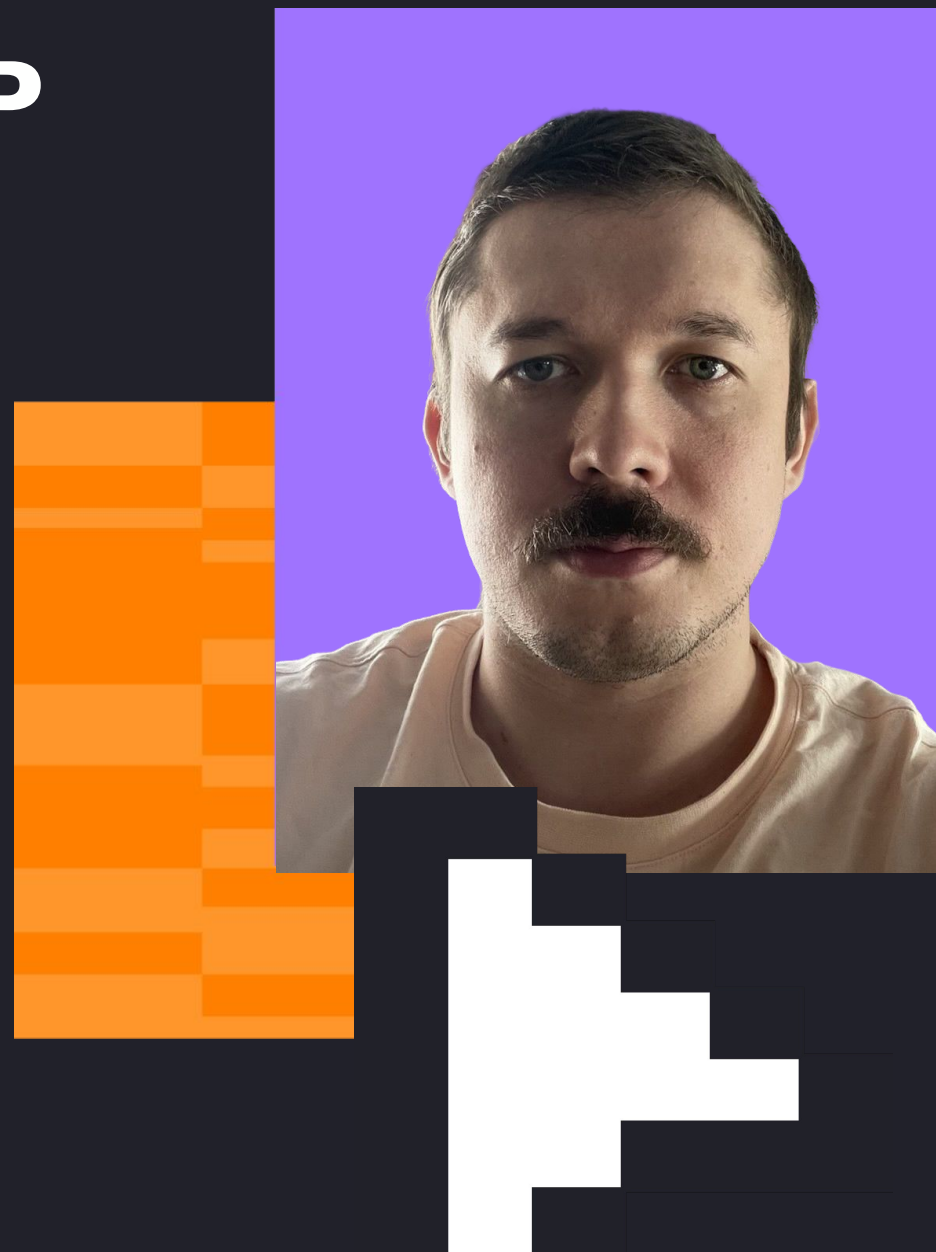


Сложно ли сделать свою IDE?

Шестаков Алексей

Ведущий
инженер-программист

Контур



Что мы делаем

Как делаем:

- Мы делаем IDE для C#

Делаем хорошо!

Зачем

- 2022
- Rider больше не купить
- JetBrains активно банит по IP и другими способами
- EAP использовать тоже нельзя

Читаем лицензию:

The **User shall comply ... sanctions**

Акт 1

Может мы не будем ничего писать?

- Писать код, конечно, никто не собирался
- OpenIDE ?
- VSCode ?
-



AlanMac95, DeviantArt 2017

**На рынке в тот момент
не было ничего готового**



АКТ 2

Пробуем собрать на базе vscode

IDE?



C#?

ЧТО МЫ ХОТИМ?

Тест-раннер

Nuget
эксплорер

Дебаггер

Интеллисенс

Солюшен
эксплорер



ЧТО МЫ ХОТИМ?

DotRush

Тест-раннер

Nuget
эксплорер

Дебаггер

Интеллисенс

Солюшен
эксплорер



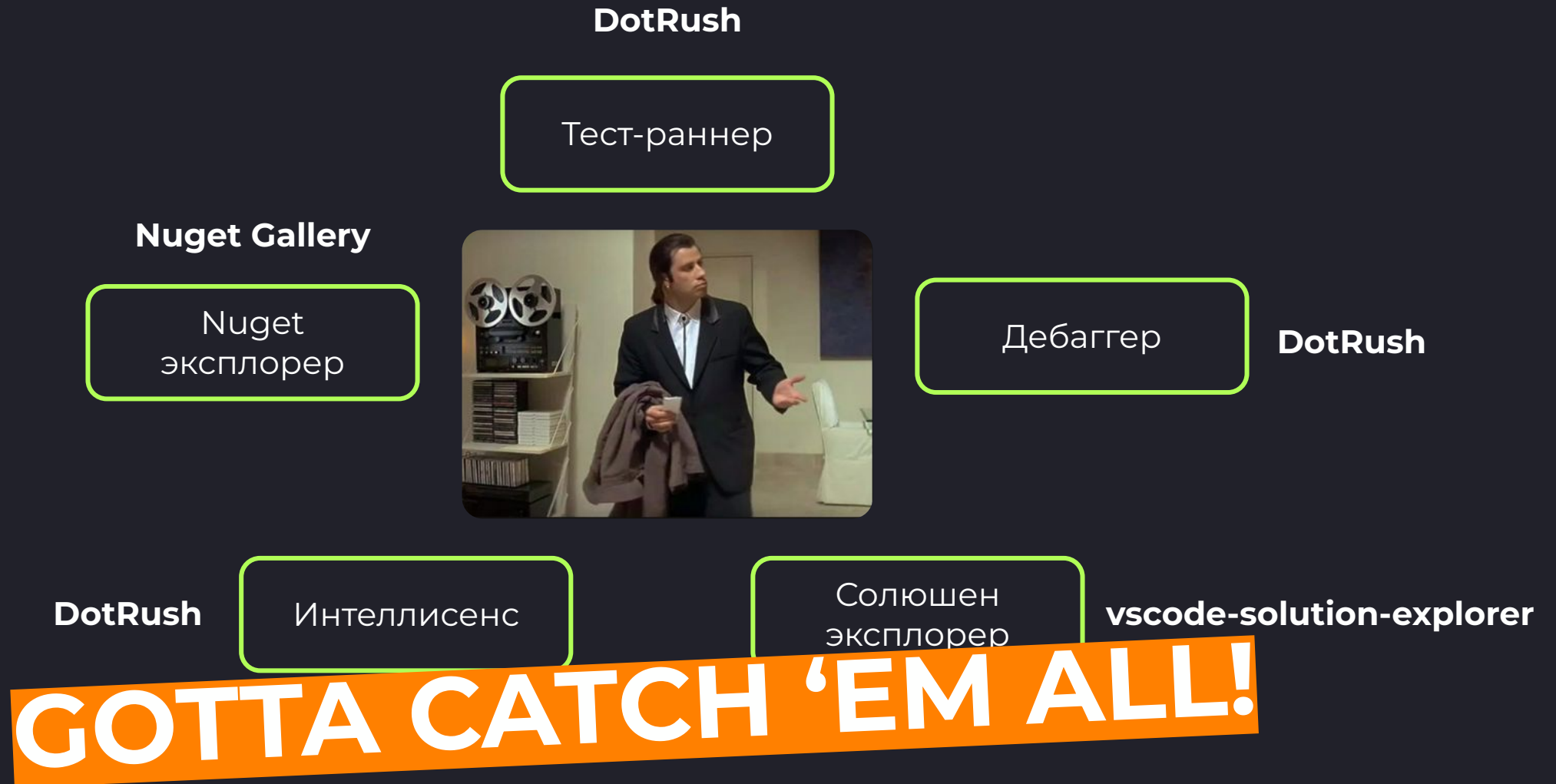
ЧТО МЫ ХОТИМ?



ЧТО МЫ ХОТИМ?



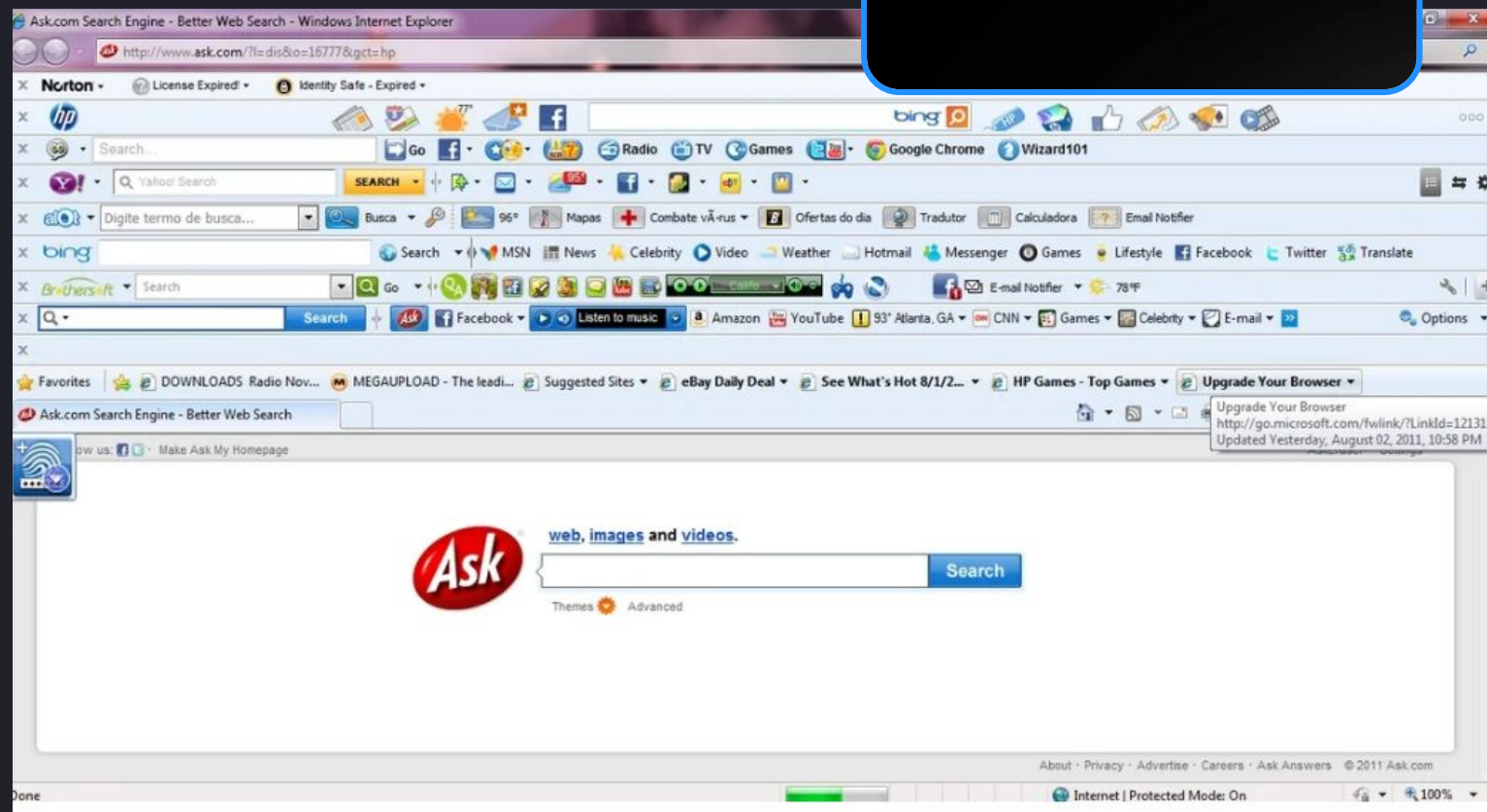
ЧТО МЫ ХОТИМ?



ZverIDE

➤ Первая сборка

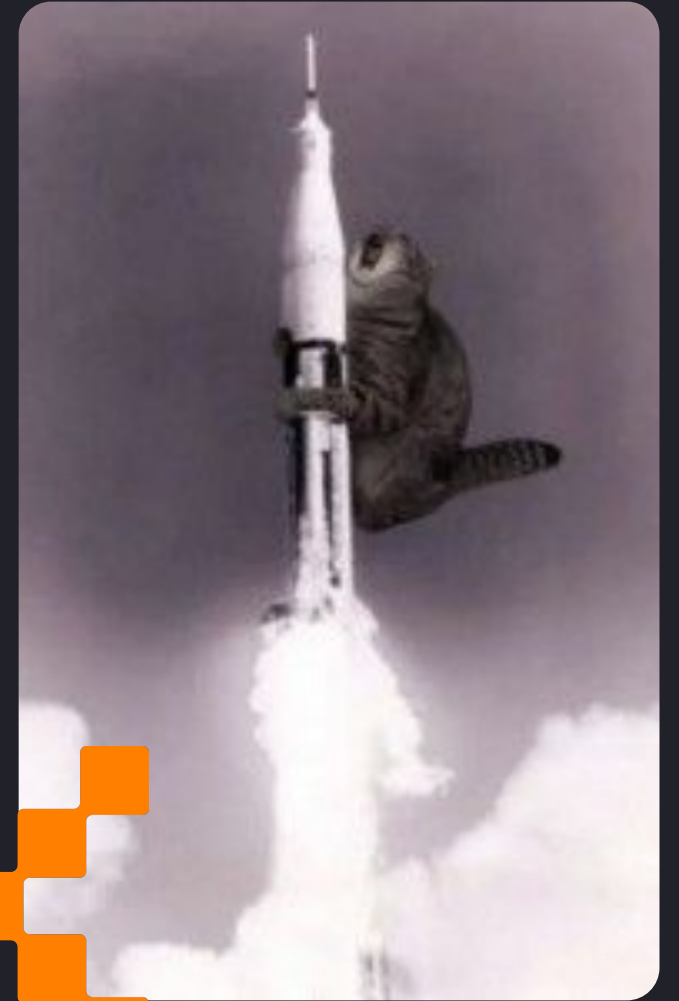
➤ Включили все что ребята просили



Результаты первых тестов

➤ Все тормозило и было криво

➤ Вопрос к зрителям, почему?



DotRush

Делает много, но не так как нужно:

- ❏ Парсит файлы сам, и делает это плохо
- ❏ Полное отсутствие разделения ответственности в коде

Пример: Контроль статус-бара хранит глобальное состояние какой проект сейчас собирать



DotRush: пример кода

Парсим csproj регулярками (не повторять)

```
string content = File.ReadAllText(projectPath);
content = XmlCommentRegex().Replace(content, string.Empty);
/* Find in current project */
var propertyMatch = new Regex($@"<{propertyName}\s?.*>(.*?)</{propertyName}>\s*\n").Matches(content);
if (propertyMatch.Count > 0)
    return propertyMatch;
var importRegex = XmlImportRegex();
/* Find in imported project */
foreach (Match importMatch in importRegex.Matches(content)) {
    var importedProjectPath = ResolveImportPath(projectPath, importMatch.Groups[1].Value);
    if (!File.Exists(importedProjectPath))
        return null;
```

Solution Explorer: Парсинг sln файла, через startsWith

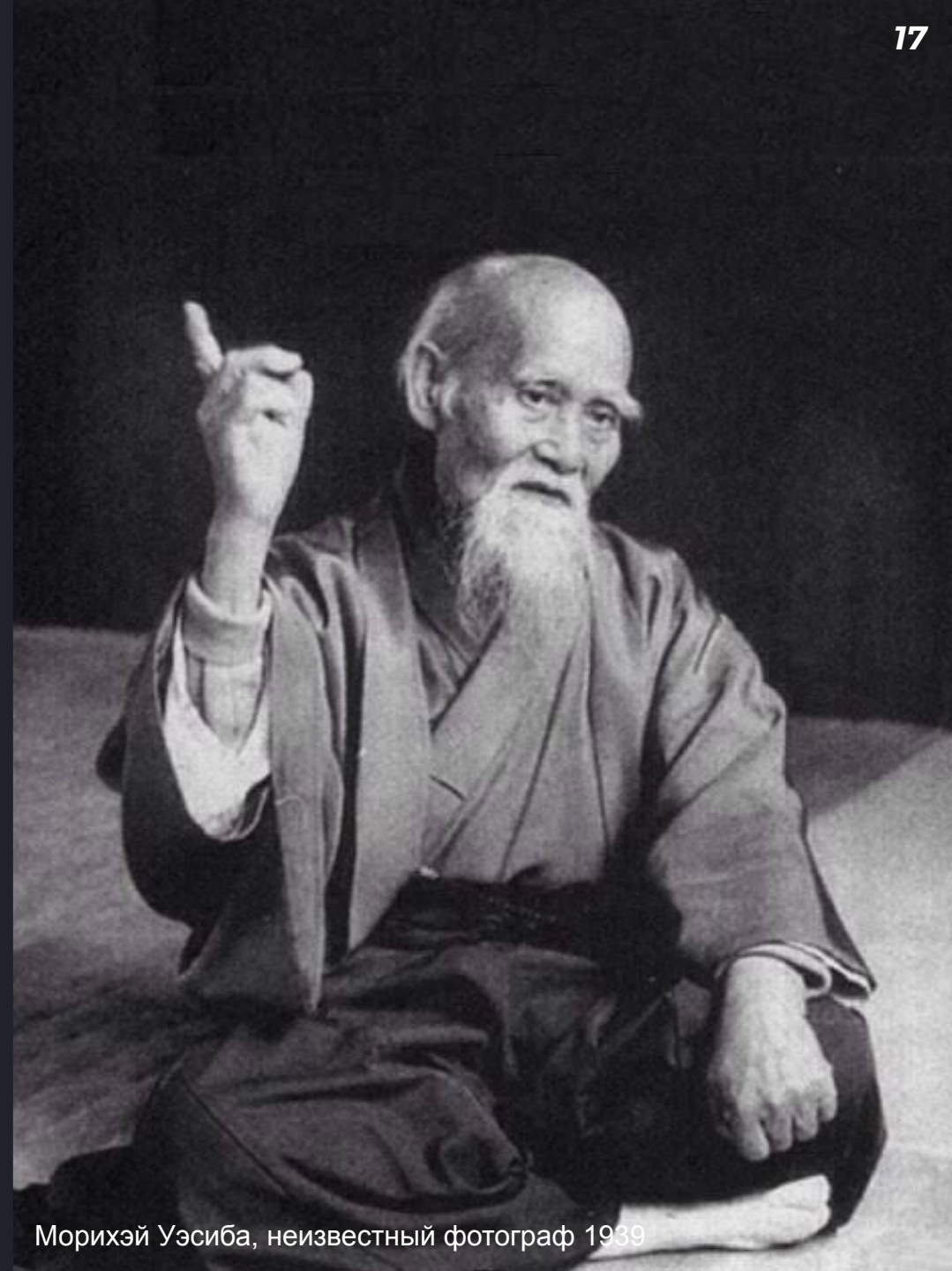
```
private parseSolution(): void {
    this.parseFileHeader();

    let str: string | null;
    let rawProjectConfigurationsEntries: { [id: string]: string } | undefined;
    while ((str = this.readLine()) !== null)
    {
        if (str.startsWith("Project("))
        {
            this.parseProject(str);
        }
        else if (str.startsWith("GlobalSection(NestedProjects)"))
        {
            this.parseNestedProjects();
        }
        else if (str.startsWith("GlobalSection(SolutionConfigurationPlatforms)"))
```

Уроки

➤ Opensource не значит battle-tested

➤ На первый взгляд ок, в масштабе
– тонна багов



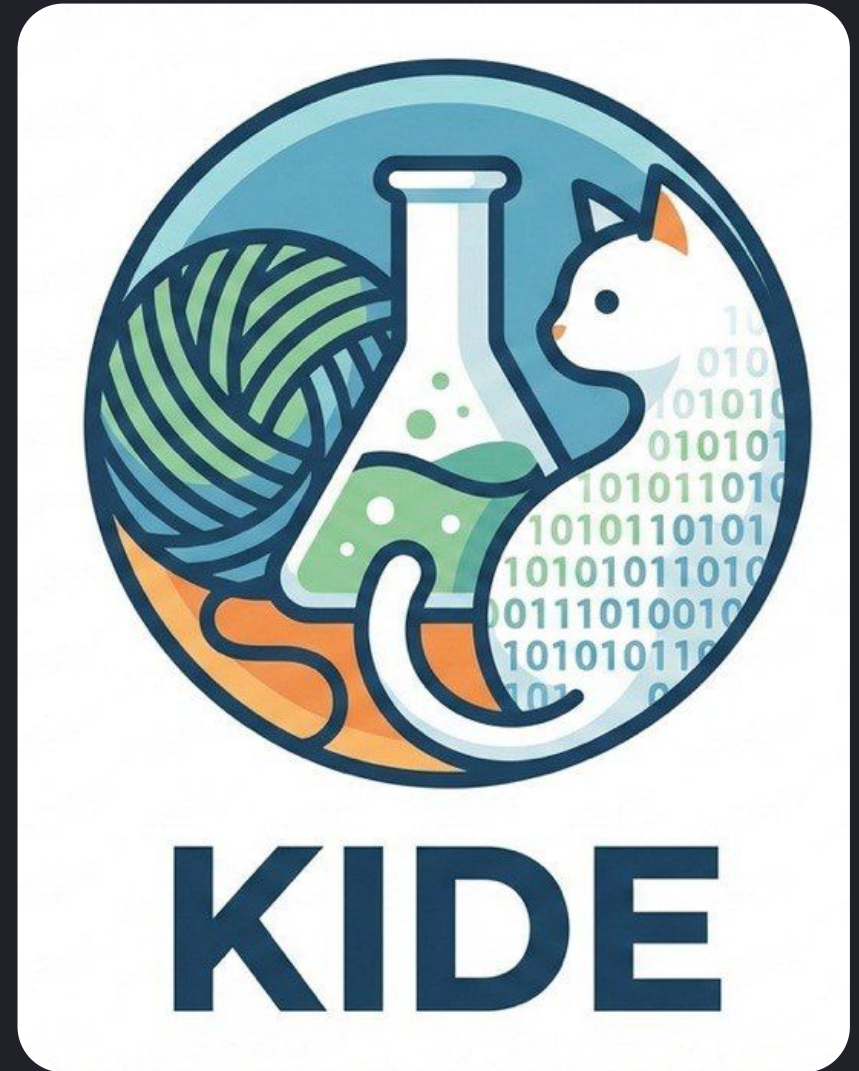
Морихэй Уэсиба, неизвестный фотограф 1939

Акт 3

Делаем IDE как нужно

KIDE (Kontur IDE)

- Создаем бренд и собираем комьюнити
- VSCodium (вырублена телеметрия, можно свой маркет)
- Наши форки плагинов
- Настройки профилей
- Логотип с котом (без него ничего не будет работать)



Что получилось

The screenshot displays the Visual Studio IDE with the following components:

- EXPLORER:** Shows the project structure for 'KIDE-TEST-EXPLORER'. The 'Kide.Server' folder is expanded, showing subfolders like 'References', 'Dispatchers', 'Extensions', 'Handlers', 'Models', 'MSBuild', 'Services', and 'NuGetTestSupport'.
- SOURCE CODE:** The file 'XUnitStrategy.cs' is open, showing the following code:


```

7 namespace Kide.Server.Handlers.TestExplorer.Strategies;
9 public class XUnitStrategy : ITestFrameworkStrategy
128     private static bool IsValidArgumentCount(IMethodSymbol method, int argCount)
144     {
145         return true;
146     }
147
148     private static string FormatValue(TypedConstant constant)
149     {
150         if (constant.IsNull)
151             return "null";
152
153         if (constant.Kind == TypedConstantKind.Enum)
154         {
155             var csString = constant.ToCSharpString();
156             var dotIdx = csString.LastIndexOf('.');
157             if (dotIdx != -1)

```
- NUGET PACKAGE MANAGER:** The 'KIDE NUGET' tab is active, showing a list of installed packages:

Package Name	Version
EmmyLua.LanguageServer.Framework	0.9.2
JUnitXml.TestLogger	6.1.0
LibGit2Sharp.NativeBinaries	2.0.323
Microsoft.Build	18.4.0
Microsoft.Build.Locator	1.7.8
Microsoft.CodeAnalysis.CSharp.Features	5.3.0
Microsoft.CodeAnalysis.Workspaces.MSBuild	5.3.0
- RIGHT SIDE:** The details for 'EmmyLua.LanguageServer.Framework' are shown, including its version (0.9.2), description, author (CppCXY), license (MIT), and project URL.

АКТ 4

Техническое устройство

Пробуем снова



Тест-раннер

Nuget
эксплорер



Дебаггер

Интеллисенс

Солюшен
эксплорер

Техническое устройство

Берем из DotRush, но переписываем 90%

Вторая попытка:

- Обозреватель тестов
- Обозреватель решений
- Обозреватель Nuget
- Дебаггер



Техническое устройство

Берем из DotRush, но переписываем 90%

Вторая попытка:

- Обозреватель тестов
- Обозреватель решений
- Обозреватель Nuget
- Дебаггер

Пишем сами

Nuget
эксплорер

Тест-раннер

NetCoreDbg

Дебаггер



Используем
открытую
часть от MS

Интеллисенс

Солюшен
эксплорер

Пишем сами

Интеллисенс



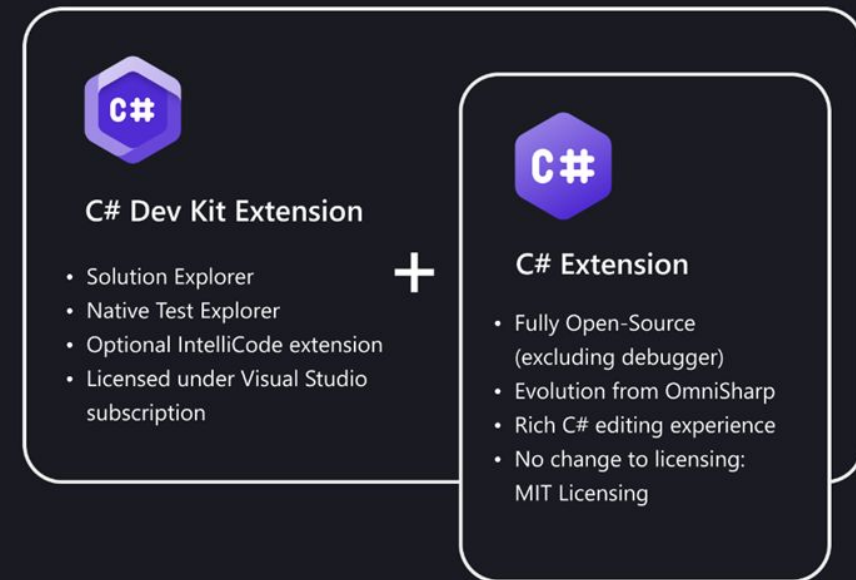
Интеллисенс берем от MS

Плагины от майкрософт
для VSCode для C#

“C#” — Интеллисенс, рефакторинги (MIT)

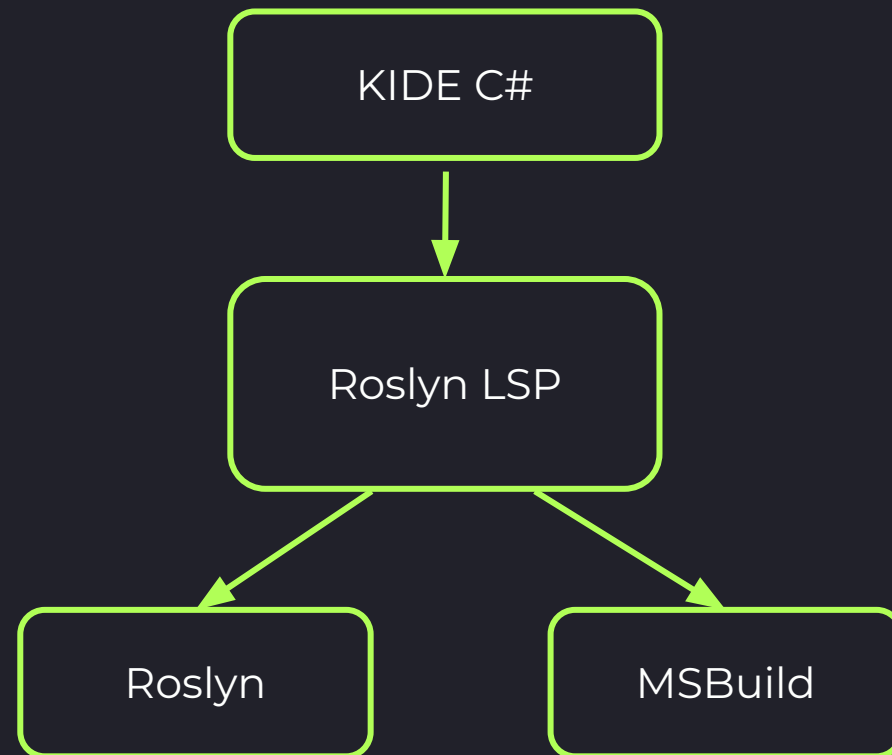
“C# DevKit” — Тест Эксплорер и Солюшен
Эксплорер (EULA)

В итоге наше решение состоит двух
плагинов, один форк от ms второй наш
плагин.



Интеллисенс

- Форк плагина C# от MS
- LSP (Language Server Protocol)
Протокол, как делать поддержку языка без привязки к конкретному редактору
- JSON-RPC
- Запускается c# сервер, который отвечает на команды



Интеллисенс

- Roslyn LSP: Часть кодовой базы Roslyn-a
- Код используется и в большом Visual Studio
- Мы пока заносим issue, форкать репозитории с компилятором c# — плохая идея)
- НО, нас игнорят по этому будет накладывать патчи



Как мы запускаем тесты



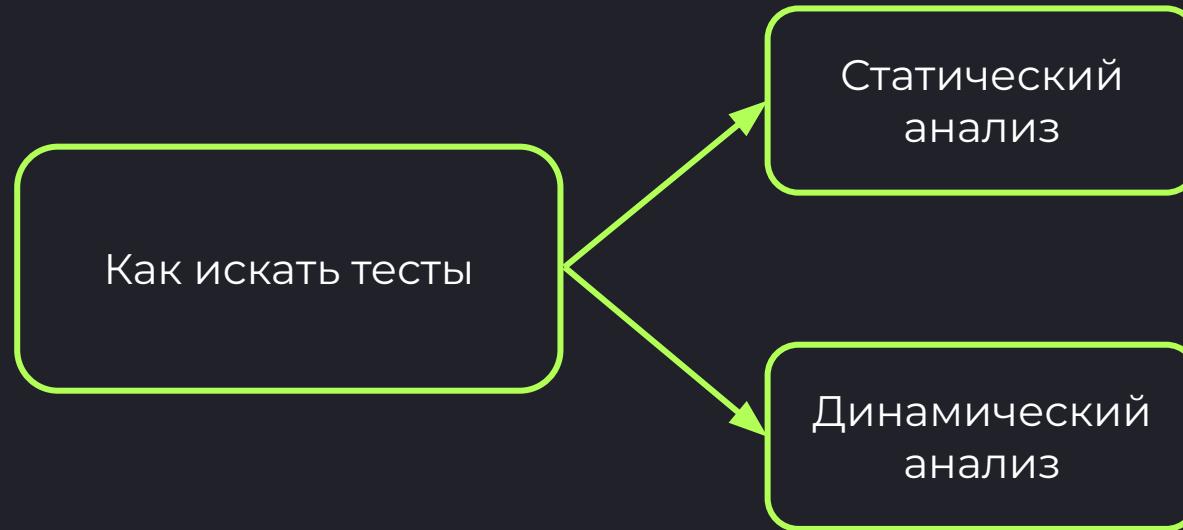
Как мы запускаем тесты

The screenshot displays the KIDE (Kotlin IDE) interface. On the left, the 'TESTING' sidebar shows a tree view of test projects. Under 'Kide.Server.Tests', the 'NUnitRuntimeIntegrationTests' project is expanded, listing various test cases. The selected test case is 'TestCaseWithCustomTestName_RunsEachIndividually(meth...)'. The main editor on the right shows the source code for 'NUnitRuntimeIntegrationTests.cs'. The code defines a 'SandboxedTestBase' class with a 'TestCaseSourceInParameterizedFixture_FilterByOneCaseName()' method. This method uses 'RpcTestHostHelper' to build a project and run tests with a specific filter. It also includes a 'Theory' section with 'InlineData' attributes for testing division operations. The 'MyFixture' class is defined with two test cases: 'Divide ten by two' and 'Divide ten by one'.

```

src > Kide.Server.Tests > TestExplorer > NUnit > NUnitRuntimeIntegrationTests.cs > {} Kide.Server.Tests > NUnitRuntime
7 namespace Kide.Server.Tests;
9 public class NUnitRuntimeIntegrationTests : SandboxedTestBase
364     public async Task TestCaseSourceInParameterizedFixture_FilterByOneCaseName()
389     }
390 };
391
392     var testDll = await RpcTestHostHelper.BuildProject(sandbox, TestContext.Current
393
394     var result = await RpcTestHostHelper.RunHostWithFilter(testDll,
395         [new TestRunInfo("NUnitTestProject.MyFixture(Https).TestWithSource(2,2)")]
396         TestContext.Current.CancellationToken);
397
398     Assert.True(result.Total > 0, $"Expected at least 1 test. Total={result.Total}
399
400
401 [Theory]
402 [InlineData("NUnitTestProject.MyFixture.Divide ten by two", "Divide ten by two", n
403 [InlineData("NUnitTestProject.MyFixture.Divide ten by one", "Divide ten by one", n
404 public async Task TestCaseWithCustomTestName_RunsEachIndividually(
405     string methodFqn, string displayName, string? parameters)
406 {
407     var sandbox = await CreateSandbox(nameof(TestCaseWithCustomTestName_RunsEachIn
408
409     AddTest(sandbox, @"
410 public class MyFixture
411 {
412     [TestCase(10, 2, TestName = "Divide ten by two")]
413     [TestCase(10, 1, TestName = "Divide ten by one")]
414     public void DivisionTest(int a, int b) {}
415
  
```

Как искать тесты



Простой тест

Атрибут класса (есть)

Атрибут метода (есть)

Для запуска теста нам нужно
знать название тест метода +
тест-класса

```
using NUnit.Framework;

namespace Calculator.Tests;

[TestFixture]
public class CalculatorTests
{
    [Test]
    public void Add_TwoNumbers_ReturnsSum()
    {
        var calculator = new Calculator();

        var result = calculator.Add(2, 3);

        Assert.That(result, Is.EqualTo(5));
    }
}
```

Сложность – Наследование

В базовом классе
может быть тест, но
атрибута еще нет

```
public abstract class CalculatorTestBase
{
    [Test]
    public void Add_TwoNumbers_ReturnsSum()
    {
        Assert.That(1 + 2, Is.EqualTo(3));
    }
}

[TestFixture]
public class CalculatorTests : CalculatorTestBase
{
    [Test]
    public void Subtract_TwoNumbers_ReturnsDifference()
    {
        Assert.That(5 - 3, Is.EqualTo(2));
    }
}
```

Сложность – Наследование

Атрибут теста может
быть на базовом
методе

```
public abstract class SerializationTestsBase
{
    [Test]
    public abstract void RoundTrips_Data();
}

[TestFixture]
public class JsonSerializerTests : SerializationTestsBase
{
    public override void RoundTrips_Data() { /* ... */ }
}
```

Сложность – Параметризация

Базовый класс может быть параметризован

Теперь это два теста хотя у нас атрибут Test а не TestCase

```
[TestFixture(1)]
[TestFixture(2)]
public abstract class ParametrizedBase
{
    private readonly int _value;
    protected ParametrizedBase(int value) => _value = value;

    [Test]
    public void Value_IsAsExpected()
    {
        Assert.That(_value, Is.InRange(1, 2));
    }
}

[TestFixture]
public class DerivedParametrizedTests : ParametrizedBase
{
    public DerivedParametrizedTests(int value) : base(value) { }
}
```

Сложность – partial

не забываем про partial
и SourceGenerators

```
public partial class CalculatorTests
{
    [Test]
    public void Add_TwoNumbers_ReturnsSum()
    {
        Assert.That(new Calculator().Add(2, 3), Is.EqualTo(5));
    }
}
```

Файл 2: CalculatorTests.Strings.cs

```
using NUnit.Framework;

public partial class CalculatorTests
{
    [Test]
    public void Multiply_TwoNumbers_ReturnsProduct()
    {
        Assert.That(new Calculator().Multiply(2, 3), Is.EqualTo(6));
    }
}
```

Как мы ищем тесты в коде (Roslyn)

В рамках одного файла мы не можем надежно сказать есть ли тест

Мы используем **Roslyn** для статического анализа

Design-Time-Build



Находим атрибуты

```
private static async Task<List<INamedTypeSymbol>> GetClassSymbolsWithAttributes(SemanticModel model,
    FrozenSet<string> attributes,
    CancellationToken ct)
{
    var root = await model.SyntaxTree.GetRootAsync(ct);
    var result = new List<INamedTypeSymbol>();
    foreach (var node in root.DescendantNodes())
    {
        if (node is not ClassDeclarationSyntax classDecl)
            continue;

        var symbol = model.GetDeclaredSymbol(classDecl);
        if (symbol is not INamedTypeSymbol namedType)
            continue;
        if (namedType.IsAbstract || namedType.IsStatic)
            continue;
        if (!SymbolHasAttributeWithInheritance(namedType, attributes))
            continue;

        result.Add(namedType);
    }
    return result;
}
```

Пример поиска атрибутов в конкретном файле

Находим атрибуты тест кейсов

```
private TestMethodCase? ToTestMethodCase(string testClassName,
    IMethodSymbol method,
    AttributeData attribute,
    int? lineNumber)
{
    var methodFqn = $"{testClassName}.{method.GetFullName()}";

    var attributeClassName = attribute.AttributeClass?.ToString();
    if (attributeClassName == null)
        return null;

    ITestFrameworkStrategy? strategy = null;
    foreach (var s in strategies)
    {
        if (s.TestCaseAttributes.Contains(attributeClassName))
        {
            strategy = s;
            break;
        }
    }
    return strategy?.CreateTestCase(method, attribute, testClassName, methodFqn, lineNumber);
}
```

У xUnit несколько вариации атрибутов

Название теста

Тестовому фреймворку требуется название теста:

Есть несколько представлений:

- представление в Roslyn
- представление через Рефлексию
- представление в vstest

```
private static string ToXUnitAttributeFormat(TypedConstant constant)
{
    if (constant.IsNull)
        return "null";

    //убираем полный тип
    //XUnitTestProject.BillingProduct.Unknown -> Unknown
    if (constant.Kind == TypedConstantKind.Enum)
    {
        var csString = constant.ToCSharpString();
        var dotIdx = csString.LastIndexOf('.');
        if (dotIdx != -1)
            return csString.Substring(dotIdx + 1);
        return csString;
    }
    if (constant.Kind == TypedConstantKind.Array)
    {
        return "[" + string.Join(", ", constant.Values.Select(v => ToXUnitAttributeFormat(v))) + "]";
    }

    // XUnit сериализует в полный формант,
    // то есть 0.7 становится 0,69999999999999996
    if (constant.Kind == TypedConstantKind.Primitive)
    {
        var value = constant.Value;
        var type = value!.GetType();
        if (type == typeof(double))
        {
            var result = ((double)value).ToString("F17");
            return result;
        }
        if (type == typeof(bool))
            return (bool)value ? "True" : "False";
    }
}
```

Испытываем боль разных атрибутов NUnit и XUnit

Тестируем поиск тестов

Куча
Edge-кейсов

Для поиска мы
сканировали
наши
ИСХОДНИКИ

```
[Fact]
public async Task FindTestCase_InheritedTestMethod()
{
    var sandbox = await CreateSandbox(nameof(FindTestCase_InheritedTestMethod),
        TestContext.Current.CancellationToken);
    var handler = new TestExplorerHandler(sandbox.Workspace);
    AddTest(sandbox, """
    public class BaseFixture
    {
        [Test]
        public void BaseTest() {}
    }

    [TestFixture]
    public class DerivedFixture : BaseFixture
    {
        [Test]
        public void DerivedTest() {}
    }
    """);

    var result = await handler.FindTestCase(
        new FindTestRequest() {
            DisplayName = "BaseTest",
            Fqn = " NUnitTestProject.DerivedFixture.BaseTest" },
        TestContext.Current.CancellationToken);

    Assert.Single(result);
}
```

Динамический анализ

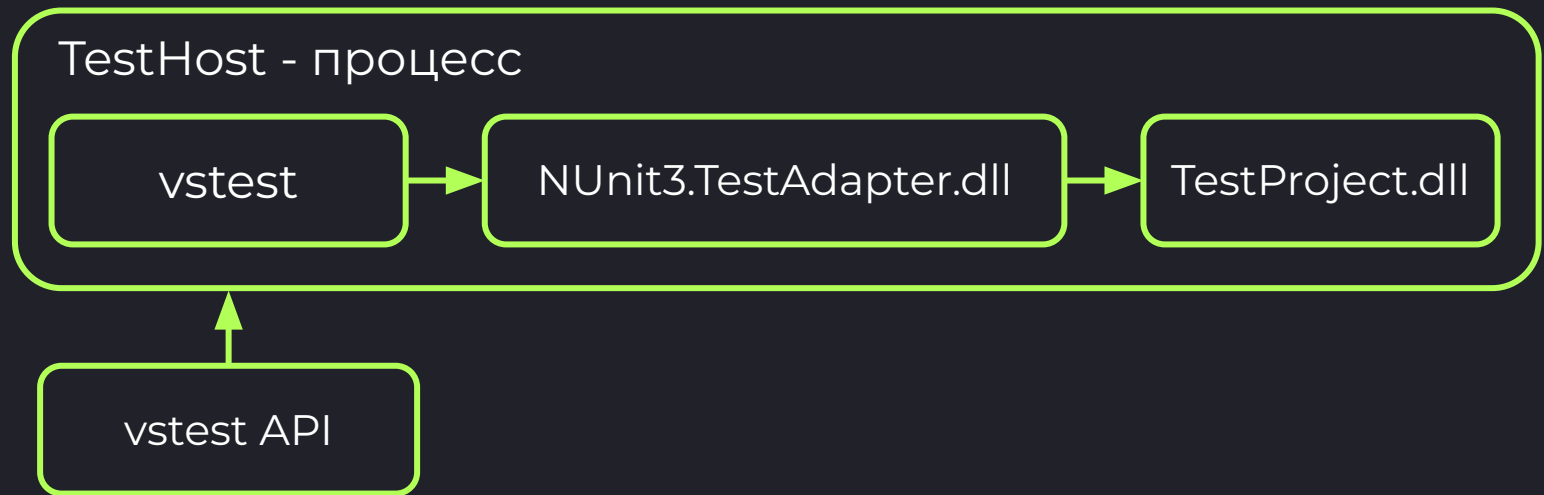
Если тест-кейсы определяются кодом, то стат-анализ бесполезен - нужно запускать код

```
public class CalculatorTests
{
    private static IEnumerable<TestCaseData> AddCases()
    {
        yield return new TestCaseData(2, 3, 5);
        yield return new TestCaseData(-1, 1, 0);
        yield return new TestCaseData(0, 0, 0);
    }

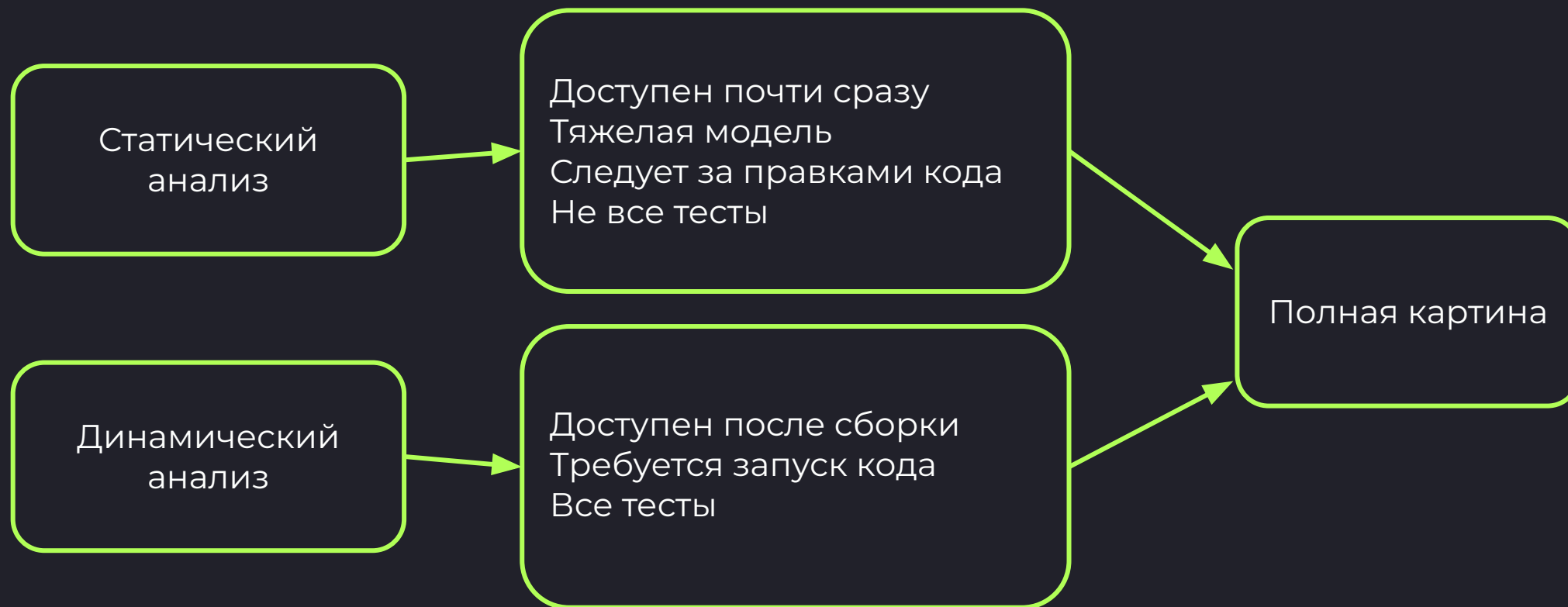
    [TestCaseSource(nameof(AddCases))]
    public void Add_ReturnsSum(int a, int b, int expected)
    {
        Assert.That(new Calculator().Add(a, b), Is.EqualTo(expected));
    }
}
```

Динамический анализ

Какие тесты есть,
нам должен сказать тест-
адаптер
конкретного
фреймворка

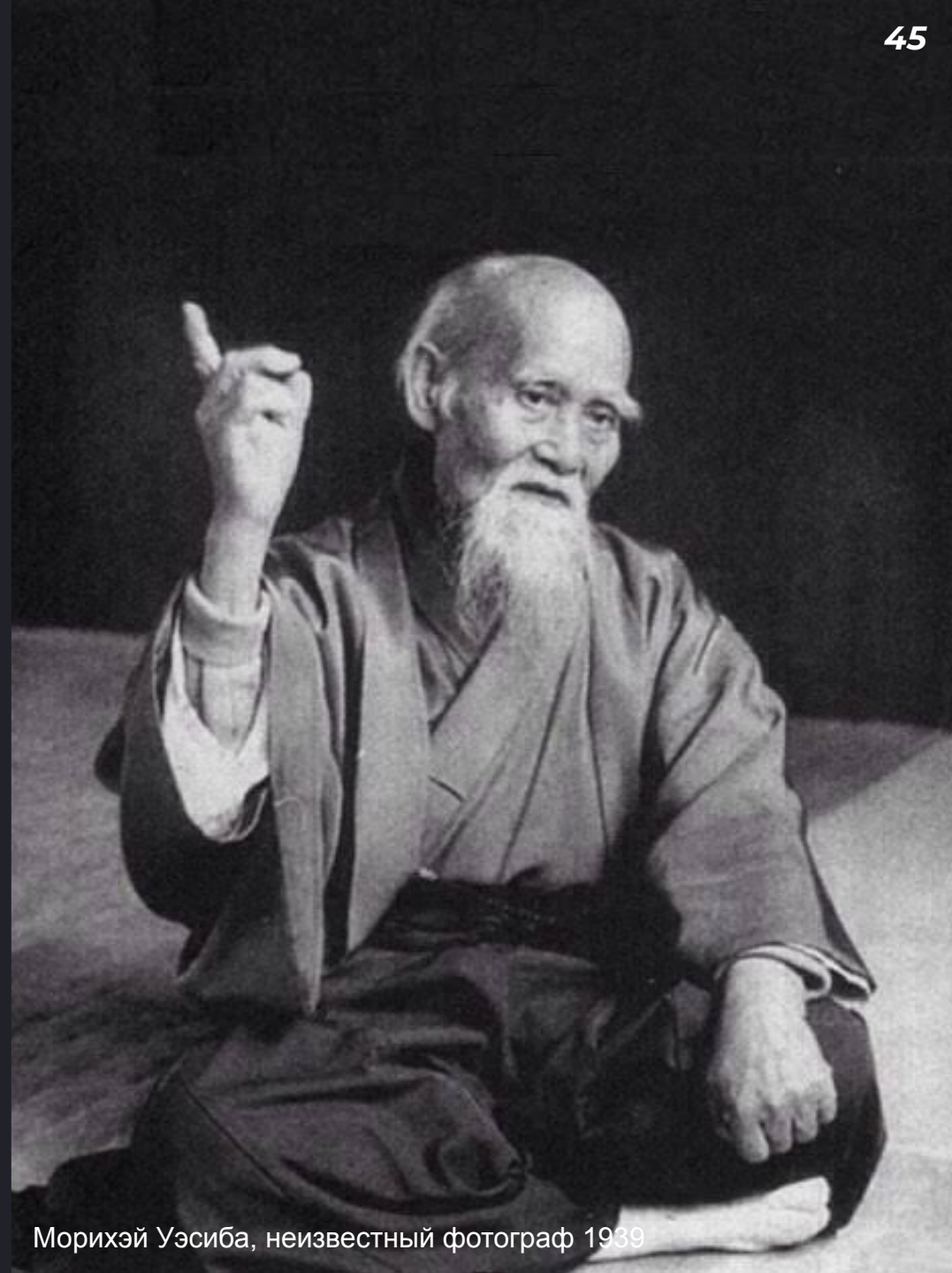


Нам нужны оба типа анализа



Уроки

- Боль, много боли и edge-case
- VSTest странная штука
- MTP у нас тоже есть.



Морихэй Уэсиба, неизвестный фотограф 1939

Как мы дебажим



Как мы дебажим

The screenshot displays the Visual Studio Code interface during a debugging session. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The address bar shows the file path: kide-test-explorer.

Left Sidebar:

- VARIABLES:** Shows local variables for the current scope: `this = Kide.Server.Tests.NUnitRuntimeIntegrationTests`, `sandbox = null`, `testDll = null`, and `result = null`.
- WATCH:** Shows a watch item: `fixtures = error: The name 'fixtures' does not exist ...`.
- CALL STACK:** Shows the call stack with frames: `Kide.Server.Tests` (RUNNING), `Kide.Server.Tests 2` (PAUSED ON BREAKPOINT), `<No name>` (PAUSED), `<No name>` (PAUSED), `.NET Timer` (PAUSED), and `.NET TP Worker` (PAUSED ON BREAKPOINT).
- BREAKPOINTS:** Shows a list of breakpoints with checkboxes and line numbers: `user-unhandled`, `all`, `build.cake` (64), `FqnPatternTests.cs` (9), `FqnPatternTests.cs` (172), `KideCommands.ts` (1236), `NUnitFindTestCaseTests.cs` (321), `NUnitRuntimeIntegrationTests.cs` (90), and `NUnitRuntimeIntegrationTests.cs` (366).

Main Editor: Shows the source code of `NUnitRuntimeIntegrationTests.cs`. The code includes a `namespace Kide.Server.Tests`, a `public class NUnitRuntimeIntegrationTests : SandboxedTestBase`, and a `public async Task TestCaseSourceInParameterizedFixture_FilterByMethodFqn_RunsAllCases()` method. The code is paused at line 366, which is highlighted in green.

Bottom Panel: Shows the **PROBLEMS** tab with a list of errors and warnings. The **OUTPUT** tab shows the `KIDE.Debugging.Host` log, which includes the file path and the output of the `xUnit.net` VSTest Adapter. The **DEBUG CONSOLE** tab shows the `KIDE Test Explorer & Debug` output, which includes the `Test Explorer` and `Test Explorer` results.

Как мы дебажим — NetCoreDbg

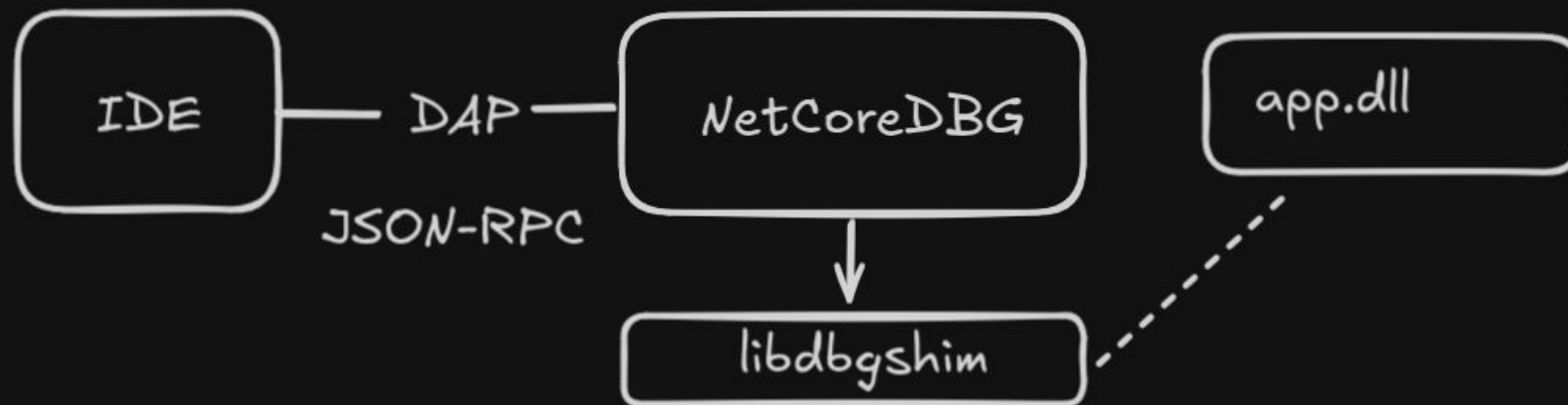
- Опенсорс дебагер от Samsung
- DAP - debug adapter protocol
Языконезависимый протокол
между редактором и дебагером

Официальный дебагер MS идет без исходников и лицензирован только для работы в VSCode но не его форках

Что мы сделали:

Заносим issue, собрали и подписали свою версию под macOS

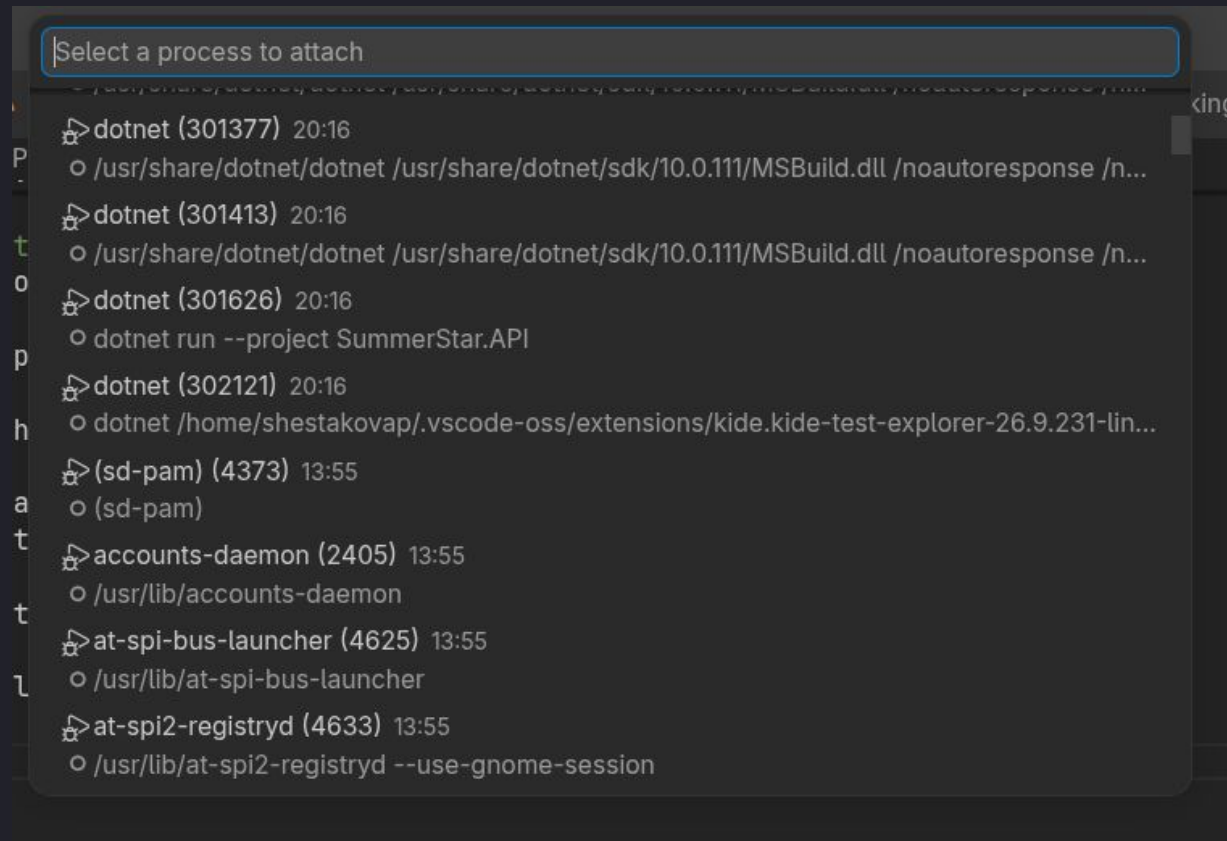
Запуск под отладкой



- Windows — named pipe `\\.\pipe\dotnet-dbgpipe-<pid>`.
- Linux/macOS — Unix domain socket `/tmp/dotnet-dbgpipe-<pid>`

Для работы NetCoreDbg работает с интерфейсом `ICorDebugProcess`

Подключение к процессу для отладки



Где взять аргументы запуска?

dotnet не имеет полноценного api для получения всей информации по запущенным процессам, используем тут платформенные низкоуровневые АПИ

Process.GetProcesses() — не содержит всю информацию

На linux проще всего - можно прочесть /proc/<pid>/cmdline а вот на остальных платформах все интереснее.

Под windows

```
private static string? GetCommandLine(SafeHPROCESS hProc)
{
    var sizeStatus = NtQueryInformationProcess(
        hProc, PROCESSINFOCLASS.ProcessCommandLineInformation, default, 0, out var len);
    if (sizeStatus != NTStatus.STATUS_INFO_LENGTH_MISMATCH || len == 0 || len > int.MaxValue)
        return null;

    string? commandLine;
    using (var mem = new SafeHGlobalHandle((int)len))
    {
        var status = NtQueryInformationProcess(
            hProc, PROCESSINFOCLASS.ProcessCommandLineInformation, mem, len, out len);
        if (status.Failed || len < UnicodeStringSize)
            return null;

        var us = mem.ToStructure<UNICODE_STRING>();
        if (us.Buffer == IntPtr.Zero || us.Length == 0)
            return null;

        commandLine = Marshal.PtrToStringUni(us.Buffer, us.Length / sizeof(char));
    }

    return commandLine;
}
```

NtQueryInformationProcess
документирован частично

Под windows

```
private static string[] SplitArgs(string commandLine)
{
    using var argv = Shell32.CommandLineToArgvW(commandLine, out int argc);
    if (argv.IsInvalid || argc == 0)
        return [];

    var ptrs = argv.ToArray<IntPtr>(argc);
    var result = new string[argc];
    for (var i = 0; i < argc; i++)
        result[i] = Marshal.PtrToStringUni(ptrs[i]) ?? string.Empty;

    return result;
}
```

W суффикс означает, что
возвращается кодировка UTF-16

MacOS

Да, sysctl можно
вызывать в дотнете

```
internal static string[] GetCommandLineArgs(int pid)
{
    try
    {
        var mib = new[] { CTL_KERN, KERN_PROCARGS2, pid };
        var len = UIntPtr.Zero;

        if (sysctl(mib, (uint)mib.Length, IntPtr.Zero, ref len, IntPtr.Zero, UIntPtr.Zero) != 0)
        {
            Logger.Warn($"sysctl size query failed for process {pid}");
            return [];
        }

        if (len == UIntPtr.Zero || len > int.MaxValue)
        {
            Logger.Warn($"sysctl returned invalid buffer length ({len}) for process {pid}");
            return [];
        }

        var buf = Marshal.AllocHGlobal((int)len);
        try
        {
            if (sysctl(mib, (uint)mib.Length, buf, ref len, IntPtr.Zero, UIntPtr.Zero) != 0)
            {
                Logger.Warn($"sysctl data query failed for process {pid}");
                return [];
            }

            return ParseArgs(pid.ToString(), buf, (int)len);
        }
        finally
        {
            Marshal.FreeHGlobal(buf);
        }
    }
}
```

Недокументированное АПИ

Как происходит аттач к процессу

Подключение dotnet отладчика отличается от нативного

1

На Windows — через `DebugActiveProcess`
+ `LoadLibrary`

На Linux — через
`ptrace(PTRACE_ATTACH)`
+ инъекция `libdbgshim` (`dlopen` через
`/proc/<pid>/mem`)

2

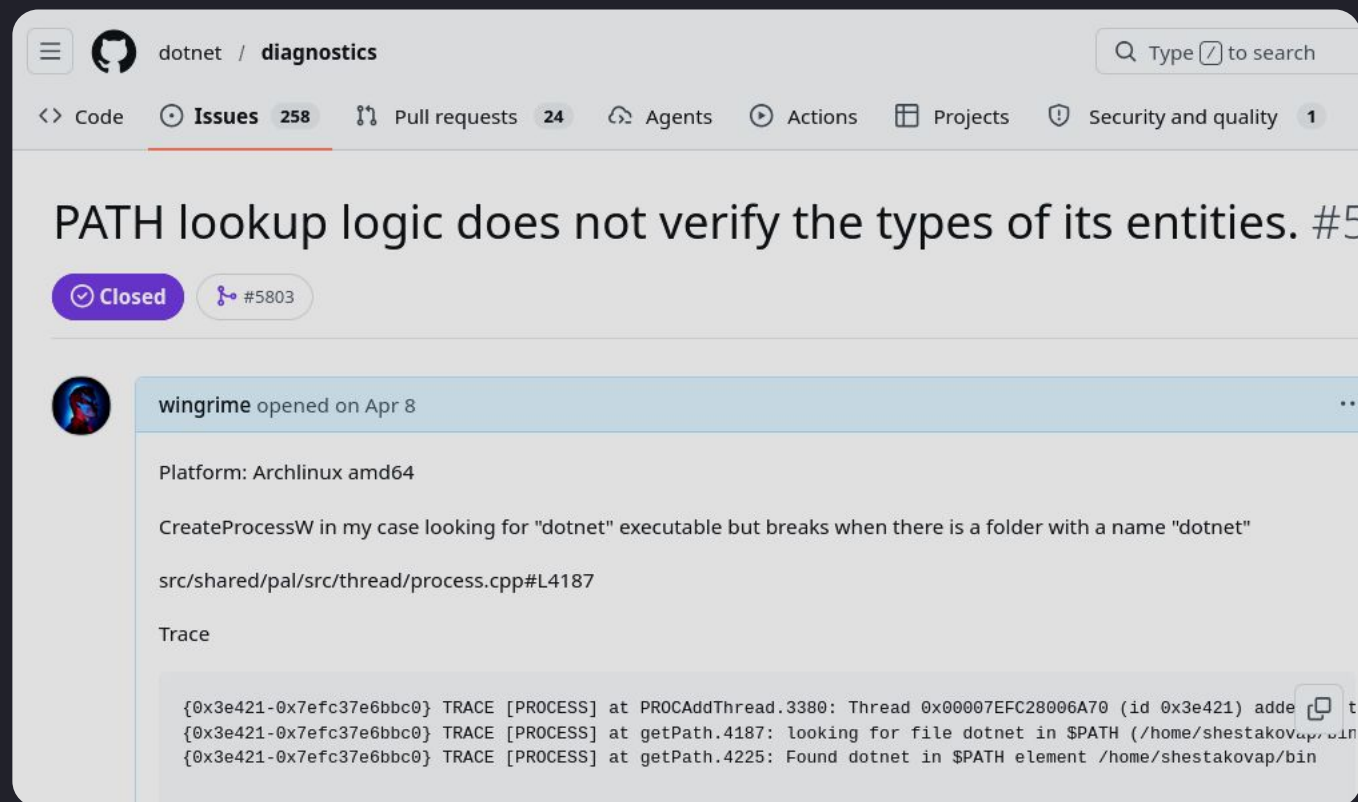
`dbgshim` находит нужную версию
рантайма в процессе и отдаёт
дебаггеру `ICorDebugProcess`

Интересные баги: Дебаггер не подключается на linux

На linux не запускался дебаггер

В PATH была папка `dotnet`

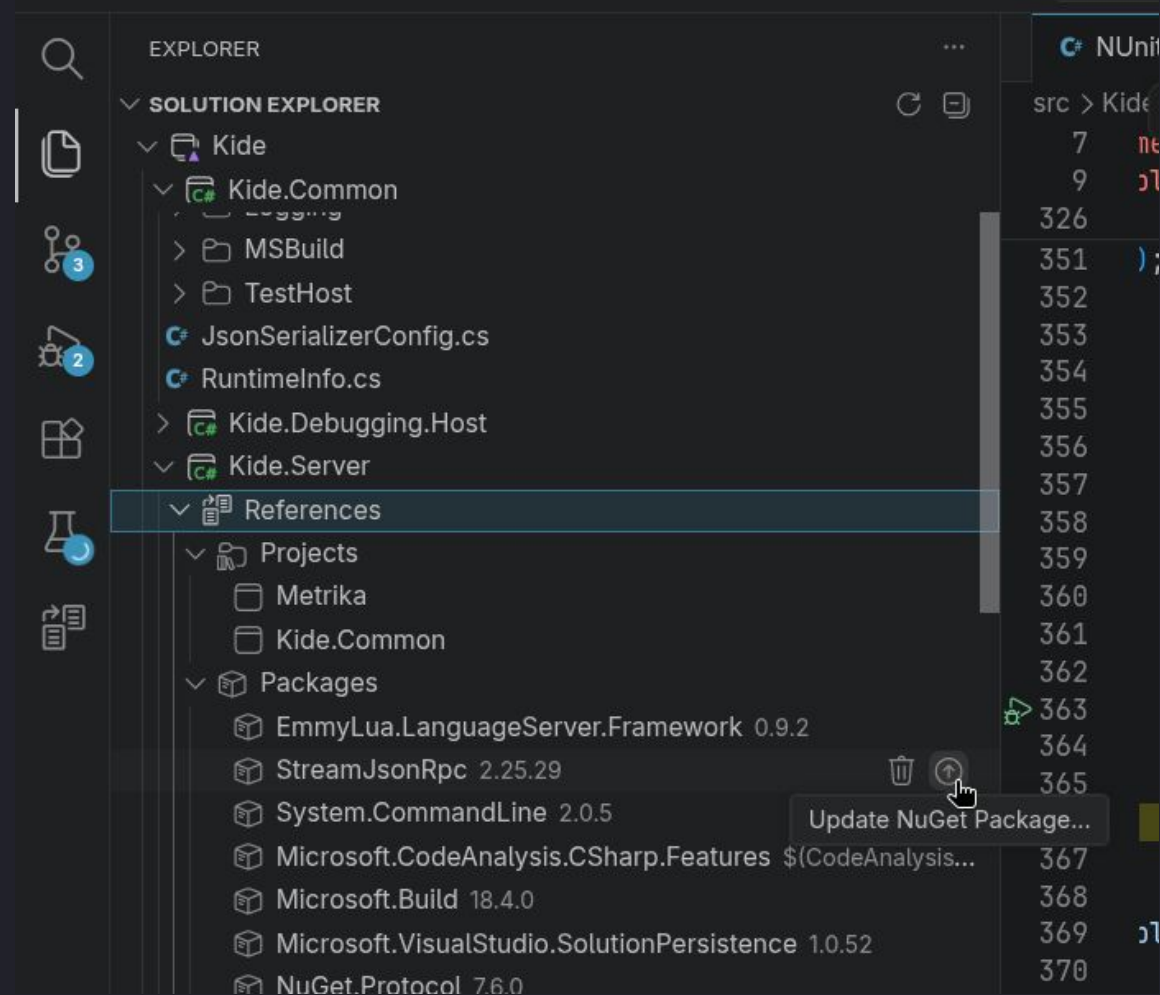
Код в libdbgshim подумал что папка это приложение и пытался запустить его
MS этот баг поправили



Solution Explorer



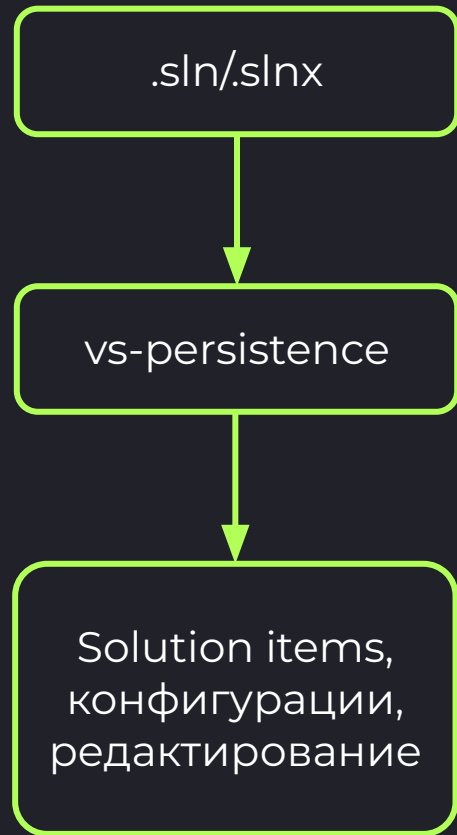
Solution Explorer



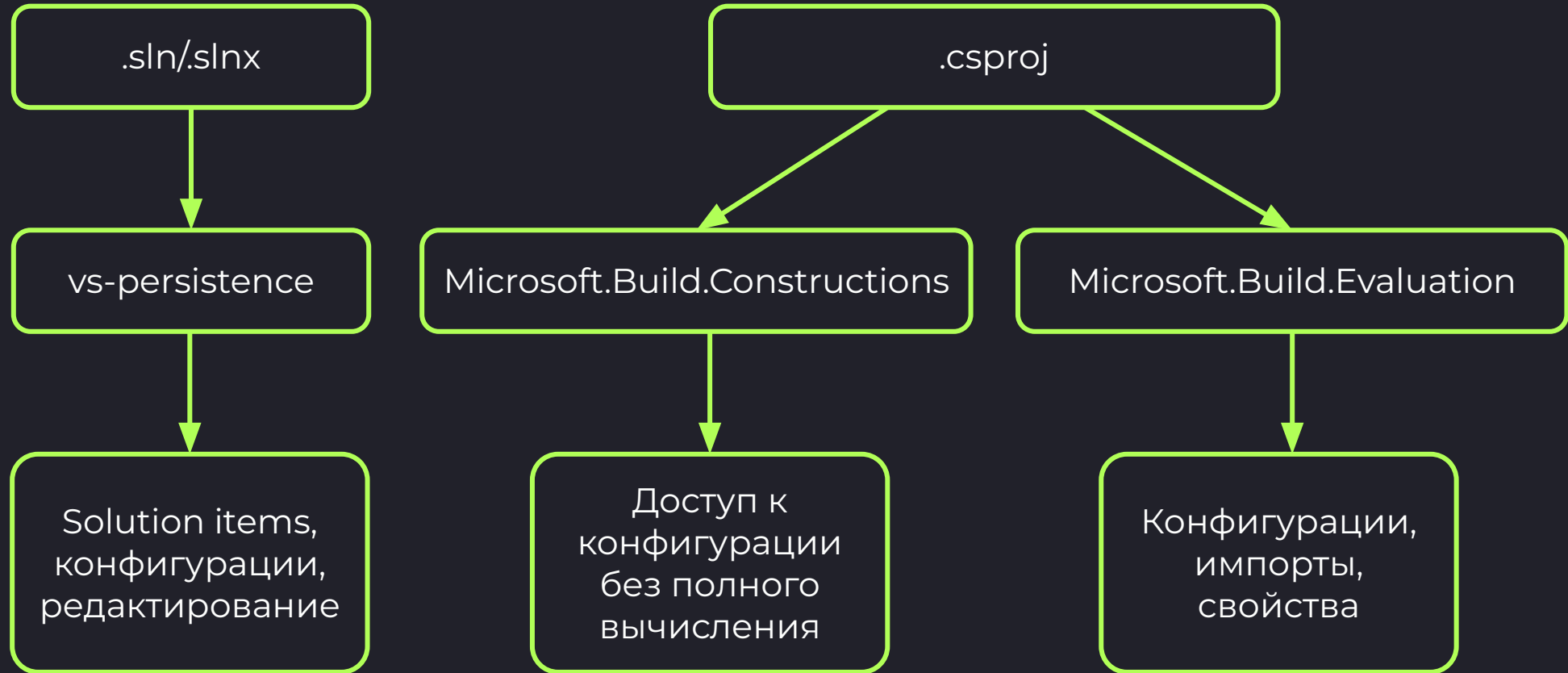
Solution Explorer: что под капотом

- Microsoft.Build.Construction – операции с .csproj (чтение, модификация)
- Microsoft.Build.Evaluation – “flatten” представление решения, вид для сборки
- vs-persistence – чтение, модификация sln/slnx файлов

Solution Explorer

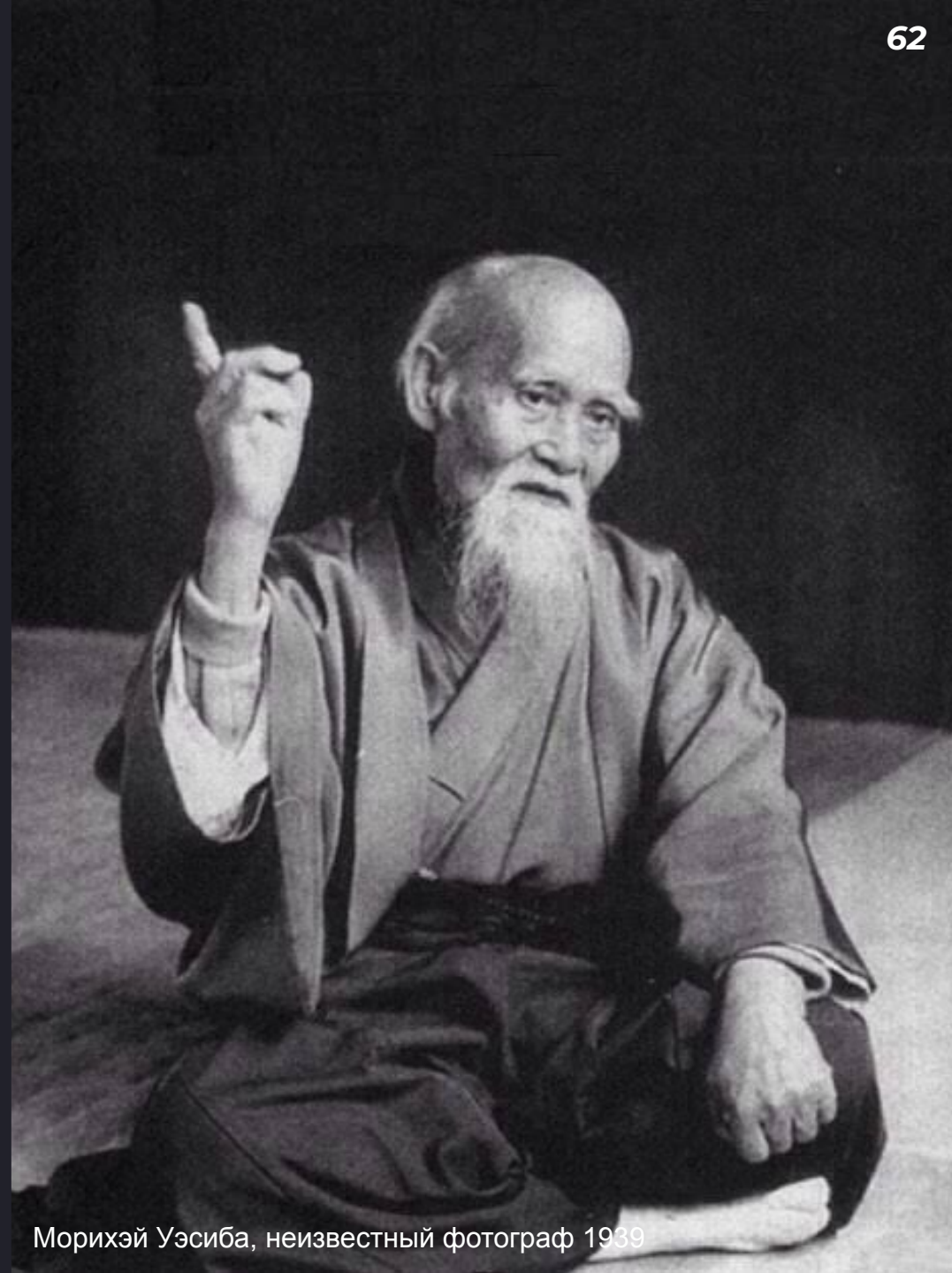


Solution Explorer



Уроки

- Подписание (нотификация) под макось
- В dotnet куча костылей чтобы все работало.



Морихэй Уэсиба, неизвестный фотограф 1939

Как мы работаем с Nuget



Как мы работаем с Nuget

The screenshot shows the Visual Studio NuGet Package Manager interface. The top bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, TEST RESULTS, PORTS, KIDE NUGET, and NUGET. Below the tabs is a search bar labeled "Search NuGet packages..." and two dropdown menus: "Current Solution" and "All sources". A checkbox for "Prerelease" is checked. The main area is divided into two sections: "INSTALLED" and "DISCOVER". Under "INSTALLED", a list of packages is shown, with "BlurHashSharp 1.3.4" selected. To the right of the package list, the details for "BlurHashSharp" are displayed, including its version (1.3.4), a trash icon, and an "INFO" section with fields for DESCRIPTION, AUTHOR(S), LICENSE, PROJECT URL, TAGS, and PUBLISHED.

Package Name	Version	Update
BlurHashSharp	1.3.4	1.4.0-pre.1
coverlet.collector	6.0.4	10.0.1
Dapper	2.1.66	2.1.79
Microsoft.Extensions.DependencyIn...	10....	11.0.0-preview.7.26381.103
Microsoft.Extensions.DependencyInje...	9...	11.0.0-preview.7.26381.103
Microsoft.Extensions.Http	9.0.3	11.0.0-preview.7.26381.103

BlurHashSharp

Version: 1.3.4

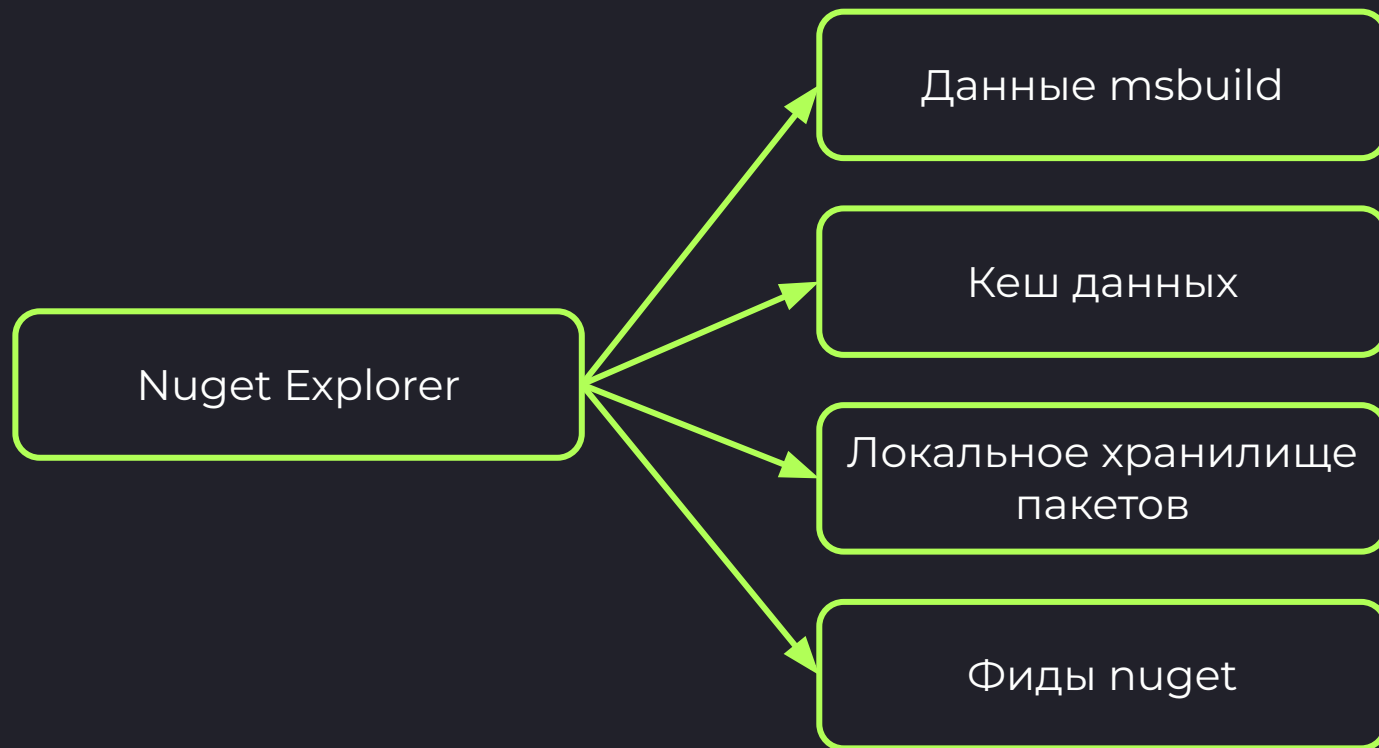
INFO

- DESCRIPTION**
BlurHash encoder for working with raw bitmaps in memory.
- AUTHOR(S)**
Bond_009
- LICENSE**
<https://licenses.nuget.org/MIT>
- PROJECT URL**
- TAGS**
BlurHash image bitmap
- PUBLISHED**

Как мы работаем с Nuget

-  Nuget SDK
-  Немного кешей
-  Локальный кеш пакета

Как мы работаем с Nuget

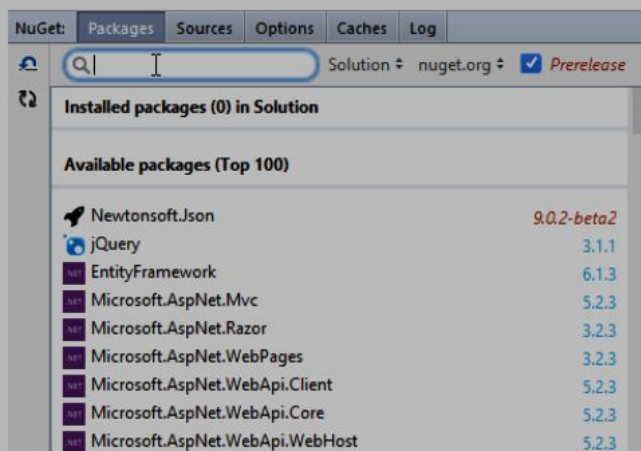


Nuget: История повторяется

Why is NuGet search in Rider so fast?

Andrey Akinshin · 2017-02-08

I'm the guy who develops the NuGet manager in [Rider](#). It's not ready yet, there are some bugs here and there, but it already works pretty well. The feature which I am most proud of is smart and fast search:

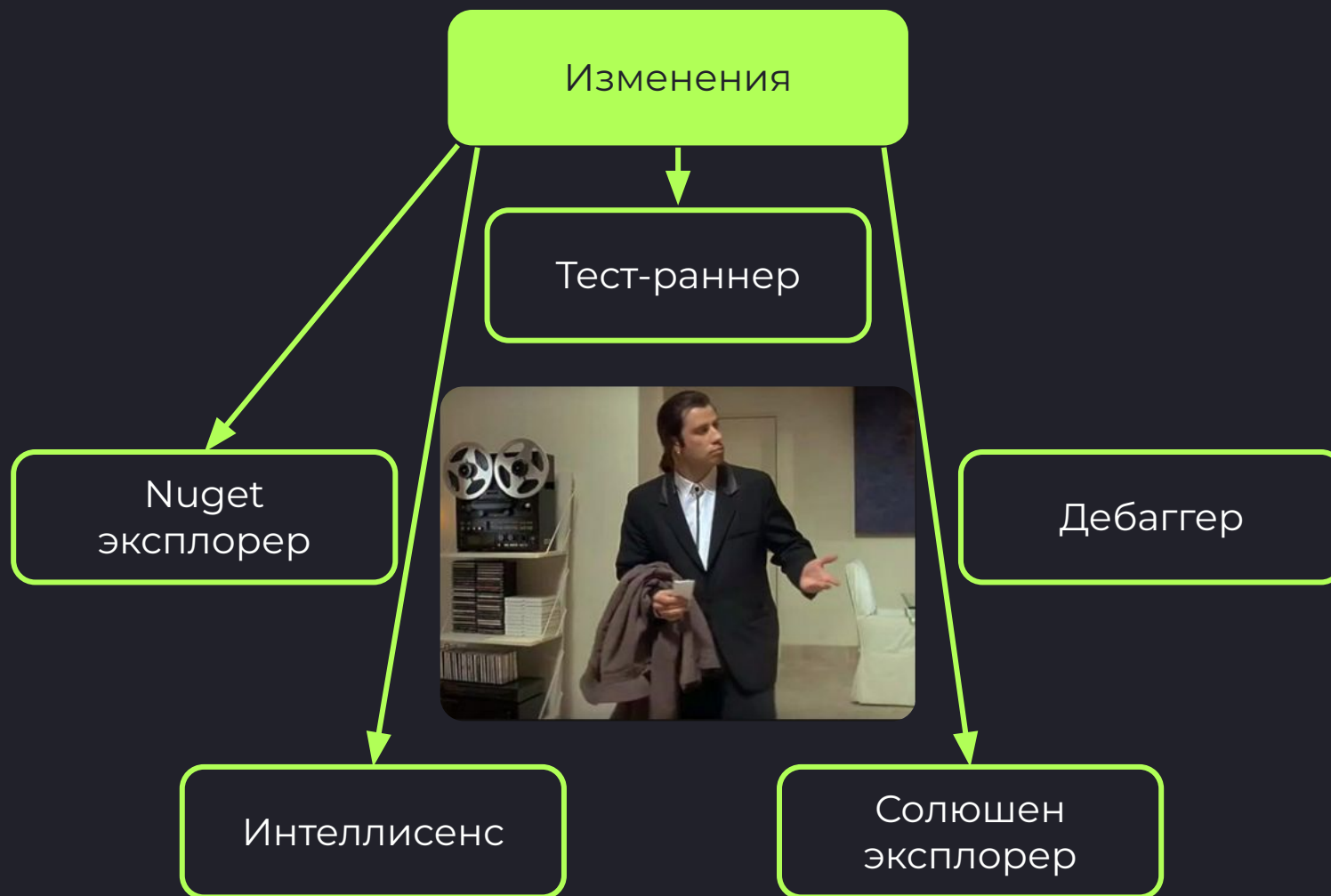


Today I want to share with you some technical details about how it was implemented.

Caching

In Rider, we are using [NuGet.Client](#). It is a package with a set of APIs which allows doing all the basic NuGet operations: install, uninstall, restore, and search. So, we can write something like this

Как мы детектим изменения



Изображение: Криминальное чтиво, Квентин Тарантино 1994

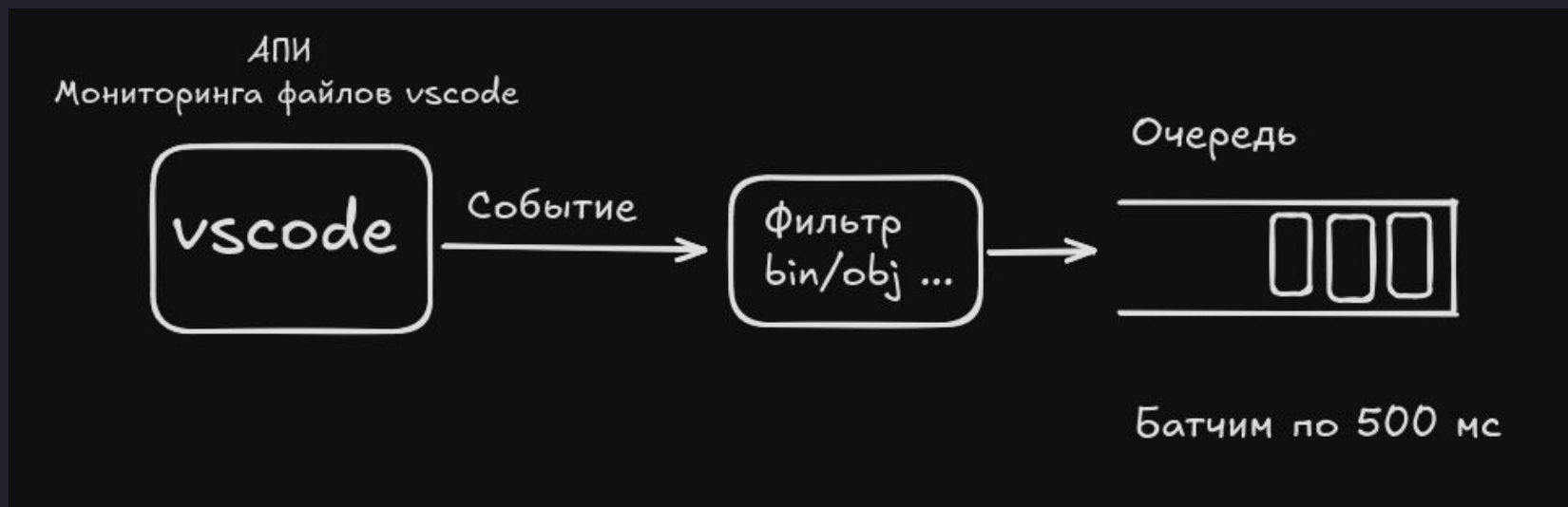
Как мы детектим изменения

Пользователь может

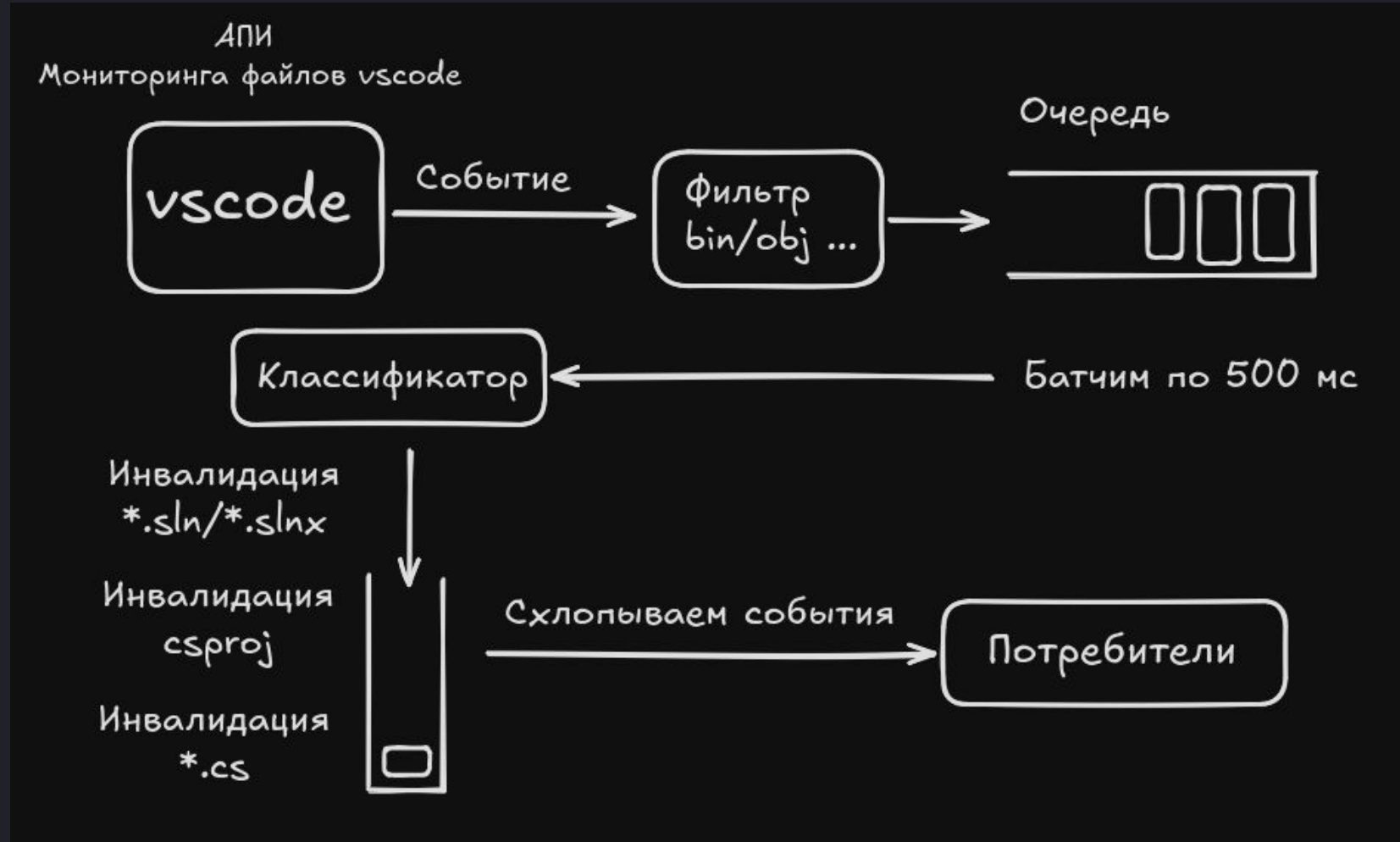
- копировать
- перемещать файлы
- чекаутить git
- ...

Нужно детектировать действия, и
делать debounce

Как мы детектим изменения



Как мы детектим изменения



Неожиданные повороты



Изображение: Криминальное чтиво, Квентин Тарантино 1994

Блокируется сборка всего, из-за `source-generators`

```
/// Swaps analyzer references to ones loaded via Roslyn's non-locking (shadow copy) analyzer loader.  
/// MSBuildWorkspace loads analyzers and source generators with the plain loader, which keeps the  
/// assemblies locked on Windows (e.g. 'obj\Debug\netstandard2.0\SourceGenerators.dll') and breaks  
/// a concurrent 'dotnet build'. The shadow copy loader copies assemblies to %TEMP% before loading.  
///  
/// Shadow copying (instead of loading from a stream) is used on Windows so that the anti-virus  
/// checker can amortize the cost of scanning the assemblies: the copies are regular files on disk  
/// that are scanned once and cached, whereas a stream-loaded assembly would be scanned on every  
/// load, which is a significant performance hit (see Roslyn's  
/// AnalyzerAssemblyLoader.CreateNonLockingLoader). The originals are only read briefly for the  
/// copy, so they are never held locked and can be overwritten by 'dotnet build'.
```

Извлечение нужного кода через рефлекссию

```
private static IAnalyzerAssemblyLoader CreateNonLockingLoader(string baseDirectory)
{
    var codeAnalysisAssembly = typeof(AnalyzerFileReference).Assembly;
    var loaderType = codeAnalysisAssembly.GetType("Microsoft.CodeAnalysis.AnalyzerAssemblyLoader");

    var createMethod = loaderType?.GetMethod("CreateNonLockingLoader",
        BindingFlags.Static | BindingFlags.NonPublic);

    var parameters = createMethod?.GetParameters();
    var arguments = new object?[parameters!.Length];
    for (var i = 0; i < parameters.Length; i++)
    {
        var parameterType = parameters[i].ParameterType;
        arguments[i] = parameterType == typeof(string)
            ? baseDirectory
            : parameterType.IsValueType ? Activator.CreateInstance(parameterType) : null;
    }

    var createdLoader = createMethod?.Invoke(null, arguments);
    return (IAnalyzerAssemblyLoader)createdLoader!;
}
```

Зависает чтение git-email у 1% пользователей

```
// Using Process to execute git config --get user.email
var process = new Process
{
    StartInfo = new ProcessStartInfo
    {
        FileName = "git",
        Arguments = "config --get user.email",
        RedirectStandardOutput = true,
        RedirectStandardError = true,
        UseShellExecute = false,
        CreateNoWindow = true
    }
};

process.Start();
string output = process.StandardOutput.ReadToEnd();
process.WaitForExit();

// Return trimmed output or empty string if no email found
return output.Trim();
```

Зависает чтение git-email у 1% пользователей

Запуск процесса git —
завешивает поток

Причина ???

Таймаут на WaitForExit
не помогает

Решение парсить самим
с libgit2 =)

```
// Using Process to execute git config --get user.email
var process = new Process
{
    StartInfo = new ProcessStartInfo
    {
        FileName = "git",
        Arguments = "config --get user.email",
        RedirectStandardOutput = true,
        RedirectStandardError = true,
        UseShellExecute = false,
        CreateNoWindow = true
    }
};

process.Start();
string output = process.StandardOutput.ReadToEnd();
process.WaitForExit();

// Return trimmed output or empty string if no email found
return output.Trim();
```

Потребление памяти?

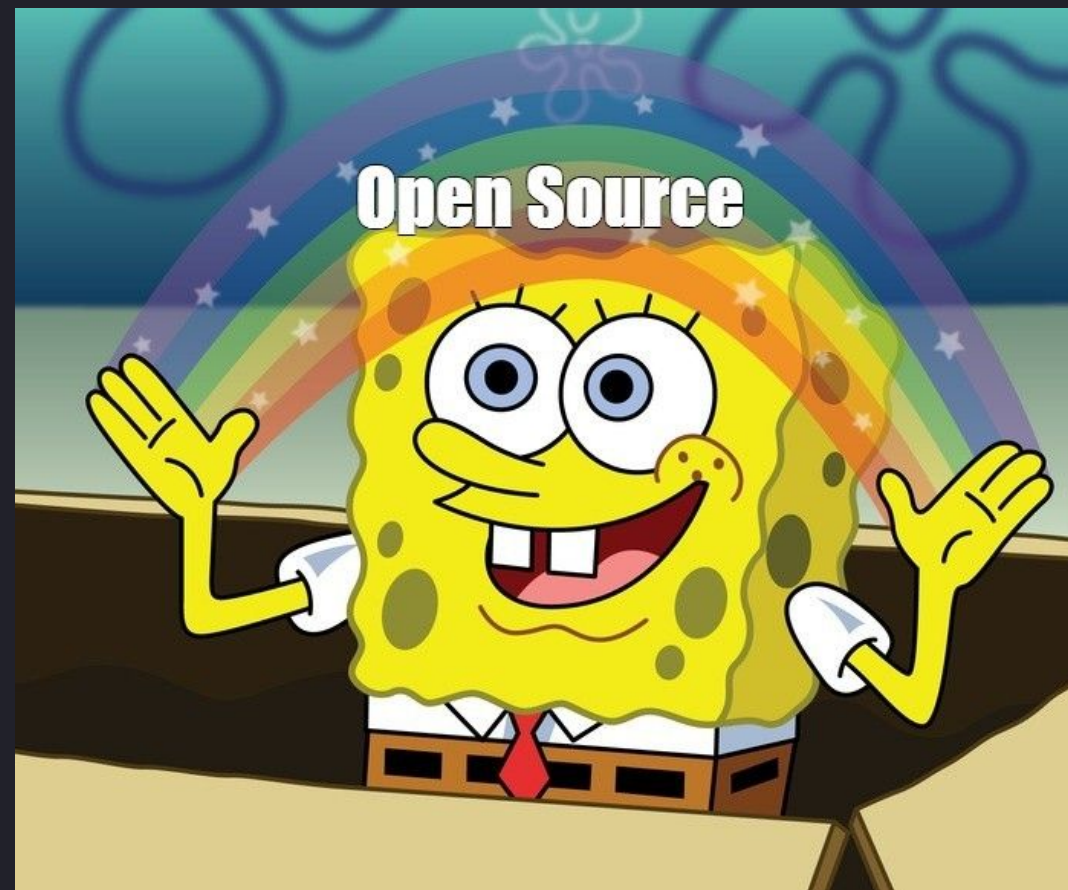
- Мы можем уменьшить потребление памяти в два раза больше чем нужно, но не все так просто.

Потребление памяти!

- У нас два плагина грузят полную модель кода (Semantic Model) в память, потребление памяти 2x

Контрибуты обратно

- Мои MR-ы и issue часто игнорят
- Придется накладывать патчи, плодить форки



Потребление памяти

➤ Из за того что у нас два плагина грузять модель кода (Semantic Model) в память, потребление памяти 2х

➤ DevKit тоже так делает



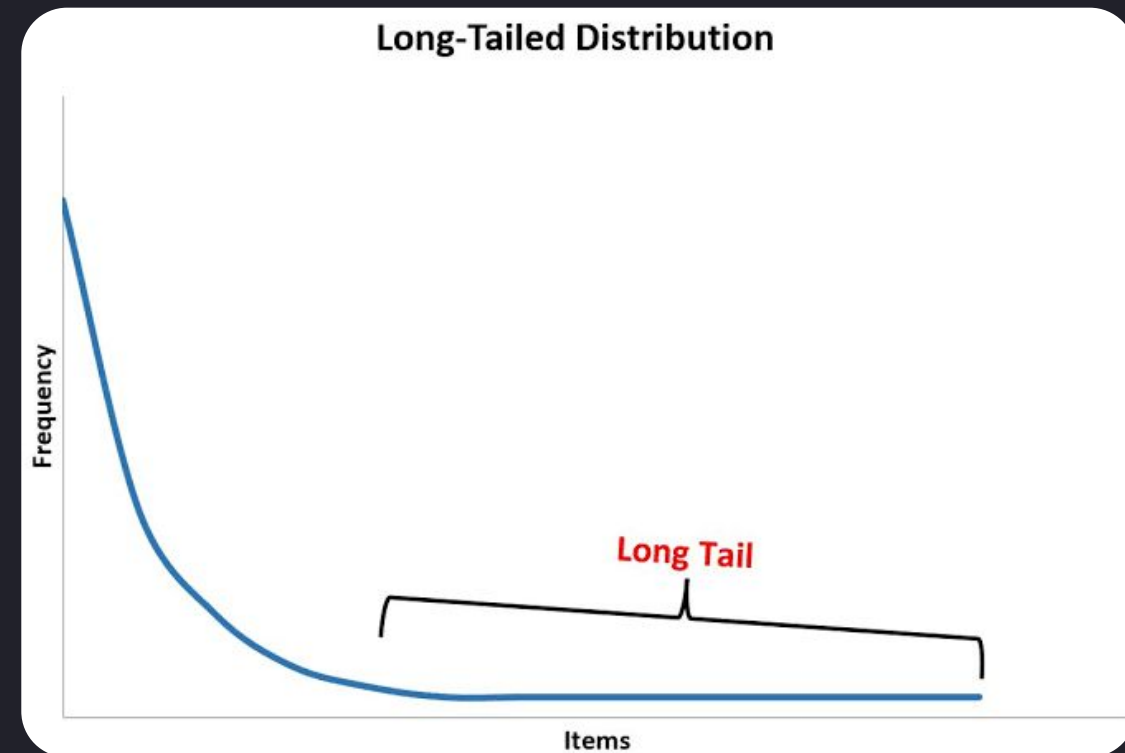
Cyanide & Happiness 2026, Kris Wilson

Дела текущие



Источник боли: Пользователи

- ❏ Длинный хвост странных проблем, хотелок
- ❏ 4 платформы для сборки
- ❏ Длинный хвост различных истории, включая организационные



Источник боли: Roslyn

- Все больше багов требует доработок в самом Roslyn
- Требуется Next level экспертизы его внутренки



Источник боли: VSCodeium

- ❏ Нет точек расширения там где ожидают пользователи
- ❏ Хочется без патчей, но копится бэклог того что нужно поправить



Что у нас хорошо, прямо сейчас

- Все базовые функции доступны
- В этом можно работать)



Что у нас пока не так хорошо



Wara Art Festival 2022

- Меньше рефакторингов, более слабый формater текстов
- Можно провалиться, ступив не туда
- Некоторая неконсистентность из-за стороннего кода

Конец!



Telegram-канал
Технологии в Контуре