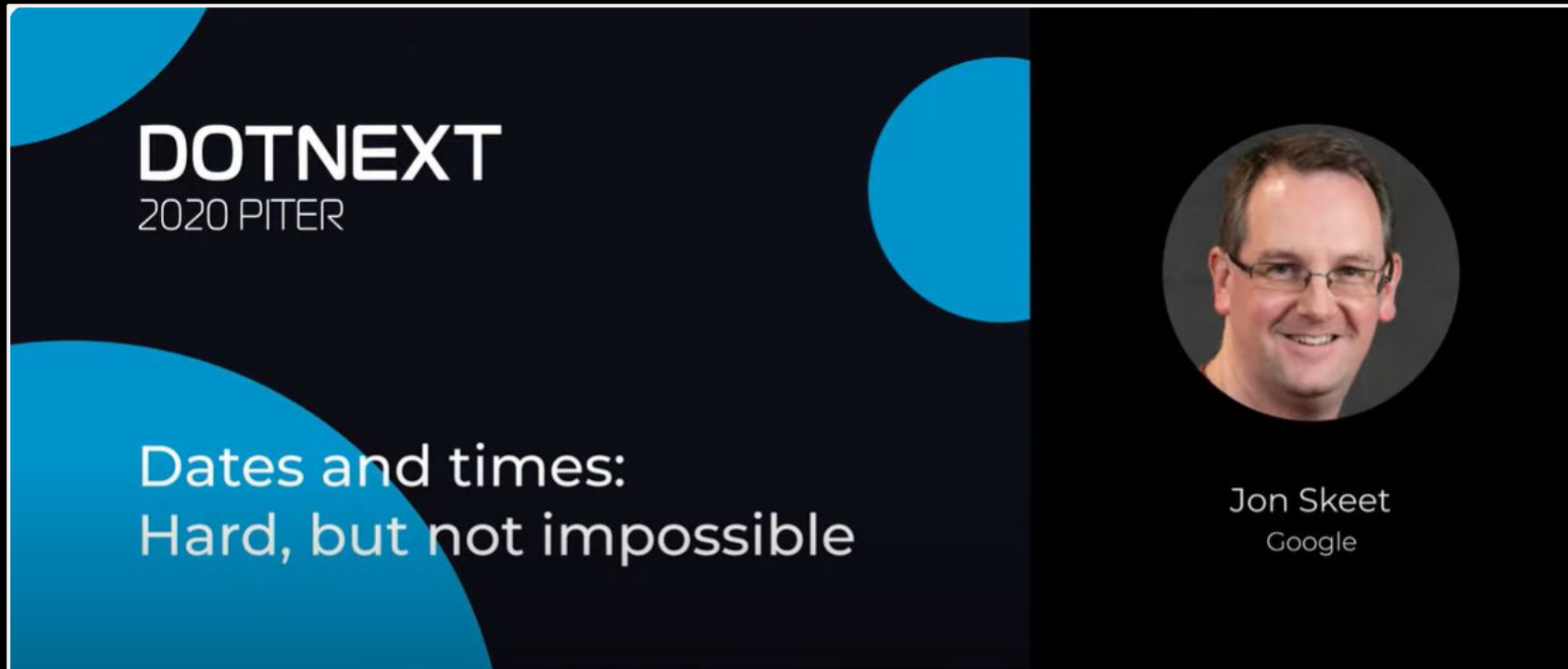
- Если номер года делится на 4, то он високосный
- Но если номер года делится на 100, то не високосный
- Но если он делится на 400, то всё-таки високосный
- Это всё ещё не точно. Ошибка в полные сутки накопится за 3300 лет



Февраль 2000


Пн	Вт	Ср	Чт	Пт	Сб	Вс
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	1	2	3	4	5
6	7	8	9	10	11	12

Ключевые концепции



DOTNEXT
2020 PITER

Dates and times:
Hard, but not impossible



Jon Skeet
Google

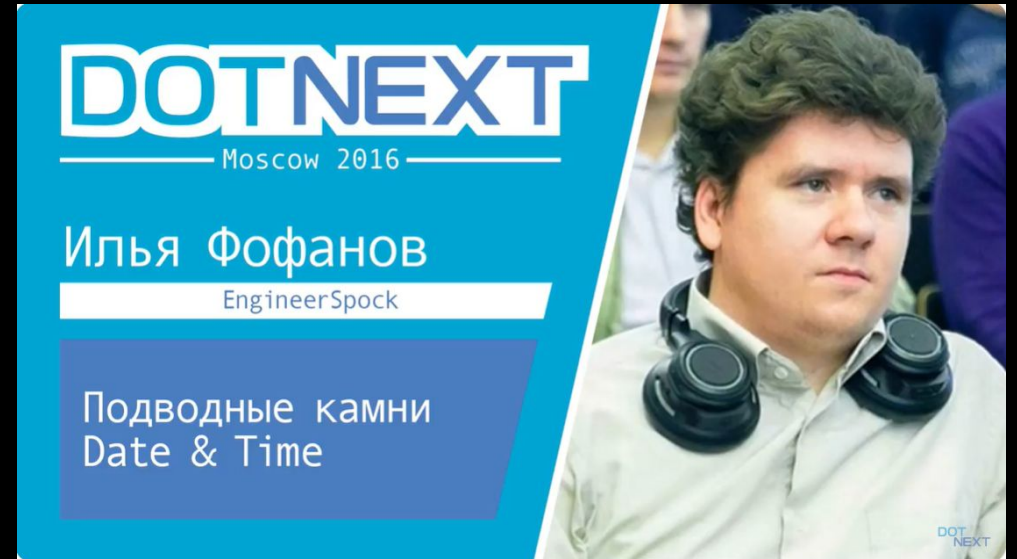
The image shows a promotional graphic for DOTNEXT 2020 PITER. It features a dark background with blue curved shapes. On the left, the event name 'DOTNEXT 2020 PITER' is written in white. Below it, the text 'Dates and times: Hard, but not impossible' is displayed. On the right, there is a circular portrait of Jon Skeet, a man with glasses, smiling. Below the portrait, his name 'Jon Skeet' and affiliation 'Google' are listed.



<https://www.youtube.com/watch?v=3j5jmbqGKEM>

Работа с датами в .NET

- DateTime
- DateTimeOffset
- DateTimeKind
- TimeSpan
- TimeZoneInfo
- CultureInfo



<https://www.youtube.com/watch?v=14I2JagZSlo>



Никогда так не делайте

Никогда так не делайте

```
var now = DateTime.Now;  
var nextYear = new DateTime(now.Year + 1,  
    now.Month, now.Day);
```

1

Никогда так не делайте

```
var now = DateTime.Now;  
var nextYear = new DateTime(now.Year + 1,  
    now.Month, now.Day);
```

1

```
const int WHOLE_YEAR_DAYS = 365;
```

2

Никогда так не делайте

```
var now = DateTime.Now;  
var nextYear = new DateTime(now.Year + 1,  
    now.Month, now.Day);
```

1

```
const int WHOLE_YEAR_DAYS = 365;
```

2

```
var call = new CallHelper();  
call.Start(DateTime.Now.Year, DateTime.Now.Month,  
    DateTime.Now.Day, DateTime.Now.Hour,  
    DateTime.Now.Minute, DateTime.Now.Second);  
  
// Long call  
  
call.Finish(DateTime.Now.Year, DateTime.Now.Month,  
    DateTime.Now.Day, DateTime.Now.Hour,  
    DateTime.Now.Minute, DateTime.Now.Second);  
  
var price = call.GetPrice();  
Console.WriteLine(price);
```

3

Никогда так не делайте

```
var now = DateTime.Now;
var nextYear = new DateTime(now.Year + 1,
    ... now.Month, now.Day);
```

1

```
const int WHOLE_YEAR_DAYS = 365;
```

2


```
var call = new CallHelper();
call.Start(DateTime.Now.Year, DateTime.Now.Month,
    ... DateTime.Now.Day, DateTime.Now.Hour,
    ... DateTime.Now.Minute, DateTime.Now.Second);

// Long call

call.Finish(DateTime.Now.Year, DateTime.Now.Month,
    ... DateTime.Now.Day, DateTime.Now.Hour,
    ... DateTime.Now.Minute, DateTime.Now.Second);

var price = call.GetPrice();
Console.WriteLine(price);
```

3



Никогда так не делайте

```
var now = DateTime.Now;
var nextYear = new DateTime(now.Year + 1,
    ... now.Month, now.Day);
```

1

```
const int WHOLE_YEAR_DAYS = 365;
```

2

```
var call = new CallHelper();
call.Start(DateTime.Now.Year, DateTime.Now.Month,
    ... DateTime.Now.Day, DateTime.Now.Hour,
    ... DateTime.Now.Minute, DateTime.Now.Second);

// Long call

call.Finish(DateTime.Now.Year, DateTime.Now.Month,
    ... DateTime.Now.Day, DateTime.Now.Hour,
    ... DateTime.Now.Minute, DateTime.Now.Second);

var price = call.GetPrice();
Console.WriteLine(price);
```

3

```
var now = DateTime.Now;
```

4

Никогда так не делайте

```
var now = DateTime.Now;
var nextYear = new DateTime(now.Year + 1,
    now.Month, now.Day);
```

1

```
var call = new CallHelper();
call.Start(DateTime.Now.Year, DateTime.Now.Month,
    DateTime.Now.Day, DateTime.Now.Hour,
    DateTime.Now.Minute, DateTime.Now.Second);

// Long call

call.Finish(DateTime.Now.Year, DateTime.Now.Month,
    DateTime.Now.Day, DateTime.Now.Hour,
    DateTime.Now.Minute, DateTime.Now.Second);

var price = call.GetPrice();
Console.WriteLine(price);
```

3

```
const int WHOLE_YEAR_DAYS = 365;
```

2

```
var now = DateTime.Now;
```

4

```
var utcNow = DateTime.UtcNow;
```

5

Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999
 - 01/01/1970 (начало эпохи UNIX)



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999
 - 01/01/1970 (начало эпохи UNIX)
 - 01/01/2000 и 01/01/2100 (и в принципе 100)

В 24-м году...



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999
 - 01/01/1970 (начало эпохи UNIX)
 - 01/01/2000 и 01/01/2100 (и в принципе 100)
 - 01/01/2030 (.NET6) и 01/01/2050 (.NET8)

В 34-м году...



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999
 - 01/01/1970 (начало эпохи UNIX)
 - 01/01/2000 и 01/01/2100 (и в принципе 100)
 - 01/01/2030 (.NET6) и 01/01/2050 (.NET8)

В 34-м году...

→ .NET6 => 1934

→ .NET8 => 2034



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999
 - 01/01/1970 (начало эпохи UNIX)
 - 01/01/2000 и 01/01/2100 (и в принципе 100)
 - 01/01/2030 (.NET6) и 01/01/2050 (.NET8)
 - 19/01/2038 03:14:07 (32-разрядные системы) и 2486 год (Linux 5.1+)



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999
 - 01/01/1970 (начало эпохи UNIX)
 - 01/01/2000 и 01/01/2100 (и в принципе 100)
 - 01/01/2030 (.NET6) и 01/01/2050 (.NET8)
 - 19/01/2038 03:14:07 (32-разрядные системы) и 2486 год (Linux 5.1+)
 - Смена часа/дня/месяца/года



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999
 - 01/01/1970 (начало эпохи UNIX)
 - 01/01/2000 и 01/01/2100 (и в принципе 100)
 - 01/01/2030 (.NET6) и 01/01/2050 (.NET8)
 - 19/01/2038 03:14:07 (32-разрядные системы) и 2486 год (Linux 5.1+)
 - Смена часа/дня/месяца/года
 - 29 февраля и интервалы с ним



Тестируемые дата и время

- Для тестов “текущие” дата и время должны задаваться вручную
- Критичные моменты:
 - 01/01/0001 и 31/12/9999
 - 01/01/1970 (начало эпохи UNIX)
 - 01/01/2000 и 01/01/2100 (и в принципе 100)
 - 01/01/2030 (.NET6) и 01/01/2050 (.NET8)
 - 19/01/2038 03:14:07 (32-разрядные системы) и 2486 год (Linux 5.1+)
 - Смена часа/дня/месяца/года
 - 29 февраля и интервалы с ним
 - Конец високосного года



Тестируемые дата и время

1 reference

```
public interface ISystemClock
```

```
{
```

1 reference

```
    public DateTime GetUtcNow();
```

```
}
```

0 references

```
public class SystemClock : ISystemClock
```

```
{
```

1 reference

```
    public DateTime GetUtcNow()
```

```
    {
```

```
        return DateTime.UtcNow;
```

```
    }
```

```
}
```


Тестируемые дата и время

```
public bool IsFuture(DateTime date, DateTime? now = null)
{
    if (now == null)
        now = DateTime.Now;

    return date >= now;
}
```

Тестируемые дата и время (.NET 8)

```
public abstract class TimeProvider
{
    0 references
    ... public static TimeProvider System { get; }
    0 references
    ... public virtual DateTimeOffset GetUtcNow();
    0 references
    ... public DateTimeOffset GetLocalNow();
    0 references
    ... public virtual TimeZoneInfo LocalTimeZone { get; }
    0 references
    ... public virtual long TimestampFrequency { get; }
    0 references
    ... public virtual long GetTimestamp();
    0 references
    ... public TimeSpan GetElapsedTime(long startingTimestamp);
    0 references
    ... public TimeSpan GetElapsedTime(long startingTimestamp, long endingTimestamp);
    0 references
    ... public virtual ITimer CreateTimer(TimerCallback callback, object? state, TimeSpan dueTime, TimeSpan period);
}
```

Арифметика дат

```
var date = new DateTime(2025, 1, 30);  
var date1 = date.AddMonths(1).AddDays(1);  
var date2 = date1.AddDays(-1).AddMonths(-1);  
  
Console.WriteLine(date);  
Console.WriteLine(date1);  
Console.WriteLine(date2);
```

Арифметика дат

```
var date = new DateTime(2025, 1, 30);  
var date1 = date.AddMonths(1).AddDays(1);  
var date2 = date1.AddDays(-1).AddMonths(-1);  
  
Console.WriteLine(date);  
Console.WriteLine(date1);  
Console.WriteLine(date2);
```

```
30.01.2025 0:00:00  
01.03.2025 0:00:00  
28.01.2025 0:00:00
```

Арифметика дат

```
var date1 = new DateTime(2025, 1, 28).AddMonths(1);  
var date2 = new DateTime(2025, 1, 29).AddMonths(1);  
var date3 = new DateTime(2025, 1, 30).AddMonths(1);  
var date4 = new DateTime(2025, 1, 31).AddMonths(1);  
  
Console.WriteLine(date1);  
Console.WriteLine(date2);  
Console.WriteLine(date3);  
Console.WriteLine(date4);
```

```
28.02.2025 0:00:00  
28.02.2025 0:00:00  
28.02.2025 0:00:00  
28.02.2025 0:00:00
```

Немного о календарях

Сколько месяцев в году?

```
string[] monthNames =  
    System.Globalization.CultureInfo.InvariantCulture  
        .DateTimeFormat.MonthNames;  
  
foreach (var name in monthNames)  
{  
    Console.WriteLine($"{name}");  
}  
  
Console.WriteLine(monthNames.Length);
```

Немного о календарях

Сколько месяцев в году?

```
string[] monthNames =  
    System.Globalization.CultureInfo.InvariantCulture  
        .DateTimeFormat.MonthNames;  
  
foreach (var name in monthNames)  
{  
    Console.WriteLine($"{name}");  
}  
  
Console.WriteLine(monthNames.Length);
```

```
"January"  
"February"  
"March"  
"April"  
"May"  
"June"  
"July"  
"August"  
"September"  
"October"  
"November"  
"December"  
"  
13
```

Немного о календарях

А какой сейчас год?

```
var cultures = CultureInfo
    .GetCultures(CultureTypes.AllCultures);

var date = new DateTime(2024, 01, 02);
foreach (var culture in cultures)
{
    var year = culture.Calendar.GetYear(date);
    if (year != 2024)
        Console.WriteLine($"{year} - {culture.Name}");
}
```


Немного о календарях

А какой сейчас год?

```
var cultures = CultureInfo
    .GetCultures(CultureTypes.AllCultures);

var date = new DateTime(2024, 01, 02);
foreach (var culture in cultures)
{
    var year = culture.Calendar.GetYear(date);
    if (year != 2024)
        Console.WriteLine($"{year} - {culture.Name}");
}
```

```
1445 - ar-SA
1402 - ckb-IR
1402 - fa
1402 - fa-AF
1402 - fa-IR
1402 - lrc
1402 - lrc-IR
1402 - mzn
1402 - mzn-IR
1402 - ps
1402 - ps-AF
2567 - th
2567 - th-TH
1402 - uz-Arab
1402 - uz-Arab-AF
```

И культурах

А какие это даты?

```
02/01/2024 00:00:00 - br-FR
1/2/2024 12:00:00 ??? - brx
1/2/2024 12:00:00 ??? - brx-IN
2. 1. 2024. 00:00:00 - bs
2.1.2024. 00:00:00 - bs-Cyrl
2.1.2024. 00:00:00 - bs-Cyrl-BA
2. 1. 2024. 00:00:00 - bs-Latn
2. 1. 2024. 00:00:00 - bs-Latn-BA
2024-01-02 00:00:00 - byn
2024-01-02 00:00:00 - byn-ER
2/1/2024 0:00:00 - ca
2/1/2024 0:00:00 - ca-AD
```

```
var cultures = CultureInfo
    .GetCultures(CultureTypes.AllCultures);

var date = new DateTime(2024, 01, 02);
foreach (var culture in cultures)
{
    CultureInfo.CurrentCulture = culture;
    Console.WriteLine($"{date} - {culture.Name}");
}
```


И культурах

А какие это даты?

```
2024-01-02T00:00:00.0000000 - br-FR
2024-01-02T00:00:00.0000000 - brx
2024-01-02T00:00:00.0000000 - brx-IN
2024-01-02T00:00:00.0000000 - bs
2024-01-02T00:00:00.0000000 - bs-Cyr1
2024-01-02T00:00:00.0000000 - bs-Cyr1-BA
2024-01-02T00:00:00.0000000 - bs-Latn
2024-01-02T00:00:00.0000000 - bs-Latn-BA
2024-01-02T00:00:00.0000000 - byn
2024-01-02T00:00:00.0000000 - byn-ER
2024-01-02T00:00:00.0000000 - ca
2024-01-02T00:00:00.0000000 - ca-AD
```

```
var cultures = CultureInfo
    .GetCultures(CultureTypes.AllCultures);

var date = new DateTime(2024, 01, 02);
foreach (var culture in cultures)
{
    CultureInfo.CurrentCulture = culture;
    Console.WriteLine($"{date:O} - {culture.Name}");
}
```



ISO 8601

YYYY-MM-DDThh:mm:ss[.SSS]

А ещё о поддержке стандартов...

ISO 8601

YYYY-MM-DDThh:mm:ss[.SSS]

2024-01-02T11:15:26.145



Microsoft SignalR



ISO 8601



SignalR для IOS

А ещё о поддержке стандартов...

ISO 8601

YYYY-MM-DDThh:mm:ss[.SSS]

2024-01-02T11:15:26.145

2024-01-02T11:15:26.100



Microsoft SignalR



ISO 8601



SignalR для IOS

А ещё о поддержке стандартов...

ISO 8601

YYYY-MM-DDThh:mm:ss[.SSS]

2024-01-02T11:15:26.145

2024-01-02T11:15:26.100

2024-01-02T11:15:26.1



Microsoft SignalR



ISO 8601



SignalR для IOS

А ещё о поддержке стандартов...

ISO 8601

```
class DateTimeConverter : JsonSerializer<DateTime>
{
    0 references
    public override void Write(Utf8JsonWriter writer, DateTime value,
        JsonSerializerOptions options)
    {
        writer.WriteStringValue(value.ToString("yyyy-MM-ddTHH:mm:ss.fff"));
    }
}
```

```
var options = new JsonSerializerOptions();
options.Converters.Add(new DateTimeConverter());
string strConverted = JsonSerializer.Serialize(date2, options);
Console.WriteLine($"{strConverted}");
```

2024-01-02T11:15:26.1

2024-01-02T11:15:26.100



Microsoft SignalR

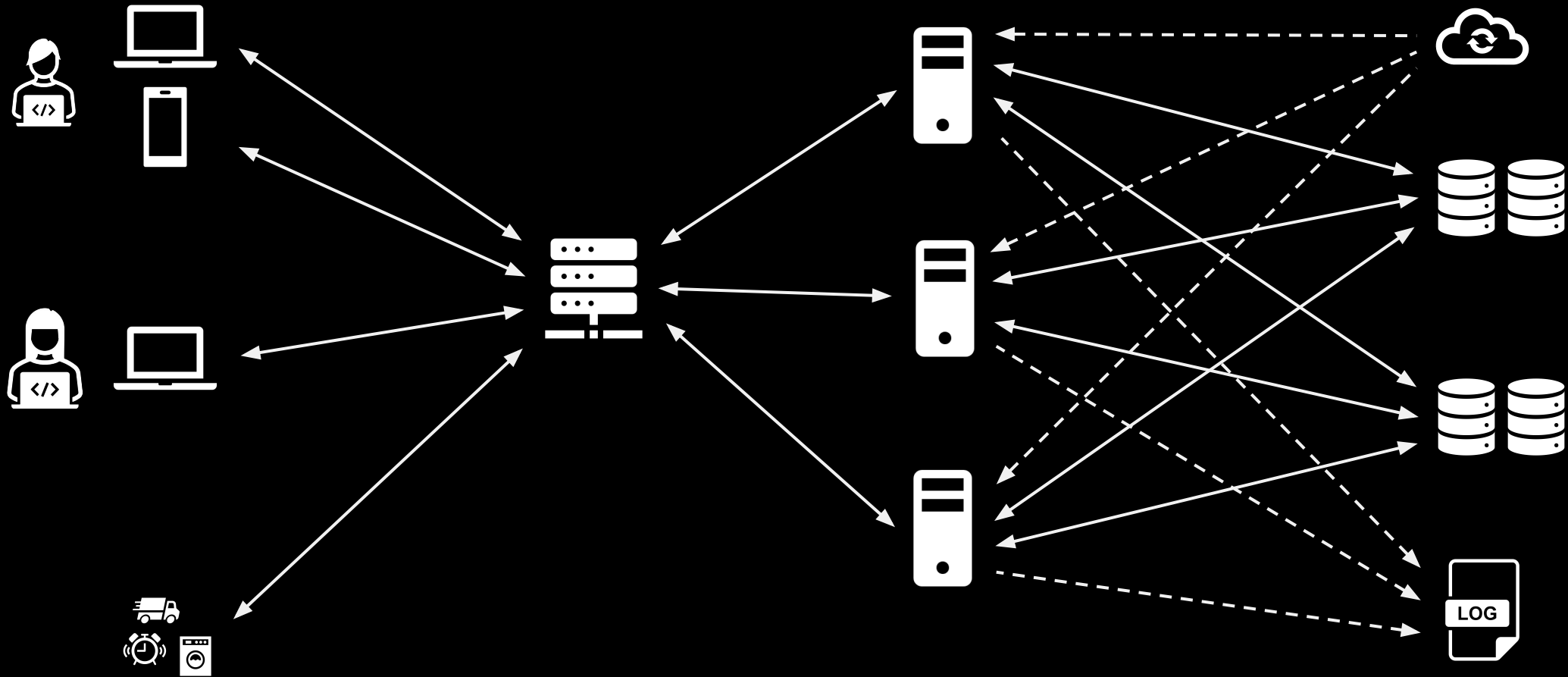


ISO 8601

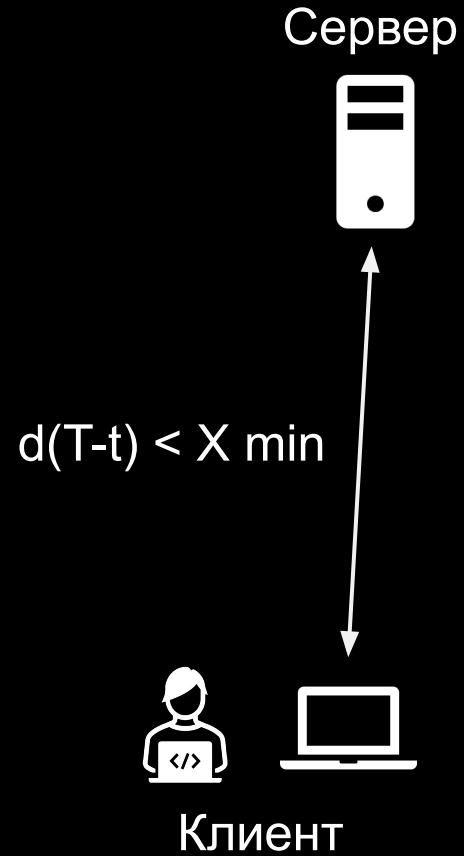


SignalR для iOS

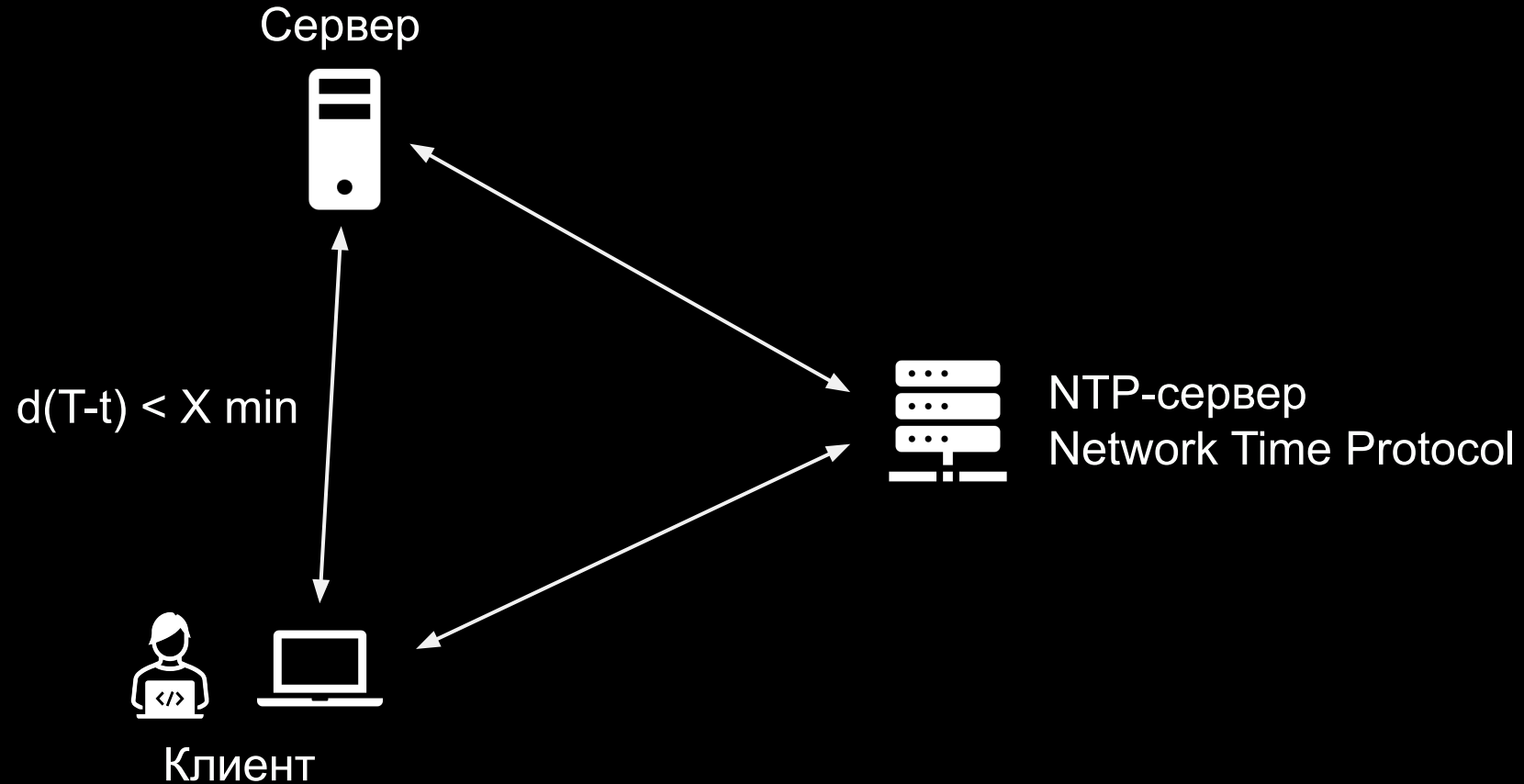
Современные приложения...



Откуда компьютеры знают о времени?



Откуда компьютеры знают о времени?



Время не всегда возрастает...

Разница по времени между вашим компьютером и остальными может накапливаться

Время с NTP-сервера (UTC+4) Время на компьютере (UTC+4)

09:38:06 249 09:38:05 345

Часы на вашем компьютере отстают на **0.904** сек.

Время с NTP-сервера показано по часовому поясу UTC+4 ([сменить](#))

Сейчас в Москве — 08:38:06 (UTC+3)

<https://www.ntp-servers.net>

Время не всегда возрастает...

Разница по времени между вашим компьютером и остальными может накапливаться

Время с NTP-сервера (UTC+4)	Время на компьютере (UTC+4)
09:38:06 ²⁴⁹	09:38:05 ³⁴⁵
Часы на вашем компьютере отстают на 0.904 сек.	
Время с NTP-сервера показано по часовому поясу UTC+4 (сменить)	
Сейчас в Москве — 08:38:06 (UTC+3)	

<https://www.ntp-servers.net>

```
2024-04-22T10:05:21.624
2024-04-22T10:05:21.641
2024-04-22T10:05:21.655
2024-04-22T10:05:21.671
2024-04-22T10:04:32.818
2024-04-22T10:04:32.834
2024-04-22T10:04:32.850
2024-04-22T10:04:32.866
2024-04-22T10:04:32.882
2024-04-22T10:04:32.897
```

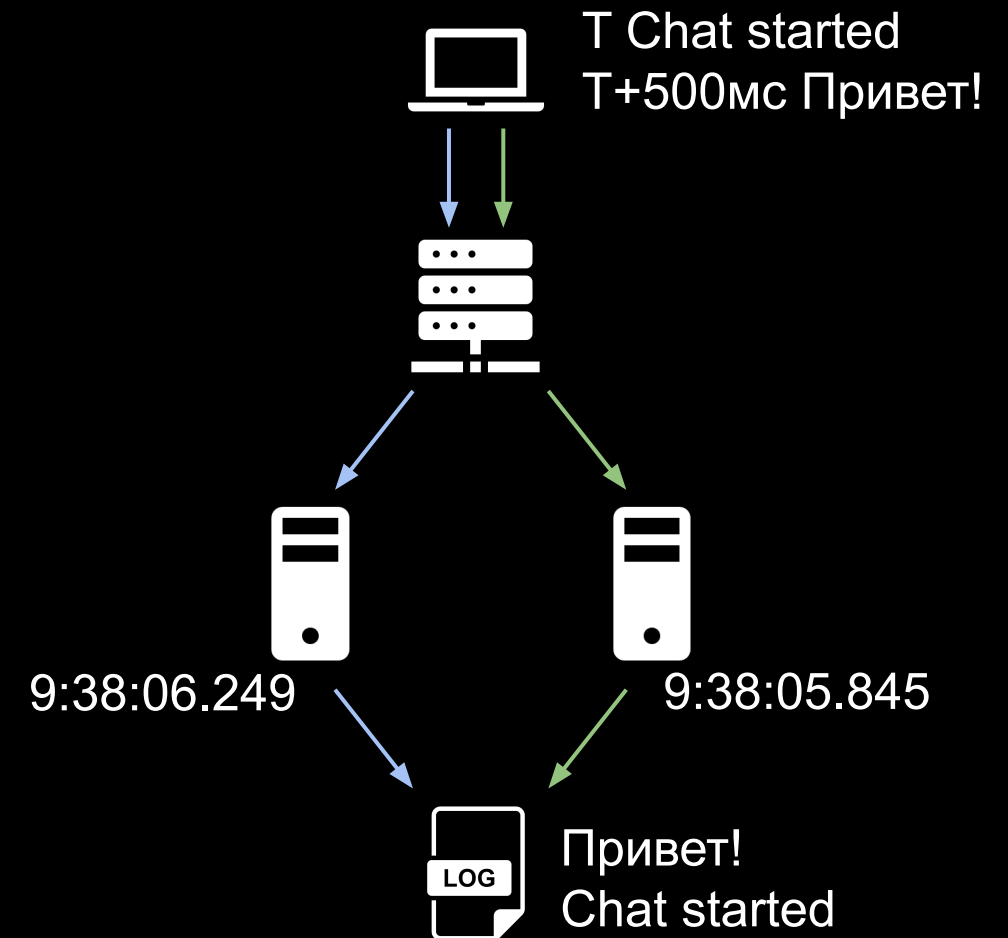
В момент синхронизации время меняется

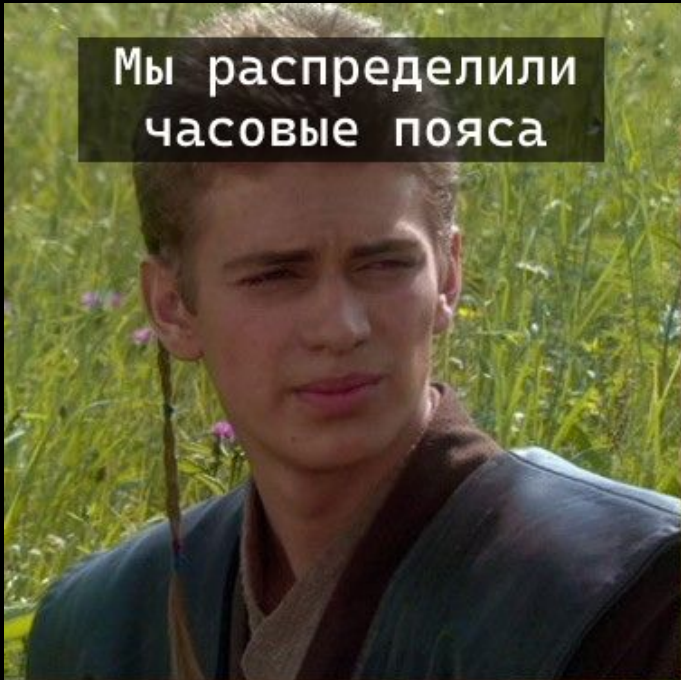
А различия бывают критичными...

Разница по времени между вашим компьютером и остальными может накапливаться

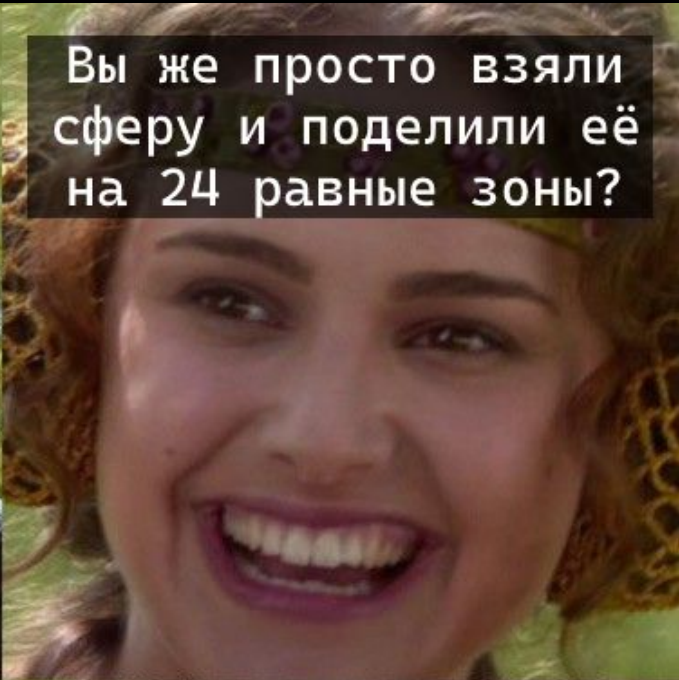
Время с NTP-сервера (UTC+4)	Время на компьютере (UTC+4)
09:38:06 ₂₄₉	09:38:05 ₃₄₅
Часы на вашем компьютере отстают на 0.904 сек.	
Время с NTP-сервера показано по часовому поясу UTC+4 (сменить)	
Сейчас в Москве — 08:38:06 (UTC+3)	

<https://www.ntp-servers.net>





Мы распределили
часовые пояса



Вы же просто взяли
сферу и поделили её
на 24 равные зоны?

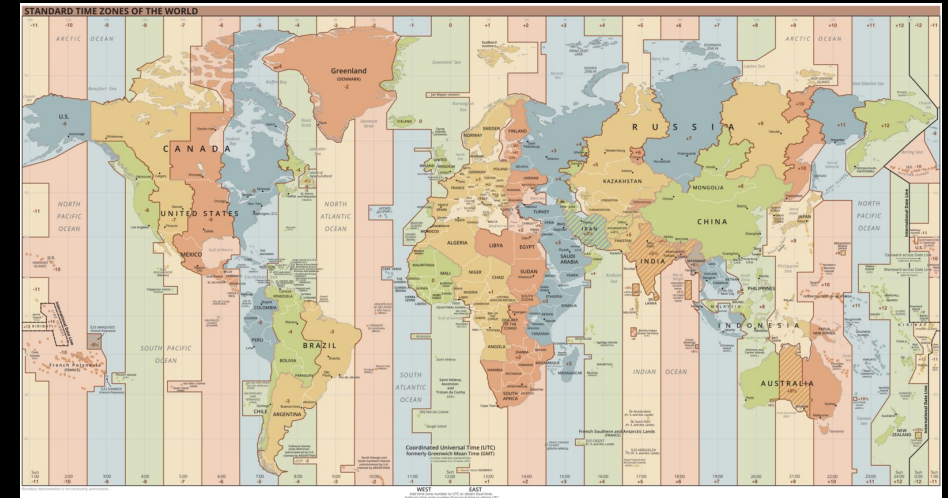


На 24 равные зоны? ..

И, наконец, таймзоны!

Всего таймзон 38:

- 23 «обычных» UTC-11... UTC+11
- UTC-12 и UTC+12
- 8 со сдвигом на полчаса
- 3 со сдвигом на 45 минут
- UTC+13 для атоллов Тонга

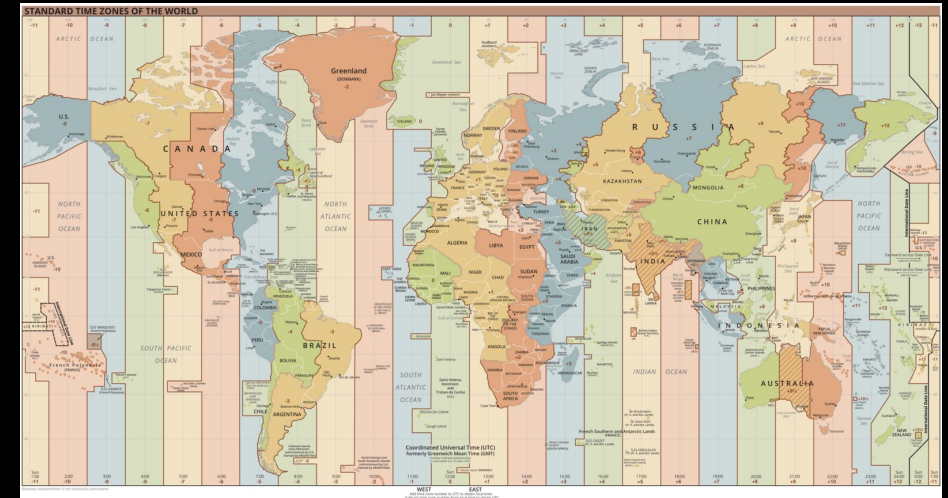


Временная география

И, наконец, таймзоны!

Всего таймзон 38:

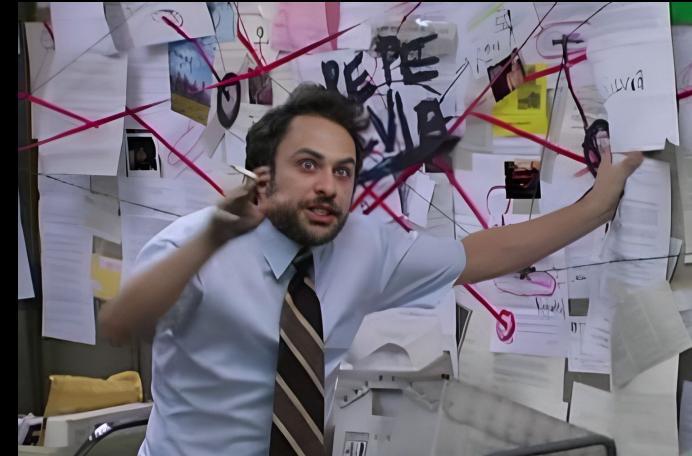
- 23 «обычных» UTC-11... UTC+11
- UTC-12 и UTC+12
- 8 со сдвигом на полчаса
- 3 со сдвигом на 45 минут
- UTC+13 для атоллов Тонга
- UTC+14 для атоллов Кирибати



Временная география

И, наконец, таймзоны!

- В круге радиусом километр может одновременно быть:
 - 14:00, 15:30, 15:45
 - Сегодня, завтра и послезавтра
- Максимальная разница по времени составляет 26 часов:
 - 13:05 Мск 10-е
 - 23:05 9-е UTC-11
 - 0:05 11-е UTC+14
- А ещё есть полюса...



Таймзоны - это просто!

Дело о пропавших сообщениях

- Скайп на трассе Саратов-Воронеж и смена таймзон



Дело о пропавших сообщениях

- Скайп на трассе Саратов-Воронеж и смена таймзон
- Что делать:
 - Отправлять время с таймзоной на сервер
 - Локально хранить время с таймзоной
 - Всегда проверять таймзону перед упорядочиванием
- Альтернатива — всё в UTC + при показе учёт локальной таймзоны

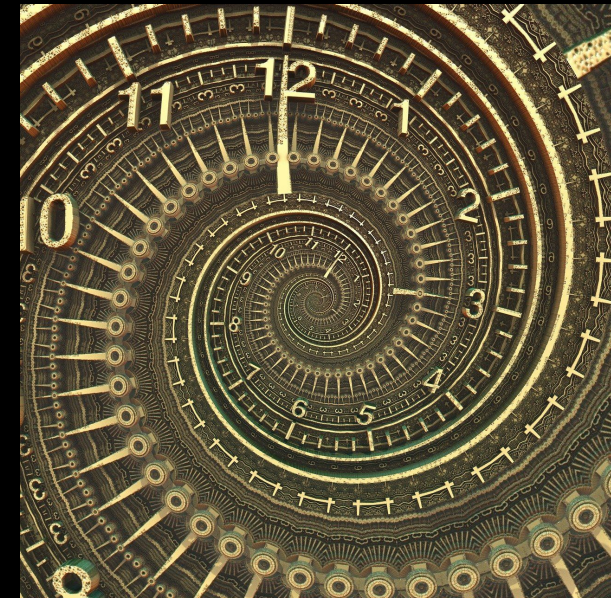


11:00 - это сколько?

Какое время имеется в виду при планировании?

- Локальное время пользователя
- Московское
- UTC
- Время сервера?

Запланировать публикацию 20.06.2024 11:00



Вам предоставили доступ к контенту 23 декабря - Последнее Испытание 20.00

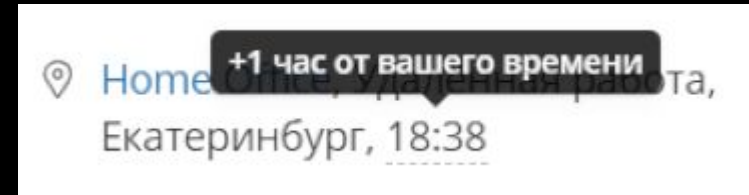
11:00 - это сколько?

Какое время имеется в виду при планировании?

- Локальное время пользователя
- Московское
- UTC
- Время сервера?

Что делать:

- Указывайте таймзону (17:00, UTC+4)
- Указывайте временной интервал (17:00, через 23 часа)
- Делайте приписку, что используете локальное время, или показывайте его рядом



Ещё немного таймзон...

- Таймзоны меняются. История/планы должны это учитывать



Ещё немного таймзон...

- Таймзоны меняются. История/планы должны это учитывать
- Что делать:
 - Исторические данные хранятся на серверах таймзон (<https://www.iana.org/time-zones>)
 - На будущее лучше запоминать координаты и считать таймзоны по ним по факту
 - Обязательно тестировать



Ещё немного таймзон...

- Разные регионы могут иметь свои записи для одних и тех же таймзон (всего 597)
- У разных компаний разные словари таймзон!

Время с NTP-сервера (UTC+4) Время на компьютере (UTC+4)

09:42:23₆₈₁ **09:42:22**₇₇₇

Часы на вашем компьютере отстают на **0.904** сек.

Время с NTP-сервера показано по часовому поясу UTC+4 ([сменить](#))

(UTC+04:00) Волгоград, Ижевск, Саратов, Самара (RTZ 3) ▾

(UTC+03:00) Найроби

(UTC+03:30) Тегеран

(UTC+04:00) Волгоград, Ижевск, Саратов, Самара (RTZ 3)

(UTC+04:00) Абу-Дави, Мускат

(UTC+04:00) Izhevsk, Samara

(UTC+04:00) Port Louis

(UTC+04:00) Saratov

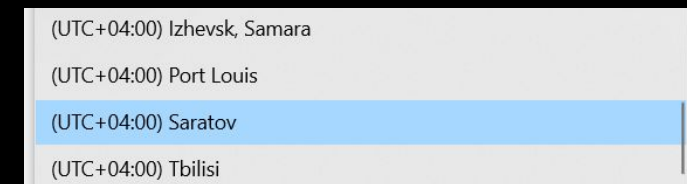
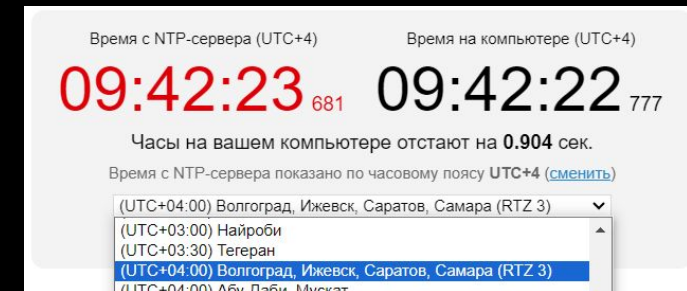
(UTC+04:00) Tbilisi

летнее время в Саратов в 2024 году

**MSK
UTC+0300**

Ещё немного таймзон...

- Разные регионы могут иметь свои записи для одних и тех же таймзон (всего 597)
- У разных компаний разные словари таймзон!
- Что делать:
 - Использовать единый стандарт в рамках компании / продукта
 - Валидировать данные при работе с внешними потребителями и быть готовым к неточностям
 - Обязательно тестировать



Ещё немного таймзон...

- Переход на летнее / зимнее время
 - Проблемы задваивания / пропадания времени, алертинг
 - Временные интервалы. Как должен вести себя интервальный будильник?



Ещё немного таймзон...

- Переход на летнее / зимнее время
 - Проблемы задваивания / пропадания времени, алертинг
 - Временные интервалы. Как должен вести себя интервальный будильник?
- Что делать:
 - Держать сервера в таймзоне без переходов (лучше в UTC)
 - Единообразно учитывать переходы при работе с клиентским временем
 - Обязательно тестировать



Интервалы...

- Контрольные keep-alive каждые N секунд и...



Интервалы...

- Контрольные keep-alive каждые N секунд и...
 - “Отвалившийся провод”
 - “Самолётный” режим
 - Закрытая крышка ноутбука
 - Неактивные вкладки в мобильных браузерах



Интервалы...

- Контрольные keep-alive каждые N секунд и...
 - “Отвалившийся провод”
 - “Самолётный” режим
 - Закрытая крышка ноутбука
 - Неактивные вкладки в мобильных браузерах
- Что делать:
 - Быть готовым к тому, что “мёртвые” клиенты могут оживать
 - Дублировать промежуточные данные на клиентах
 - Я вам ещё не надоел с тестированием?



Call to action

- Выкиньте глобус своего города/страны/региона
- Используйте стандартные/популярные решения вместо велосипедов
- Пишите код, пригодный для ручного/машинного тестирования времени
- Планируйте покрытие тестами краевых случаев



Спасибо за внимание!

Вопросы?

Александр Кузнецов
ведущий разработчик