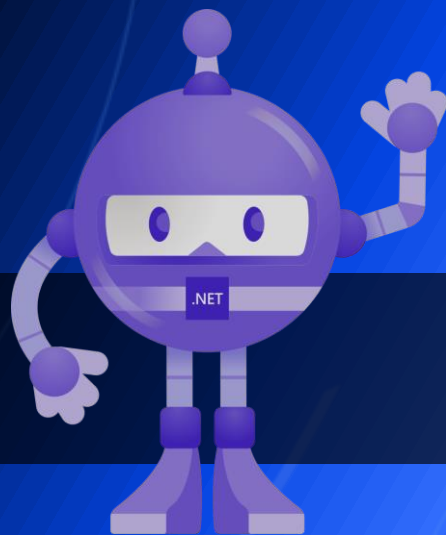
ЧТО НОВОГО

В .NET9 И C#13



@SBenzenko

Сергей Бензенко

Картинка: https://commons.wikimedia.org/wiki/File:Dotnet-bot.svg

home.asp

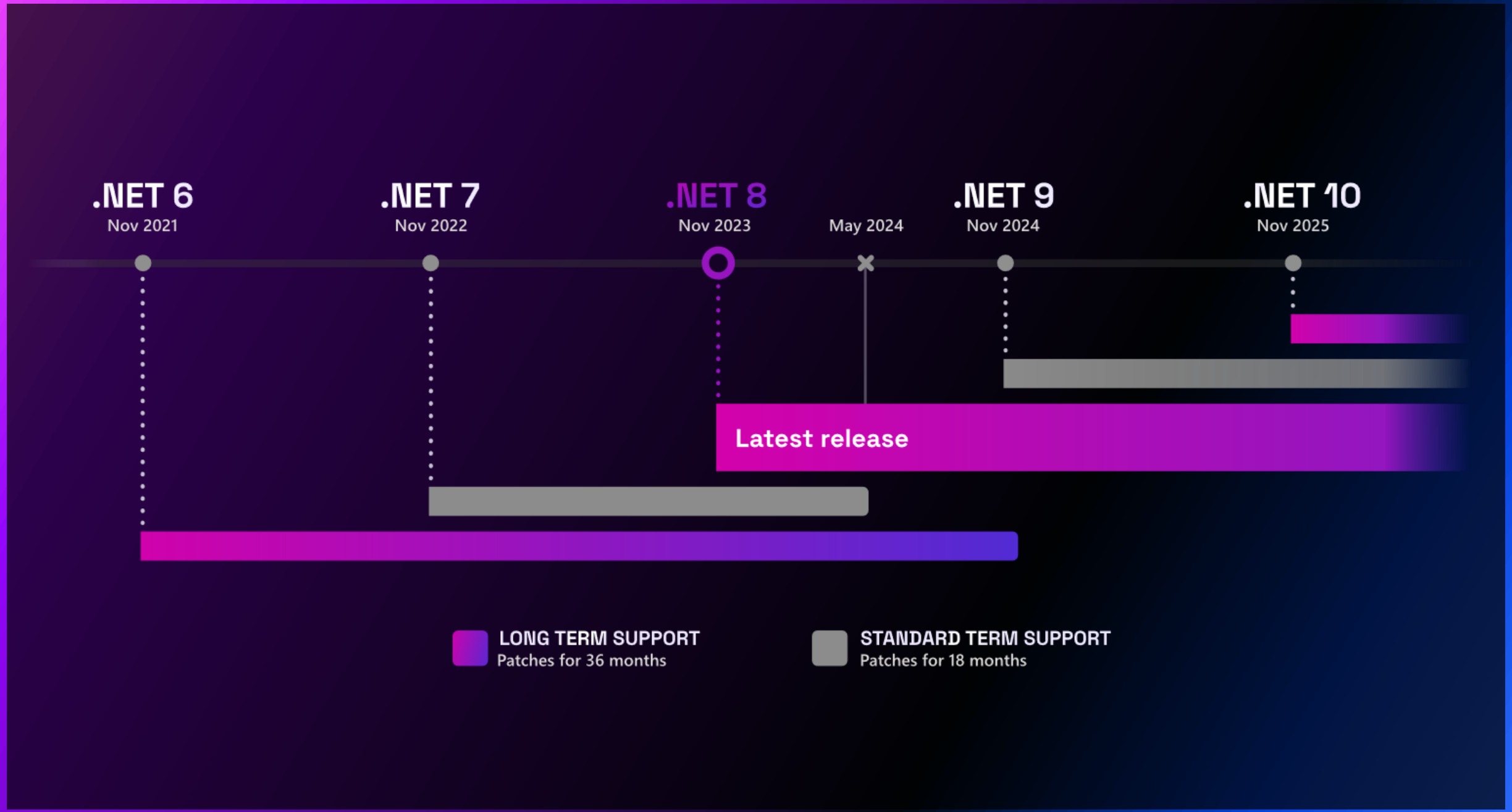
```
<!--#INCLUDE FILE=include/functions.asp-->

<%
    Dim iTranType : iTranType = CInt(RQ("tp"))

    if ( RQ("di") = "" and GetSes("sup_id") = "" and RQ("trans") = "" ) then _
        AbortToPage iSecure, GLM("SES_EXP"), "history.go(-2);", SIDE_NOTIFY

    Dim bCheckSend  : bCheckSend  = True
    Dim sNtfAction  : sNtfAction  = "CLOSE"
    Dim iMarkAsRead : iMarkAsRead = 0
    Dim sTransID    : sTransID    = RQ("trans")
    Dim sPriceFld   : sPriceFld   = GLM("PRICE")
   Dim sCurrMsg    : sCurrMsg    = GLM("SPECIFY_CUR")

%>

<BODY TOPMARGIN=5 LEFTMARGIN=5 RIGHTMARGIN=5 BOTTOMMARGIN=5 LINK=<%=sLink%> ALINK=<%=sAlink%> VLINK=<%=sVlink%>>
<form NAME="f" METHOD="POST" ACTION="home.htm?tp=<%=iTranType%>#form">
<input TYPE="HIDDEN" NAME="act" VALUE="">
<input TYPE="HIDDEN" NAME="sbt" VALUE="">

<table WIDTH="100%" CELLPADDING="0" CELLSPACING="0" BORDER="0">
    <tr>
        <td>
            <SPAN CLASS=title><%=sTitle%></span><br>
            <img SRC="../img/dtc.gif" WIDTH=4><br>

      <td ALIGN=RIGHT WIDTH=25% CLASS=small NOWRAP>

            <A onmouseover="st('<%=GLM("SUP_INQ")%>');return true" onmouseout="st(ds);return true" HREF="Javascript:history.go(-1);"><%=GLM("SUP_INQ")%></a><img SRC="../img/dtc.gif" WIDTH=8><br>
            <A onmouseover="st('<%=GLM("CLOSE")%>');return true" onmouseout="st(ds);return true" HREF="Javascript:window.close();"><%=GLM("CLOSE")%></a><img SRC="../img/dtc.gif" WIDTH=8></td>

            <img SRC="../img/dtc.gif" WIDTH=8><br>


<% if ( iStyle <= 1 ) then %>

            <SPAN>

            <center>
            <a onmouseover="return true;" onmouseout="st(ds);return true;" HREF="Javascript:act(document.f.txtAdd.value)"><FONT COLOR=BLACK><b><%=Replace(GLM("ADD_LINES"),"[NUM]","</a><INPUT NAME=txtAdd SIZE=2 MAXLENGTH=1 VALUE=1><a onmouseover=""return true;"" onmouseout=""st(ds);return true;"" HREF=Javascript:act(document.f.txtAdd.value)><FONT COLOR=BLACK>")%><b></font></a>
              |  
            <a onmouseover="return true;" onmouseout="st(ds);return true;" HREF="Javascript:act(0)"><FONT COLOR=BLACK><b><%=GLM("DEL_CHECKED")%></b></font></a><br>
            </center>

            <img SRC="../img/dtc.gif" HEIGHT="4"><br>

<%  end if %>

            <table WIDTH="100%" CELLPADDING="1" CELLSPACING="0" BORDER="0" BGCOLOR=#CC0033>
                <tr CLASS=headbar>
                    <td ALIGN="MIDDLE" NOWRAP>
                         <input TYPE="CHECKBOX" NAME="parts" VALUE="0" onclick="bchk(this.checked)" CHECKED> </td>
                    <td NOWRAP VALIGN="BOTTOM" CLASS="headbar"><font color=BLACK><b><%=GLM("PN")%> </td>
                    <td NOWRAP VALIGN="BOTTOM" CLASS="headbar"><font color=BLACK><b><%=GLM("MFR")%> </font></td>
                    <td NOWRAP VALIGN="BOTTOM" CLASS="headbar"><font color=BLACK><b><%=GLM("DC")%> </font></td>
                    <td NOWRAP WIDTH="100%" VALIGN="BOTTOM" CLASS="headbar"><font color=BLACK><b><%=GLM("DESC")%> </font></td>
```

1
2 | {.Net}
3

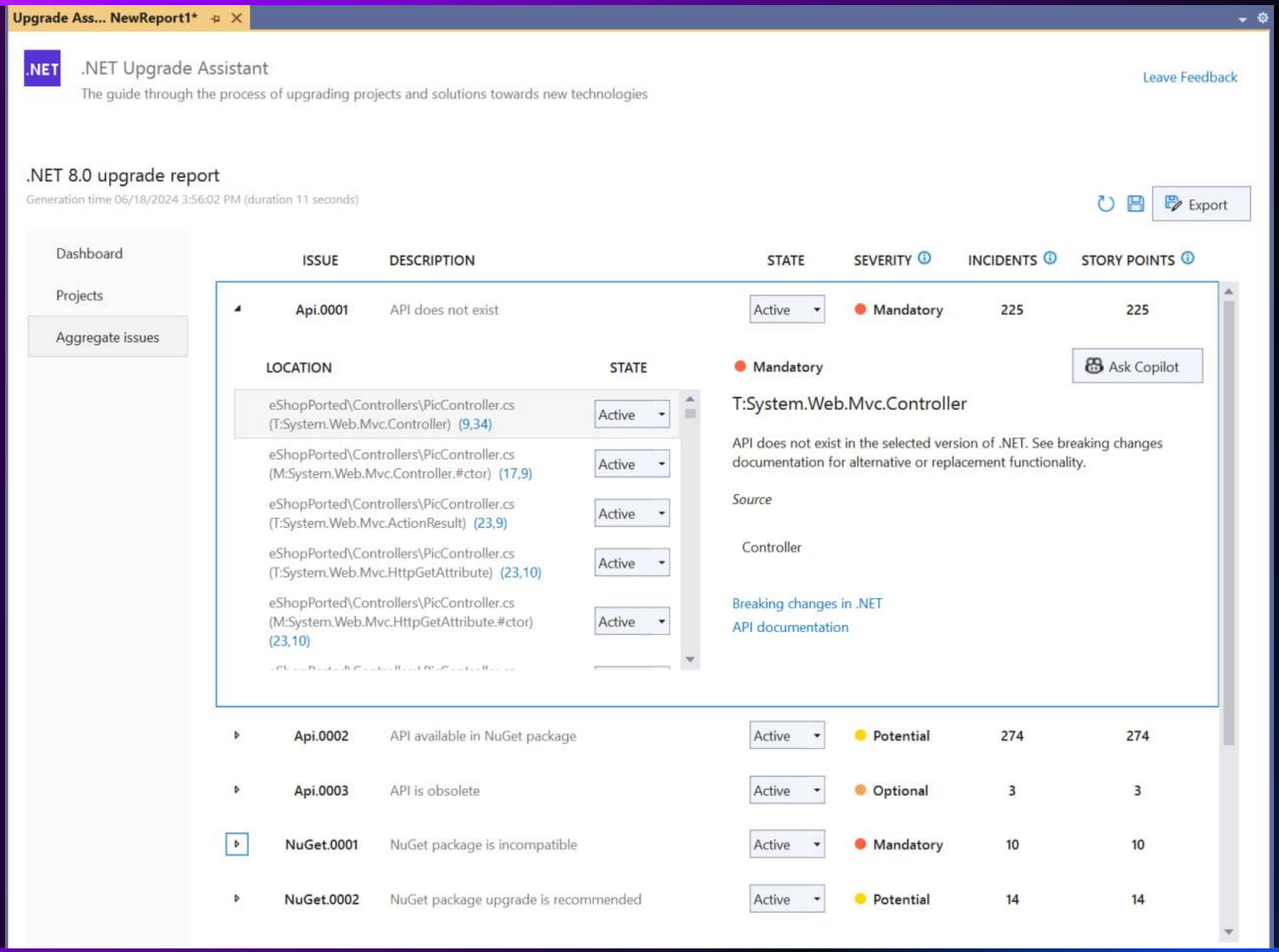
.NET Разработчик

@NetDeveloperDiary

Installing .NET 9 (3/160) ...



Картинка: https://dotnet.microsoft.com/blob-assets/images/illustrations/release-schedule-dark.svg



Картинка: https://devblogs.microsoft.com/visualstudio/wp-content/uploads/sites/4/2024/06/Issues-example.png

СТРАТЕГИЯ

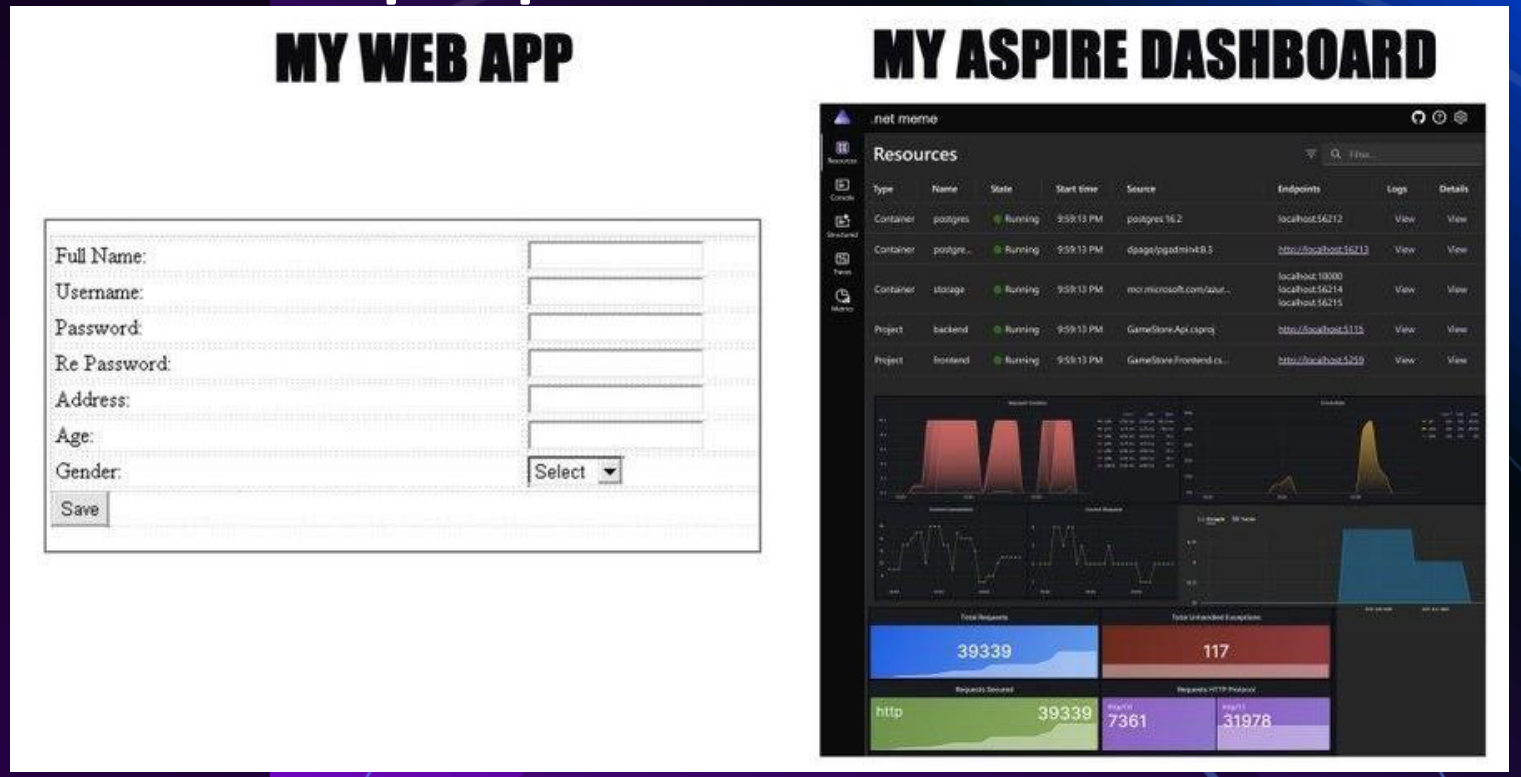
СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

- .NET Aspire



СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

- .NET Aspire
- Native AOT и DATAS

СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

- .NET Aspire
- Native AOT и DATAS
- Visual Studio и VS Code

СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

- .NET Aspire
- Native AOT и DATAS
- Visual Studio и VS Code

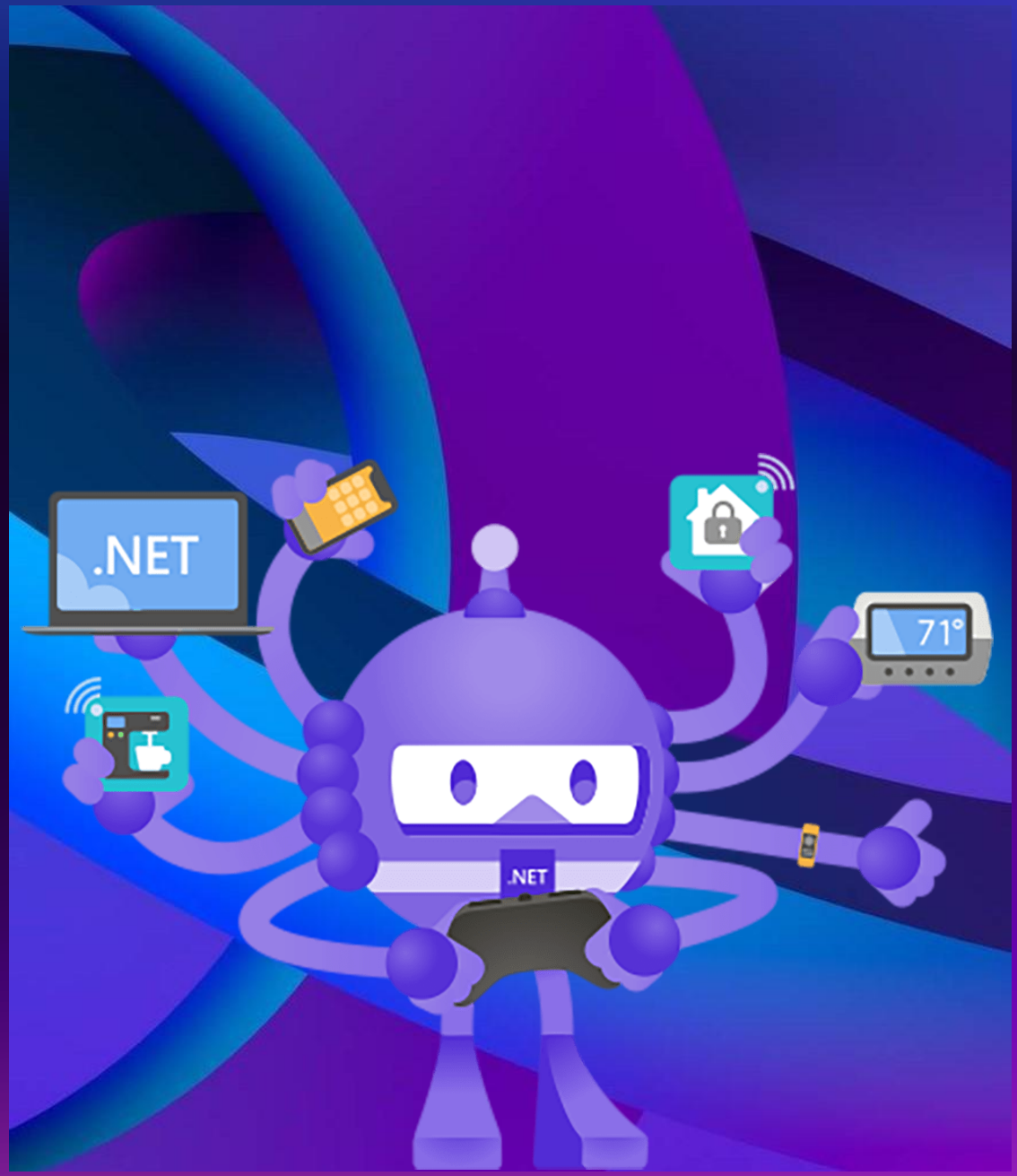
2. AI

1. C# и .NET

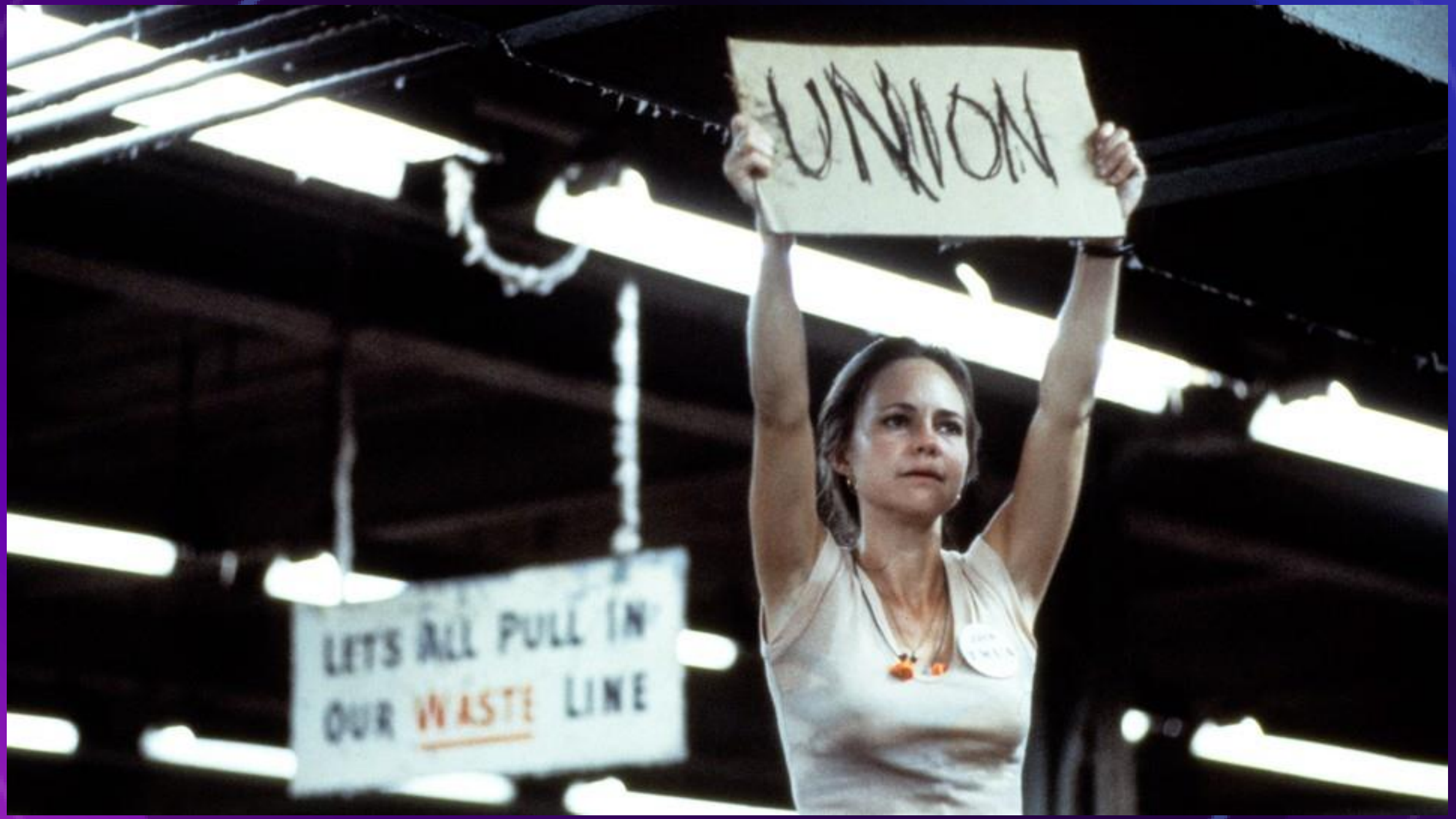
2. ASP.NET Core и Blazor

3. EF Core

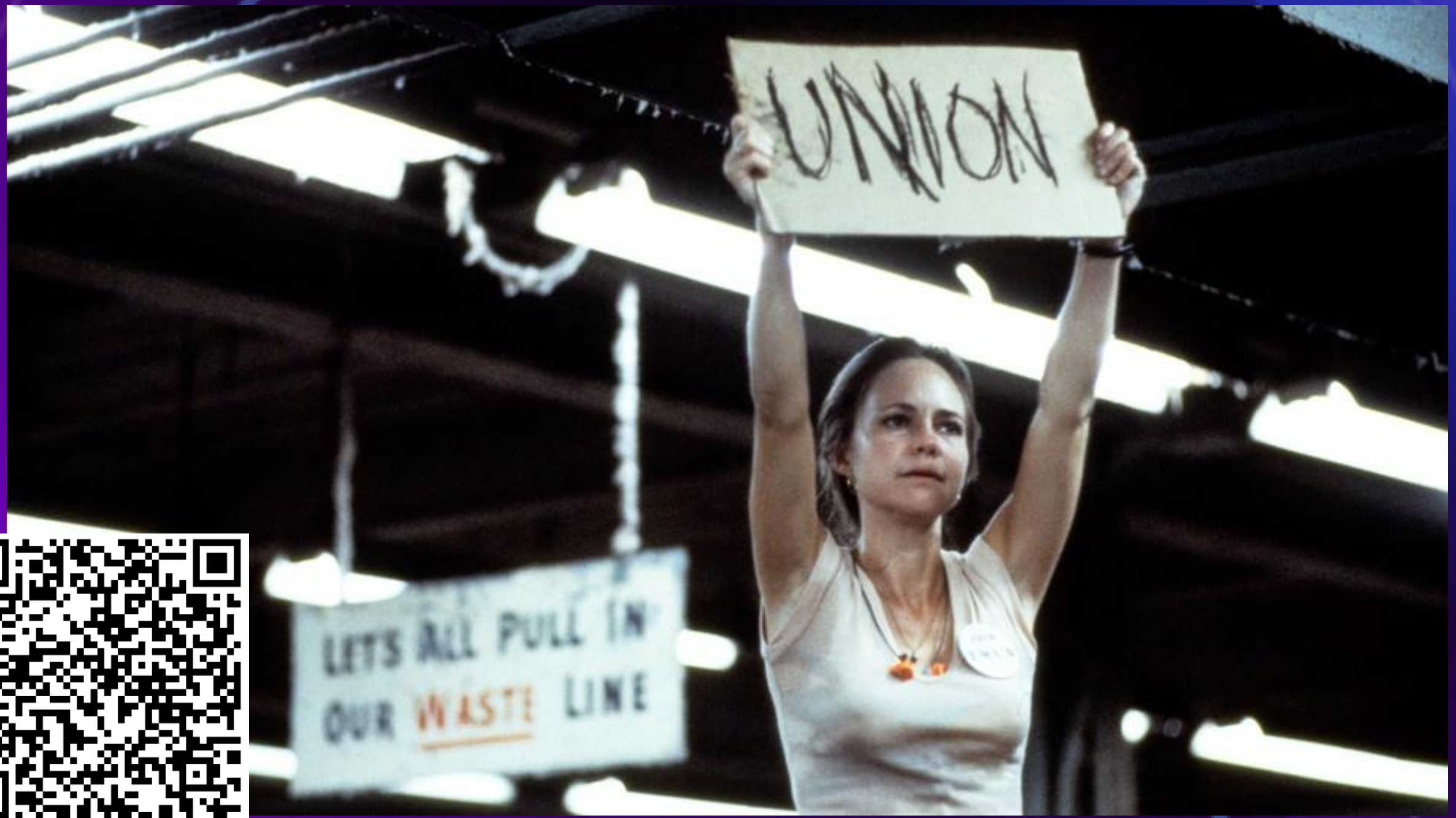
4. Десктоп



Картинка: https://i0.wp.com/puresourcecode.com/wp-content/uploads/2021/11/dotnet-bot-maui-cross-platform-development.png/



Картинка: https://www.theatlantic.com/entertainment/archive/2019/03/norma-rae-40th-anniversary-racial-solidarity-unions-labor-movement/583924/



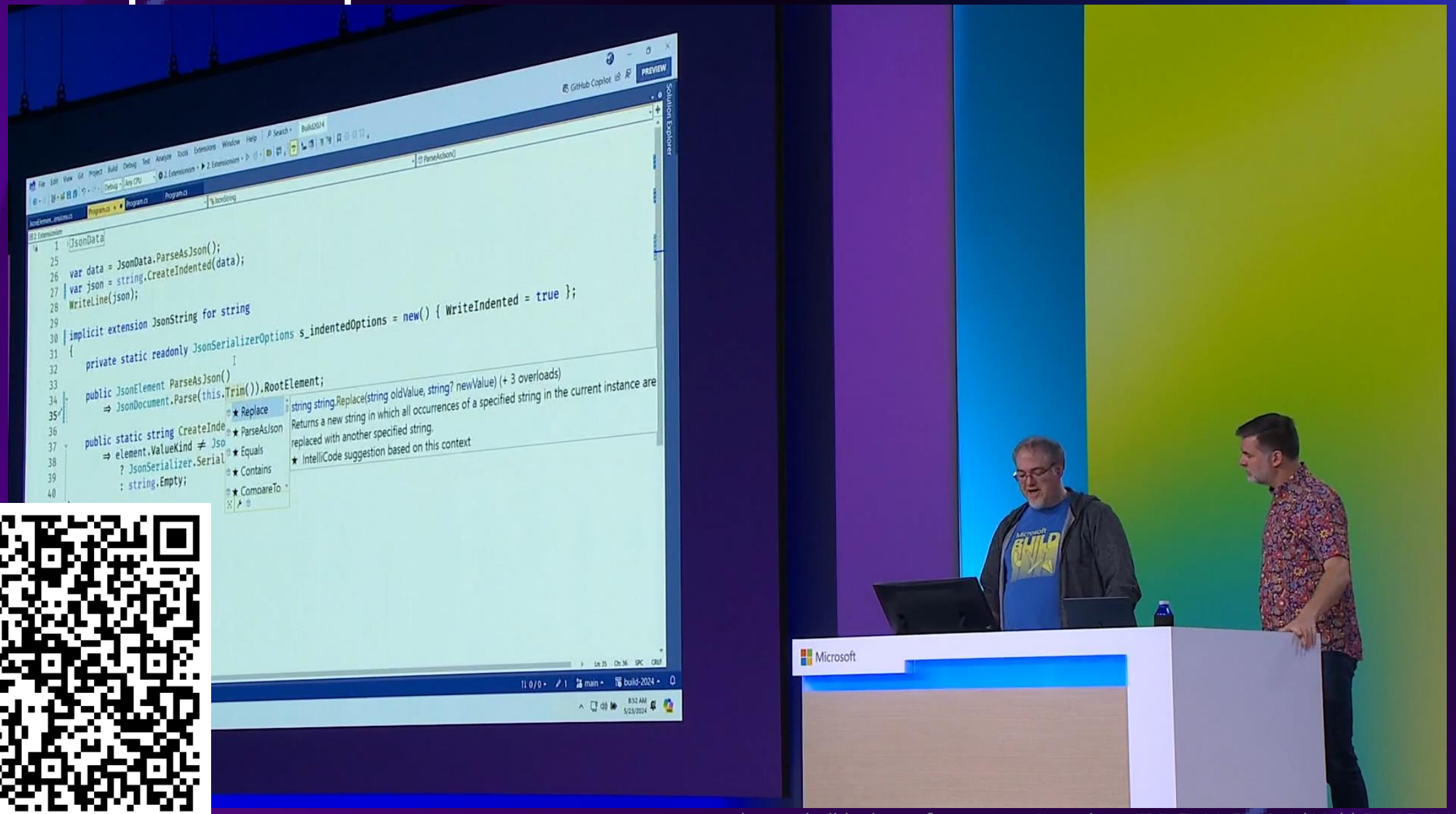
Картинка: https://www.theatlantic.com/entertainment/archive/2019/03/norma-rae-40th-anniversary-racial-solidarity-unions-labor-movement/583924/

C# И .NET

1. Полуавтоматические свойства
2. Улучшения SearchValues
3. Коллекции в params
4. Обработка асинхронных задач
5. Изменения ref-переменных
6. Regex.EnumerateSplits
7. Доступ к коллекциям с помощью Span'ов
8. Флаги функций
9. Новые методы LINQ
10. Новый тип блокировки
11. UUID версии 7
12. Парсинг имени типа

Типы-расширения

Типы-расширения



Картинка: https://build.microsoft.com/en-US/sessions/689e5104-72e9-4d02-bb52-77676d1ec5bc

Типы-расширения

Картинка: South Park

Полуавтоматические свойства

Полуавтоматические свойства

```
public string Name { get; set; }
```

Полуавтоматические свойства

```
public string Name { get; set; }
```

```
private string name;

public string Name
{
   get => name;
   set => name = value;
}
```

C#12

Полуавтоматические свойства

```
public string Name { get; set; }
```

```
private string name;

public string Name
{
   get => name;
   set => name = value.Trim();
}
```

C#12

Полуавтоматические свойства

```
public string Name { get; set; }
```

```
private string name;

public string Name
{
  get => name;
  set => name = value.Trim();
}
```

C#12

```
public string Name
{
  get;
  set => field = value.Trim();
}
```

C#13+

Полуавтоматические свойства

```
private string field;

public string Field
{
   get => field;
   set => field = value.Trim();
}
```

Полуавтоматические свойства

```
private string field;

public string Field
{
  get => field;
  set => field = value.Trim();
}
```

```
private string field;

public string Field
{
  get => @field;
  set => @field = value.Trim();
}
```

Улучшения SearchValues

Улучшения SearchValues

```
var vowels = SearchValues.Create(new[] { 'a', 'e', 'i', 'o', 'u' });
```

Улучшения SearchValues

```
var vowels = SearchValues.Create(new[] { 'a', 'e', 'i', 'o', 'u' });

ReadOnlySpan<char> text = "Hello, World!";
```

Улучшения SearchValues

```
var vowels = SearchValues.Create(new[] { 'a', 'e', 'i', 'o', 'u' });

ReadOnlySpan<char> text = "Hello, World!";

Console.WriteLine(text.ContainsAny(vowels));
```

```
True
```

Улучшения SearchValues

```
var sorts = SearchValues.Create(
  ["Bubble", "Gnome", "Bogosort"],
  StringComparison.OrdinalIgnoreCase);
```

Улучшения SearchValues

```
var sorts = SearchValues.Create(
  ["Bubble", "Gnome", "Bogosort"],
  StringComparison.OrdinalIgnoreCase);


var text = "I think the bubble sort would be the wrong way to go.";
```

Улучшения SearchValues

```
var sorts = SearchValues.Create(
  ["Bubble", "Gnome", "Bogosort"],
  StringComparison.OrdinalIgnoreCase);
```

```
var text = "
```


```
";
```

Улучшения SearchValues

```
var sorts = SearchValues.Create(
  ["Bubble", "Gnome", "Bogosort"],
  StringComparison.OrdinalIgnoreCase);

var text = "
```


```
";

Console.WriteLine(text.AsSpan().IndexOfAny(sorts));
```

```
3141592
```

Улучшения SearchValues

```
var sorts = SearchValues.Create(
  ["Bubble", "Gnome", "Bogosort"],
  StringComparison.OrdinalIgnoreCase);

var text = "
```


```
";

Console.WriteLine(text.AsSpan().IndexOfAny(sorts));
```

```
3141592
```

Коллекции в params

Коллекции в params

```
void PrintNums(params int[] nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}

PrintNums(1, 2, 3);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}

PrintNums(1, 2, 3);
PrintNums([1, 2, 3]);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}

PrintNums(1, 2, 3);
PrintNums([1, 2, 3]);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}

PrintNums(1, 2, 3);
PrintNums([1, 2, 3]);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}

List<int> list = [1, 2, 3];
PrintNums(list);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
  foreach (var n in nums)
    Console.WriteLine(n);
}

List<int> list = [1, 2, 3];
PrintNums(list);
```

Обработка задач по мере их завершения

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
  var wait = Random.Shared.Next(500, 5000);
  await Task.Delay(wait);
  return order;
}
```

Обработка задач по мере их завершения

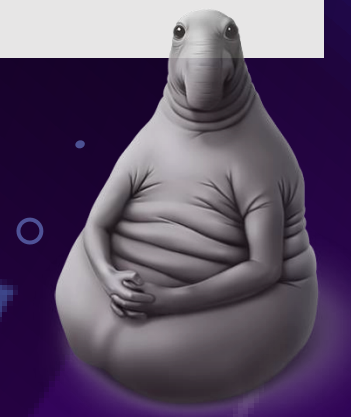
```
async Task<int> Calculate(int order)
{
  var wait = Random.Shared.Next(500, 5000);
  await Task.Delay(wait);
  return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);
```

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
  var wait = Random.Shared.Next(500, 5000);
  await Task.Delay(wait);
  return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

var results = await Task.WhenAll(tasks);
foreach (var r in results)
  Console.Write($"{r} ");
```

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
  var wait = Random.Shared.Next(500, 5000);
  await Task.Delay(wait);
  return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

var results = await Task.WhenAll(tasks);
foreach (var r in results)
  Console.Write($"{r} ");
```



Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
  var wait = Random.Shared.Next(500, 5000);
  await Task.Delay(wait);
  return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

var results = await Task.WhenAll(tasks);
foreach (var r in results)
  Console.Write($"{r} ");
```

```
1 2 3 4 5 6 7 8 9 10
```

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
  var wait = Random.Shared.Next(500, 5000);
  await Task.Delay(wait);
  return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

await foreach (var finished in Task.WhenEach(tasks))
{
  Console.WriteLine(await finished);
}
```

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
  var wait = Random.Shared.Next(500, 5000);
  await Task.Delay(wait);
  return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

await foreach (var finished in Task.WhenEach(tasks))
{
  Console.WriteLine(await finished);
}
```

```
4 7 1 5 2 10 6 3 8 9
```

Изменения ref-переменных

Изменения ref-переменных

1. ref и unsafe в async методах и итераторах

```
async Task MyMethodAsync()
{
  var result = await AsyncMethod1();
  var span = result.AsSpan();
  DoSomething(span);
  await AsnycMethod2();
}
```

Изменения ref-переменных

1. ref и unsafe в async методах и итераторах

```
async Task MyMethodAsync()
{
  var result = await AsyncMethod1();
  var span = result.AsSpan();
  DoSomething(span);
  await AsnycMethod2();
}
```

Изменения ref-переменных

1. ref и unsafe в async методах и итераторах

```
async Task MyMethodAsync()
{
  var result = await AsyncMethod1();
  var span = result.AsSpan();
  DoSomething(span);
  await AsnycMethod2();
}
```

2. Интерфейсы для ref-структур

Изменения ref-переменных

1. ref и unsafe в async методах и итераторах

```
async Task MyMethodAsync()
{
  var result = await AsyncMethod1();
  var span = result.AsSpan();
  DoSomething(span);
  await AsnycMethod2();
}
```

2. Интерфейсы для ref-структур

3. Ограничение allows ref struct

Regex.EnumerateSplits

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine(s);
}
```

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine(s);
}
```

Regex.EnumerateSplits

```
ReadOnlySpan<char> input = "Привет, DotNext! Как дела?";

foreach (Range r in Regex.EnumerateSplits(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine($"{input[r]}");
}
```

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine(s);
}
```

Regex.EnumerateSplits

```
ReadOnlySpan<char> input = "Привет, DotNext! Как дела?";

foreach (Range r in Regex.EnumerateSplits(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine($"{input[r]}");
}
```

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine(s);
}
```

Regex.EnumerateSplits

```
ReadOnlySpan<char> input = "Привет, DotNext! Как дела?";

foreach (Range r in Regex.EnumerateSplits(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine($"{input[r]}");
}
```

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine(s);
}
```

Regex.EnumerateSplits

```
ReadOnlySpan<char> input = "Привет, DotNext! Как дела?";

foreach (Range r in Regex.EnumerateSplits(input, "[аоуиэыеёюя]"))
{
  Console.WriteLine($"{input[r]}");
}
```

Доступ к коллекциям с помощью Span’ов

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
  Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

  Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
    counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

  foreach (Range r in Regex.EnumerateSplits(input, @"\b\W+\b"))
  {
    ReadOnlySpan<char> word = input[r];
    lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
  }

  return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
  Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

  Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
    counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

  foreach (Range r in Regex.EnumerateSplits(input, @"\b\W+\b"))
  {
    ReadOnlySpan<char> word = input[r];
    lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
  }

  return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
  Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

  Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
    counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

  foreach (Range r in Regex.EnumerateSplits(input, @"\b\W+\b"))
  {
    ReadOnlySpan<char> word = input[r];
    lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
  }

  return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
  Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

  Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
    counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

  foreach (Range r in Regex.EnumerateSplits(input, @"\b\W+\b"))
  {
    ReadOnlySpan<char> word = input[r];
    lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
  }

  return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
  Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

  Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
    counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

  foreach (Range r in Regex.EnumerateSplits(input, @"\b\W+\b"))
  {
    ReadOnlySpan<char> word = input[r];
    lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
  }

  return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
  Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

  Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
    counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

  foreach (Range r in Regex.EnumerateSplits(input, @"\b\W+\b"))
  {
    ReadOnlySpan<char> word = input[r];
    lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
  }

  return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
  Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

  Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
    counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

  foreach (Range r in Regex.EnumerateSplits(input, @"\b\W+\b"))
  {
    ReadOnlySpan<char> word = input[r];
    lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
  }

  return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
  Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

  Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
    counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

  foreach (Range r in Regex.EnumerateSplits(input, @"\b\W+\b"))
  {
    ReadOnlySpan<char> word = input[r];
    lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
  }

  return counts;
}
```

```
var text = "
```


```
";
foreach (var word in CountWords(text))
  Console.WriteLine(word);
```

Флаги функций с поддержкой тримминга

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
  [FeatureSwitchDefinition("Feature.IsSupported")]
  internal static bool IsSupported =>
    AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
      ? isEnabled
      : true;

  internal static void Implementation() => …;
}
```

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
  [FeatureSwitchDefinition("Feature.IsSupported")]
  internal static bool IsSupported =>
    AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
      ? isEnabled
      : true;

  internal static void Implementation() => …;
}
```

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
  [FeatureSwitchDefinition("Feature.IsSupported")]
  internal static bool IsSupported =>
    AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
      ? isEnabled
      : true;

  internal static void Implementation() => …;
}
```

```
 <ItemGroup>
    <RuntimeHostConfigurationOption Include="Feature.IsSupported"
       Value="false" Trim="true" />
  </ItemGroup>
```

MyApp.csproj

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
  [FeatureSwitchDefinition("Feature.IsSupported")]
  internal static bool IsSupported =>
    AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
      ? isEnabled
      : true;

  internal static void Implementation() => …;
}
```

```
<ItemGroup>
  <RuntimeHostConfigurationOption Include="Feature.IsSupported"
    Value="false" Trim="true" />
</ItemGroup>
```

MyApp.csproj

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
  [FeatureSwitchDefinition("Feature.IsSupported")]
  internal static bool IsSupported =>
    AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
      ? isEnabled
      : true;

  internal static void Implementation() => …;
}
```

```
 <ItemGroup>
    <RuntimeHostConfigurationOption Include="Feature.IsSupported"
       Value="false" Trim="true" />
  </ItemGroup>
```

MyApp.csproj

```
if (Feature.IsSupported)
  Feature.Implementation();
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
  [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
  internal static bool IsDynamicSupported =>
    RuntimeFeature.IsDynamicCodeSupported;

  [RequiresDynamicCode("Feature requires dynamic code support.")]
  internal static void DynamicImplementation() => …; // Использует dynamic
}
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
  [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
  internal static bool IsDynamicSupported =>
    RuntimeFeature.IsDynamicCodeSupported;

  [RequiresDynamicCode("Feature requires dynamic code support.")]
  internal static void DynamicImplementation() => …; // Использует dynamic
}
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
  [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
  internal static bool IsDynamicSupported =>
    RuntimeFeature.IsDynamicCodeSupported;

  [RequiresDynamicCode("Feature requires dynamic code support.")]
  internal static void DynamicImplementation() => …; // Использует dynamic
}
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
  [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
  internal static bool IsDynamicSupported =>
    RuntimeFeature.IsDynamicCodeSupported;

  [RequiresDynamicCode("Feature requires dynamic code support.")]
  internal static void DynamicImplementation() => …; // Использует dynamic
}
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
  [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
  internal static bool IsDynamicSupported =>
    RuntimeFeature.IsDynamicCodeSupported;

  [RequiresDynamicCode("Feature requires dynamic code support.")]
  internal static void DynamicImplementation() => …; // Использует dynamic
}
```

```
<PublishAot>true</PublishAot>
```

MyApp.csproj

```
if (Feature.IsDynamicSupported)
  Feature.DynamicImplementation();
```

Новые методы LINQ

Новые методы LINQ

1. CountBy

```
public record Person(string FirstName, string LastName);

List<Person> people = [
  new("Steve", "Jobs"),
  new("Elon", "Musk"),
  new("Steve", "Smith")
];

foreach (var p in people.CountBy(p => p.FirstName))
  Console.WriteLine($"{p.Value} людей с именем {p.Key}");
```

```
2 людей с именем Steve
1 людей с именем Elon
```

Новые методы LINQ

2. AggregateBy

```
public record Person(string Name, string Department, int Salary);

List<Person> people = [
  new("Jobs", "Sales", 180),
  new("Musk", "IT", 290),
  new("Smith", "IT", 120)
];

var aggregateBy = people.AggregateBy(
  p => p.Department, x => 0, (x, y) => x + y.Salary);

foreach (var agg in aggregateBy)
  Console.WriteLine($"ФОТ отдела {agg.Key}: {agg.Value}");
```

```
ФОТ отдела Sales: 180
ФОТ отдела IT: 410
```

Новые методы LINQ

3. Index

```
public record Person(string FirstName, string LastName);

List<Person> people = [
  new("Steve", "Jobs"),
  new("Elon", "Musk"),
  new("Steve", "Smith")
];

foreach (var (index, item) in people.Index())
  Console.WriteLine($"№{index}: {item}");
```

```
№1: Steve Jobs
№2: Elon Musk
№3: Steve Smith
```

Новый тип блокировки

Новый тип блокировки

```
object _lock = new();
void DoSomething()
{
  lock (_lock)
  {
    // Что-то делаем
  }
}
```

C# 12

Новый тип блокировки

```
object _lock = new();
void DoSomething()
{
  lock (_lock)
  {
    // Что-то делаем
  }
}
```

C# 12

```
Lock _lock = new();
void DoSomething()
{
  lock (_lock)
  {
    // Что-то делаем
  }
}
```

C# 13+

Новый тип блокировки

```
object _lock = new();
void DoSomething()
{
  lock (_lock)
  {
    // Что-то делаем
  }
}
```

C# 12

```
Lock _lock = new();
void DoSomething()
{
  lock (_lock)
  {
    // Что-то делаем
  }
}
```

C# 13+

Новый тип блокировки

```
object _lock = new();
void DoSomething()
{
  lock (_lock)
  {
    // Что-то делаем
  }
}
```

C# 12

```
Lock _lock = new();
void DoSomething()
{
  lock (_lock)
  {
    // Что-то делаем
  }
}
```

C# 13+

UUID версии 7

UUID версии 7

```
for (int i = 0; i < 5; i++)
  Console.WriteLine(Guid.NewGuid());
```

```
f529a20a-6d23-4568-9d3e-01b18f1ff40e
b2c9f7ee-fcf4-4dbf-8194-9f0354a05d79
528419ee-ab84-451d-b568-4bd26fca27be
720c11de-e9fa-4959-bb1f-6bf1c6d3bea6
d581b627-eb46-41d7-876d-58c0d9af78d7
```

UUID версии 7

```
for (int i = 0; i < 5; i++)
  Console.WriteLine(Guid.NewGuid());
```

```
f529a20a-6d23-4568-9d3e-01b18f1ff40e
b2c9f7ee-fcf4-4dbf-8194-9f0354a05d79
528419ee-ab84-451d-b568-4bd26fca27be
720c11de-e9fa-4959-bb1f-6bf1c6d3bea6
d581b627-eb46-41d7-876d-58c0d9af78d7
```

```
for (int i = 0; i < 5; i++)
  Console.WriteLine(Guid.CreateVersion7());
```

```
01918fe2-56f1-7131-8e68-f1d37b8dc2ce
01918fe2-56f1-7e85-825c-217376c27fa8
01918fe2-56f1-78ea-a008-606df446098e
01918fe2-56f1-7023-bdfe-9fe6a8f49b9c
01918fe2-56f1-71fc-80fb-791913750271
```

UUID версии 7

```
for (int i = 0; i < 5; i++)
  Console.WriteLine(Guid.NewGuid());
```

```
f529a20a-6d23-4568-9d3e-01b18f1ff40e
b2c9f7ee-fcf4-4dbf-8194-9f0354a05d79
528419ee-ab84-451d-b568-4bd26fca27be
720c11de-e9fa-4959-bb1f-6bf1c6d3bea6
d581b627-eb46-41d7-876d-58c0d9af78d7
```

```
for (int i = 0; i < 5; i++)
  Console.WriteLine(Guid.CreateVersion7());
```

```
01918fe2-56f1-7131-8e68-f1d37b8dc2ce
01918fe2-56f1-7e85-825c-217376c27fa8
01918fe2-56f1-78ea-a008-606df446098e
01918fe2-56f1-7023-bdfe-9fe6a8f49b9c
01918fe2-56f1-71fc-80fb-791913750271
```

```
var uuidTest = Guid.CreateVersion7(timeProvider.GetUtcNow());
```

Парсинг имени типа

```
using System.Reflection.Metadata;

class SerializationBinder
{
  public Type GetType(ReadOnlySpan<char> input)
  {
    if (!TypeName.TryParse(input, out TypeName parsed))
    {
      …
    }
    if (parsed.IsSimple // не дженерик, не указатель, не массив
        && parsed.AssemblyName.Name == "TrustedAssembly")
    {
      return Type.GetType(parsed.AssemblyQualifiedName, throwOnError: true);
    }
    …
  }
}
```

Парсинг имени типа

```
using System.Reflection.Metadata;

class SerializationBinder
{
  public Type GetType(ReadOnlySpan<char> input)
  {
    if (!TypeName.TryParse(input, out TypeName parsed))
    {
      …
    }
    if (parsed.IsSimple // не дженерик, не указатель, не массив
        && parsed.AssemblyName.Name == "TrustedAssembly")
    {
      return Type.GetType(parsed.AssemblyQualifiedName, throwOnError: true);
    }
    …
  }
}
```

Новые типы и API

Base64Url

```
ReadOnlySpan<byte> bytes = …;
string encoded = Base64Url.EncodeToString(bytes);
```

Новые типы и API

Base64Url

```
ReadOnlySpan<byte> bytes = …;
string encoded = Base64Url.EncodeToString(bytes);
```

OrderedDictionary<TKey, TValue>

```
OrderedDictionary<string, int> d = new() { { "a", 1 }, { "b", 2 } };
d.RemoveAt(1);
d.Insert(0, "d", 4);
```

```
[d, 4] [a, 1]
```

Новые типы и API

Base64Url

```
ReadOnlySpan<byte> bytes = …;
string encoded = Base64Url.EncodeToString(bytes);
```

OrderedDictionary<TKey, TValue>

```
OrderedDictionary<string, int> d = new() { { "a", 1 }, { "b", 2 } };
d.RemoveAt(1);
d.Insert(0, "d", 4);
```

```
[d, 4] [a, 1]
```

ReadOnlySet<T>

```
var set = new HashSet<int> { 1, 2, 3 };
var readonlySet = new ReadOnlySet<int>(set);
```

Новые типы и API

Base64Url

```
ReadOnlySpan<byte> bytes = …;
string encoded = Base64Url.EncodeToString(bytes);
```

OrderedDictionary<TKey, TValue>

```
OrderedDictionary<string, int> d = new() { { "a", 1 }, { "b", 2 } };
d.RemoveAt(1);
d.Insert(0, "d", 4);
```

```
[d, 4] [a, 1]
```

ReadOnlySet<T>

```
var set = new HashSet<int> { 1, 2, 3 };
var readonlySet = new ReadOnlySet<int>(set);
```

TensorPrimitives и Tensor<T>

```
using System.Numerics.Tensors;
ReadOnlySpan<double> vector1 = [1, 2, 3];
ReadOnlySpan<double> vector2 = [4, 5, 6];
Console.WriteLine(TensorPrimitives.CosineSimilarity(vector1, vector2));
```

```
0.9746318461970762
```

~~BinaryFormatter~~

System.Runtime.Serialization.Formatters (8.1.0.0) | System.Runtime.Serialization.Formatters.Binary.BinaryFormatter | Deserialize(Stream serializationSt

```
1  // Licensed to the .NET Foundation under one or more agreements.
2  // The .NET Foundation licenses this file to you under the MIT license.
3
4  using ...
6
7  namespace System.Runtime.Serialization.Formatters.Binary
8  {
9      public sealed partial class BinaryFormatter : IFormatter
10     {
11         [RequiresDynamicCode(message: IFormatter.RequiresDynamicCodeMessage)]
12         [RequiresUnreferencedCode(message: IFormatter.RequiresUnreferencedCodeMessage)]
13         public object Deserialize(Stream serializationStream)
14             => throw new PlatformNotSupportedException(SR.BinaryFormatter_Removed);
15
16         [RequiresUnreferencedCode(message: IFormatter.RequiresUnreferencedCodeMessage)]
17         public void Serialize(Stream serializationStream, object graph)
18             => throw new PlatformNotSupportedException(SR.BinaryFormatter_Removed);
19     }
20 }
21
```

Installing .NET 9 (102/160) ...

ASP.NET CORE И BLAZOR

1. Гибридный кэш
2. Умные компоненты
3. Новый вид сжатия статических ресурсов
4. OpenAPI
5. Blazor

Гибридный кэш

Гибридный кэш

Кэш в памяти

```
builder.Services.AddMemoryCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    async entry => await _repo.GetDataAsync(name, id, ct);
  );
}
```

Гибридный кэш

Кэш в памяти

```
builder.Services.AddMemoryCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    async entry => await _repo.GetDataAsync(name, id, ct);
  );
}
```

Распределённый кэш

```
builder.Services.AddStackExchangeRedisCache();

…
```

Гибридный кэш

Кэш в памяти

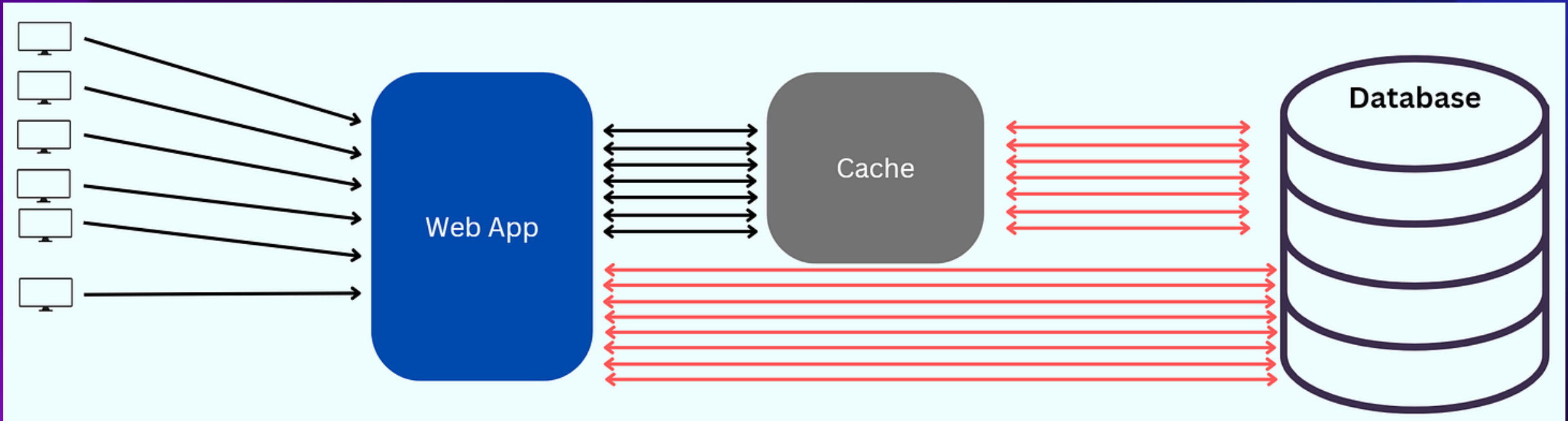
```
builder.Services.AddMemoryCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    async entry => await _repo.GetDataAsync(name, id, ct);
  );
}
```

Распределённый кэш

```
builder.Services.AddStackExchangeRedisCache();

…
```

Гибридный кэш

«Паника в кэше»



Картинка: https://zahere.com/cache-stampede-a-problem-the-industry-fights-every-day

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    async cancel => await _repo.GetPostAsync(name, id, cancel),
    token: ct
  );
}
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    async cancel => await _repo.GetPostAsync(name, id, cancel),
    token: ct
  );
}
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    async cancel => await _repo.GetPostAsync(name, id, cancel),
    token: ct
  );
}
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    (name, id), // вся информация для получения данных
    static async (state, cancel) =>
      await _repo.GetPostAsync(state.name, state.id, cancel),
    token: ct
  );
}
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    (name, id), // вся информация для получения данных
    static async (state, cancel) =>
      await _repo.GetPostAsync(state.name, state.id, cancel),
    token: ct
  );
}
```

```
[ImmutableObject(true)]
public sealed class SomeData { … }
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    (name, id), // вся информация для получения данных
    static async (state, cancel) =>
      await _repo.GetPostAsync(state.name, state.id, cancel),
    token: ct
  );
}
```

```
[ImmutableObject(true)]
public sealed class SomeData { … }
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    (name, id), // вся информация для получения данных
    static async (state, cancel) =>
      await _repo.GetPostAsync(state.name, state.id, cancel),
    token: ct
  );
}
```

```
[ImmutableObject(true)]
public sealed class SomeData { … }
```

Гибридный кэш

HybridCache

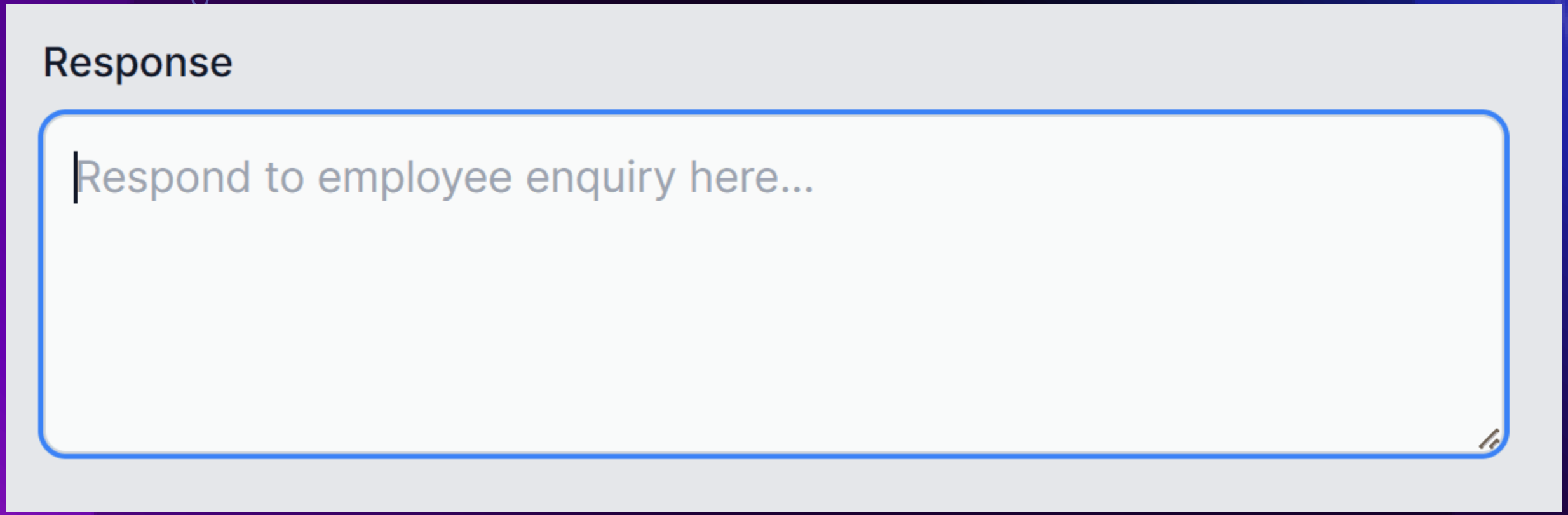
```
builder.Services.AddHybridCache().WithSerializer(…);
…
public async Task<SomeData> GetDataAsync(
  string name, int id, CancellationToken ct = default)
{
  return await _cache.GetOrCreateAsync(
    $"data:{name}:{id}", // уникальный ключ
    (name, id), // вся информация для получения данных
    static async (state, cancel) =>
      await _repo.GetPostAsync(state.name, state.id, cancel),
    token: ct
  );
}
```

```
[ImmutableObject(true)]
public sealed class SomeData { … }
```

.NET Smart Components

.NET Smart Components

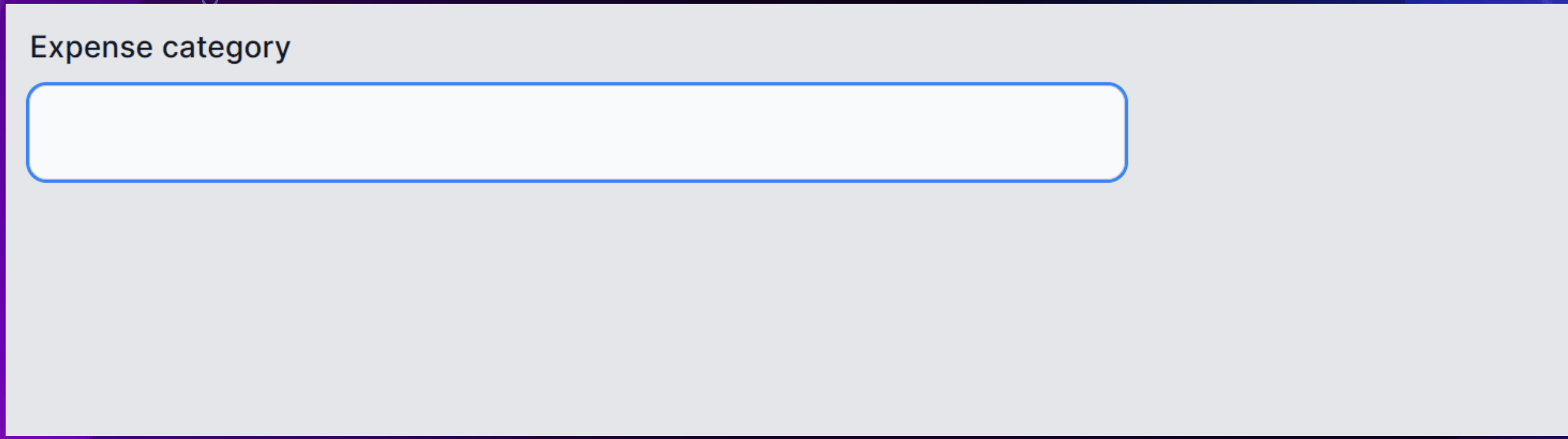
Умное поле для ввода



Картинка: https://visualstudiomagazine.com/Articles/2024/03/20/~/media/ECG/visualstudiomagazine/Images/2024/03/smart_textarea.ashx

.NET Smart Components

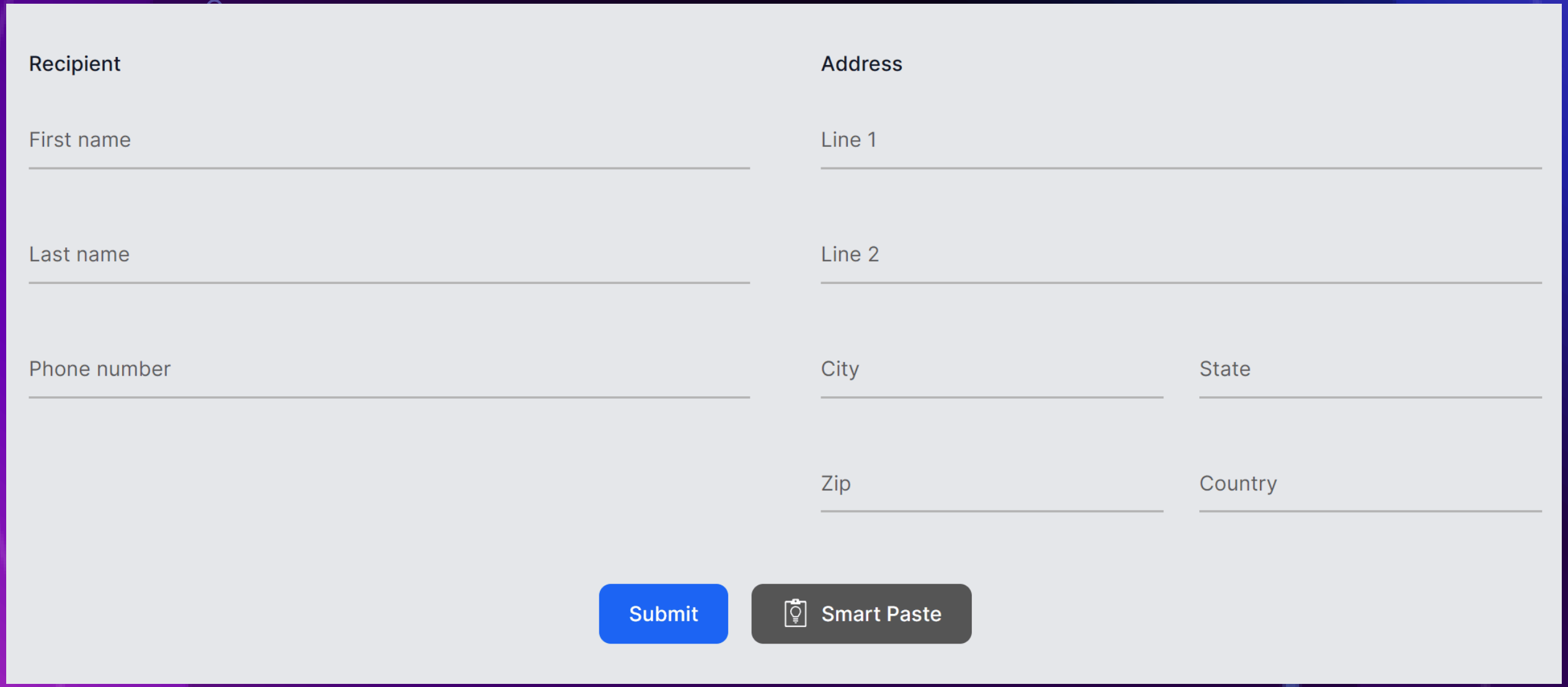
Умный выпадающий список



Картинка: https://visualstudiomagazine.com/Articles/2024/03/20/~/media/ECG/visualstudiomagazine/Images/2024/03/smart_combobox.ashx

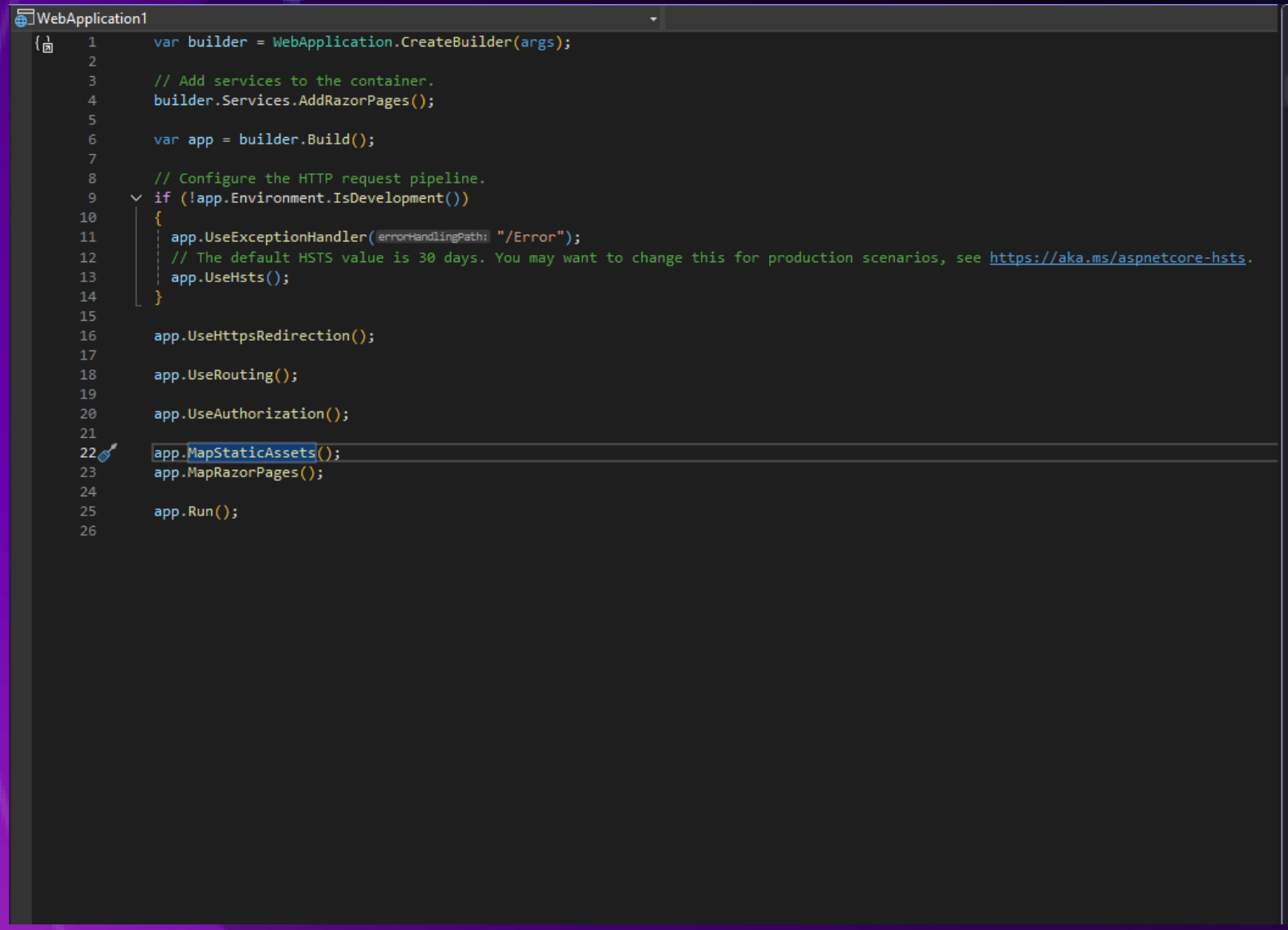
.NET Smart Components

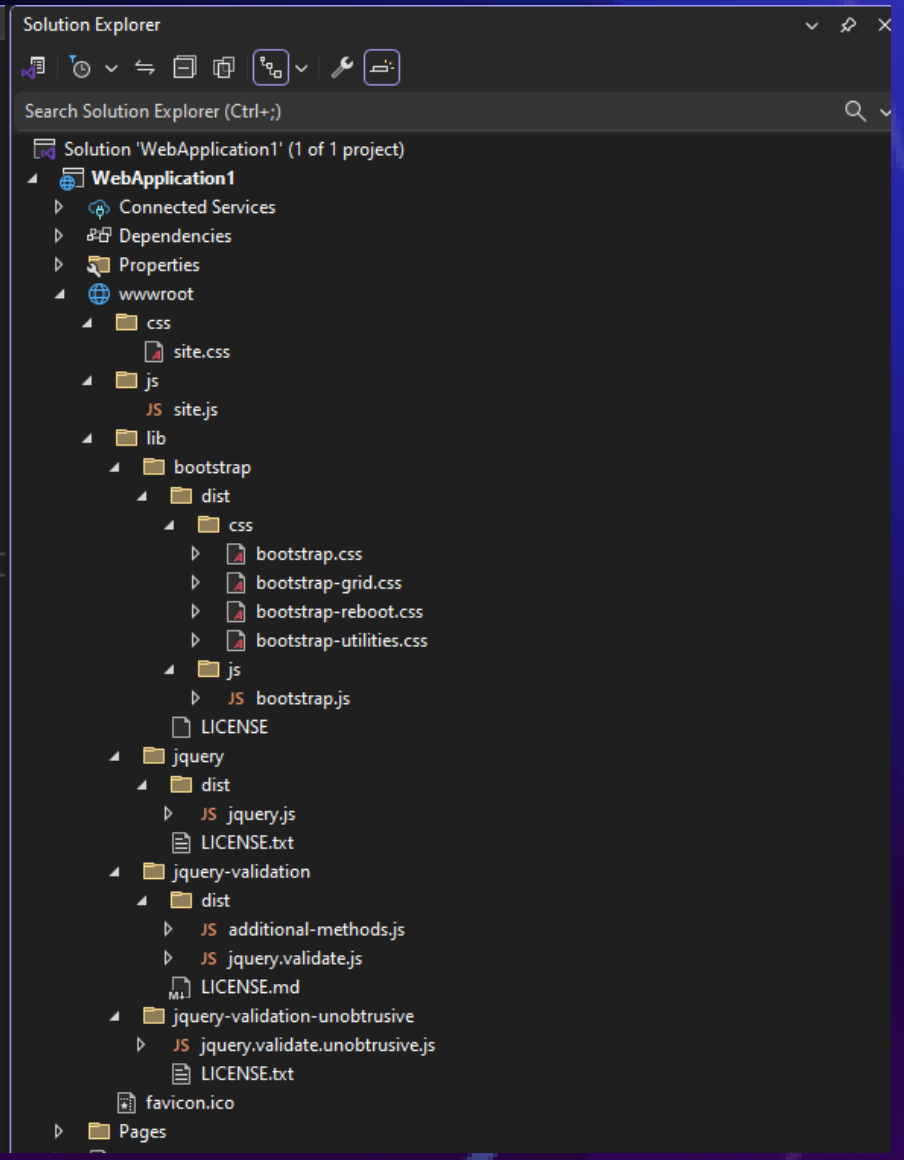
Умная вставка



Картинка: https://visualstudiomagazine.com/Articles/2024/03/20/~/media/ECG/visualstudiomagazine/Images/2024/03/smart_paste.ashx

MapStaticAssets





MapStaticAssets

Шаблон Razor Pages

File	Original	Compressed	% Reduction
bootstrap.min.css	163	17.5	89.26%
jquery.js	89.6	28	68.75%
bootstrap.min.js	78.5	20	74.52%
Total	331.1	65.5	80.20%

Fluent UI Blazor

File	Original	Compressed	% Reduction
fluent.js	384	73	80.99%
fluent.css	94	11	88.30%
Total	478	84	82.43%

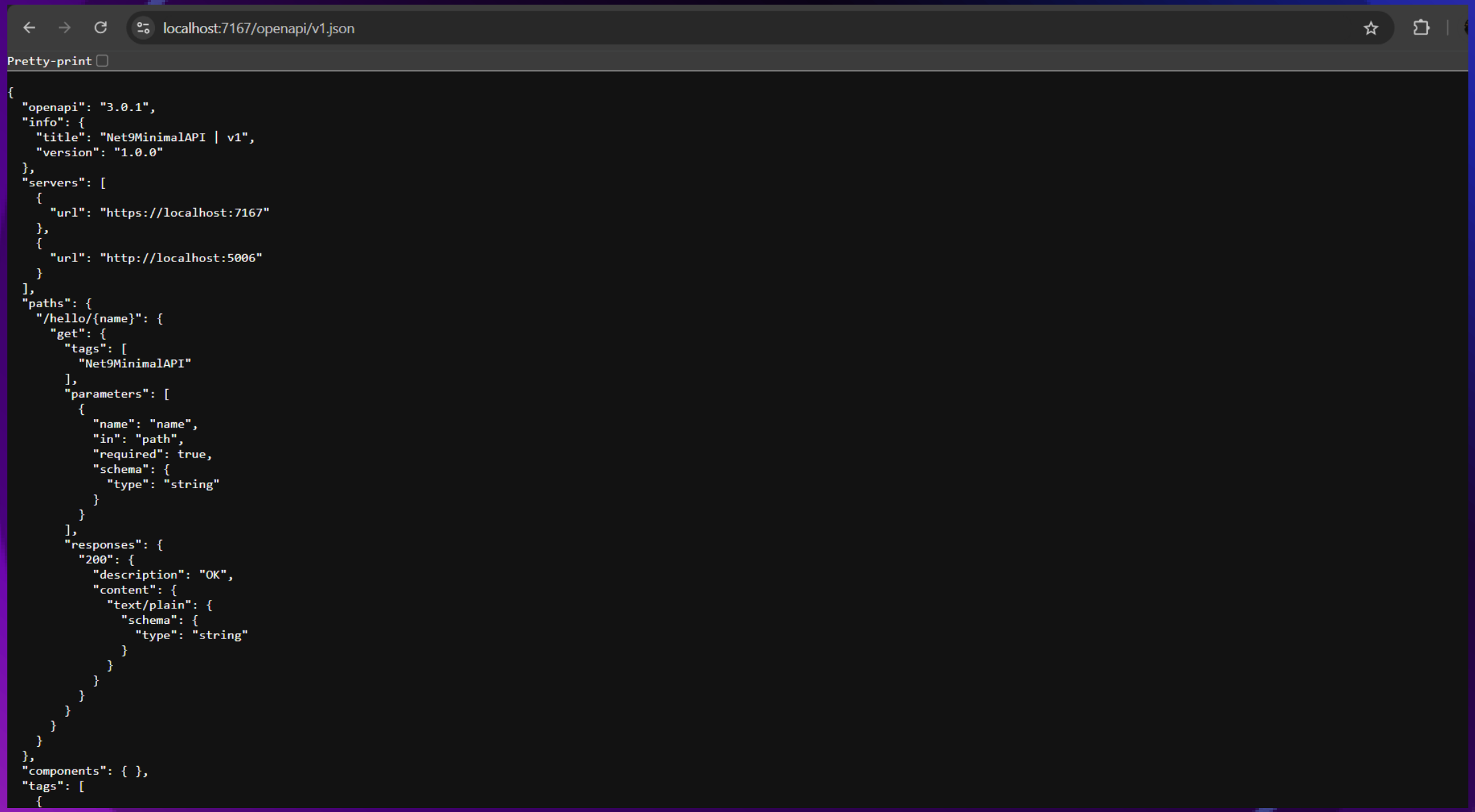
MudBlazor

File	Original	Compressed	% Reduction
MudBlazor.min.css	541	37.5	93.07%
MudBlazor.min.js	47.4	9.2	80.59%
Total	588.4	46.7	92.07%

OpenAPI

```
var builder = WebApplication.CreateBuilder();

builder.Services.AddOpenApi();

var app = builder.Build();

app.MapOpenApi();

app.MapGet("/hello/{name}", (string name) => $"Hello {name}"!);
app.Run();
```

OpenAPI


localhost:7167/openapi/v1.json
Pretty-print
{
"openapi": "3.0.1",
"info": {
"title": "Net9MinimalAPI | v1",
"version": "1.0.0"
},
"servers": [
{
"url": "https://localhost:7167"
},
{
"url": "http://localhost:5006"
}
],
"paths": {
"/hello/{name}": {
"get": {
"tags": [
"Net9MinimalAPI"
],
"parameters": [
{
"name": "name",
"in": "path",
"required": true,
"schema": {
"type": "string"
}
}
],
"responses": {
"200": {
"description": "OK",
"content": {
"text/plain": {
"schema": {
"type": "string"
}
}
}
}
}
}
}
},
"components": { },
"tags": [
{

Blazor

- Загрузки страниц и времени старта
- Пользовательский опыт
- Интерактивный рендеринг
- Логика восстановления соединения в Blazor Server
- Сжатие сообщений WebSocket
- Скаффолдинг моделей и Identity в Visual Studio
- Производительность и функциональность Hot Reload
- Отладка Blazor WebAssembly

EF CORE

EF CORE

1. AOT

EF CORE

1. AOT

```
> dotnet ef dbcontext optimize
```

EF CORE

1. AOT

```
> dotnet ef dbcontext optimize
```

```
protected override void
 OnConfiguring(DbContextOptionsBuilder options)
  => options
   .UseModel(CompiledModels.MyContextModel.Instance);
```

EF CORE

1. AOT

   с помощью Microsoft.EntityFrameworkCore.Tasks

```
<PropertyGroup>
  <EFOptimizeContext>true</EFOptimizeContext>
</PropertyGroup>
```

MyApp.csproj

EF CORE

1. AOT

   с помощью Microsoft.EntityFrameworkCore.Tasks

```
<PropertyGroup>
  <EFOptimizeContext>true</EFOptimizeContext>
  <EFStartupProject>..\App\App.csproj</EFStartupProject>
</PropertyGroup>
```

MyApp.csproj

EF CORE

1. AOT
2. Перевод LINQ в SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам

```
var groupByAddress = await context.Stores
    .GroupBy(b => b.Address)
    .Select(g => new { g.Key, Count = g.Count() })
    .ToListAsync();
```

LINQ

```
SELECT Address_City, Address_Line1, Address_Zip, COUNT(*) AS Count
FROM Stores
GROUP BY Address_City, Address_Line1, Address_Zip;
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант

```
var conf = await context.Conferences
 .Where(e => e.Title == "DotNext" && e.Year == year)
 .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = N'DotNext' AND Year = @__year_0
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант

```
var conf = await context.Conferences
 .Where(e => e.Title == "DotNext" && e.Year == year)
 .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = N'DotNext' AND Year = @__year_0
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант

```
var conf = await context.Conferences
 .Where(e => e.Title == "DotNext" && e.Year == year)
 .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = N'DotNext' AND Year = @__year_0
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант

```
var conf = await context.Conferences
 .Where(e => e.Title == "DotNext" && e.Year == EF.Constant(id))
 .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = N'DotNext' AND Year = 2024
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант

```
var conf = await context.Conferences
 .Where(e=>e.Title==EF.Parameter("DotNext") && e.Year == year)
 .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = @__p_0 AND Year = @__year_1
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант
   - Встраивание некоррелирующих подзапросов

```
var lectures = await context.Lectures
  .Where(l => l.Title.Contains(".NET"));

var results = lectures
 .Where(l => l.Year > 2020)
 .Select(p => new { Lecture = l, Total = lectures.Count() })
 .ToArray();
```

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант
   - Встраивание некоррелирующих подзапросов
   - Count > 0 -> EXISTS

```linq
var blogsWithPost = await context.Blogs
    .Where(b => b.Posts.Count > 0)
    .ToListAsync();
```

LINQ

```sql
SELECT * FROM Blogs b
WHERE EXISTS (
  SELECT 1 FROM Posts p WHERE b.Id = p.BlogId);
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант
   - Встраивание некоррелирующих подзапросов
   - Count > 0 -> EXISTS
   - Оптимизация OPENJSON WITH

EF CORE

1. AOT
2. Перевод LINQ в SQL
   - GroupBy по сложным типам
   - Обработка констант
   - Встраивание некоррелирующих подзапросов
   - Count > 0 -> EXISTS
   - Оптимизация OPENJSON WITH
   - Math.Max / Min -> GREATEST / LEAST

EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях

```
await context.Stores
  .ExecuteUpdateAsync(
    s => s.SetProperty(b => b.Address.Line1, newAddr.Line1)
          .SetProperty(b => b.Address.City, newAddr.City)
          .SetProperty(b => b.Address.Zip, newAddr.Zip));
```

EF8

```
await context.Stores
  .ExecuteUpdateAsync(
    s => s.SetProperty(b => b.Address, newAddr));
```

EF9+

EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях
4. Примитивные коллекции только для чтения

EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях
4. Примитивные коллекции только для чтения
5. Кэш последовательностей

EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях
4. Примитивные коллекции только для чтения
5. Кэш последовательностей
6. Fill-factor для ключей и индексов

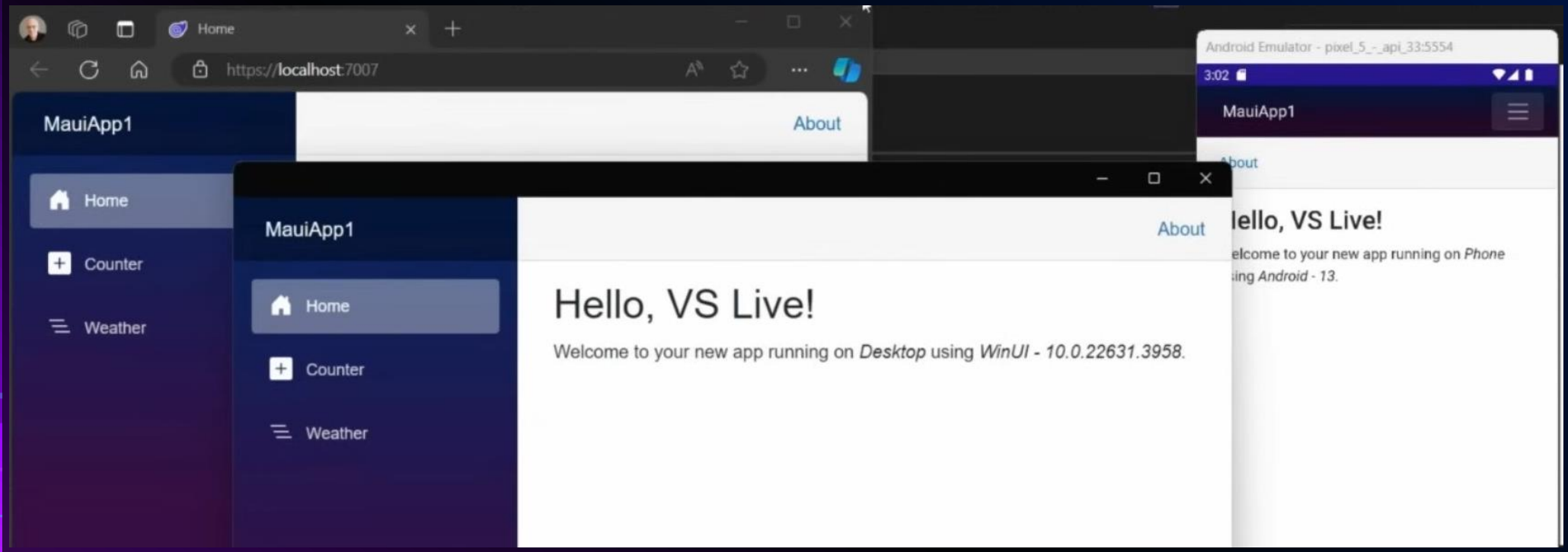
EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях
4. Примитивные коллекции только для чтения
5. Кэш последовательностей
6. Fill-factor для ключей и индексов
7. Azure Cosmos DB для NoSQL

MAUI

MAUI

1. Blazor Hybrid



Картинка: https://youtu.be/oOCWssf8TFw

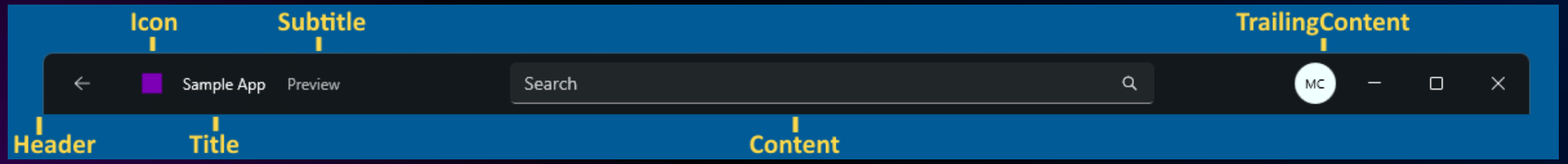
MAUI

1. Blazor Hybrid
2. Новые компоненты
   - HybridWebView

Картинка: https://devclass.com/wp-content/uploads/2024/08/hybridwebview-1024x763.jpg

MAUI

1. Blazor Hybrid
2. Новые компоненты
   - HybridWebView
   - TitleBar



Картинка: https://learn.microsoft.com/en-us/dotnet/maui/whats-new/media/dotnet-9/titlebar-overview.png

MAUI

1. Blazor Hybrid
2. Новые компоненты
3. Android 15, iOS 17.2 и macOS 14.2

MAUI

1. Blazor Hybrid
2. Новые компоненты
3. Android 15, iOS 17.2 и macOS 14.2
4. Asset packs

```
<ItemGroup>
  <AndroidAsset Update="Assets/MyLargeAsset.mp4"
    AssetPack="myassets" />
</ItemGroup>
```

MyApp.csproj

WINFORMS

WINFORMS

1. Поддержка тёмной темы

WINFORMS

1. Поддержка тёмной темы
2. Отрисовка компонентов и шрифтов

WINFORMS

1. Поддержка тёмной темы
2. Отрисовка компонентов и шрифтов
3. Поддержка асинхронности в UI

WINFORMS

1. Поддержка тёмной темы
2. Отрисовка компонентов и шрифтов
3. Поддержка асинхронности в UI
4. Поддержка AI компонентов

WINFORMS

1. Поддержка тёмной темы
2. Отрисовка компонентов и шрифтов
3. Поддержка асинхронности в UI
4. Поддержка AI компонентов
5. MVVM для байндинга данных
   с помощью CommunityToolkit.Mvvm

СПАСИБО

Сергей Бензенко

@SBenzenko

Телеграм канал «.NET Разработчик»

@NetDeveloperDiary