



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(национальный исследовательский университет)»**

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Технологии разработки программно-информационных систем

Отчет по практическому заданию № 2

**CI/CD**

Выполнил студент группы МЗО-108СВ-25

Давыдов А.П.

Принял:

Титов Ю.П.

Москва, 2025

# Задание

Настройте систему непрерывной интеграции (CI) для вашего проекта, например, используя сервисы GitHub Actions, GitLab CI/CD, Jenkins, Travis CI или другие инструменты. Пример «Настройка CI с помощью GitHub Actions»:

Откройте репозиторий проекта на GitHub. Перейдите во вкладку Actions в верхней части страницы репозитория. Нажмите на кнопку Setup a workflow yourself или выберите один из шаблонов для создания нового workflow.

Создайте файл с названием main.yml в папке .github/workflows/ вашего репозитория и определите в нем шаги вашего CI/CD процесса.

Сделайте коммит и отправьте изменения в ваш репозиторий на GitHub.

Проверка работы CI: после отправки изменений в репозиторий перейдите во вкладку Actions на GitHub. Отобразите в отчете список запущенных workflow и их статус. Проверьте логи выполнения шагов вашего CI/CD-процесса.

Настройте CI таким образом, чтобы он автоматически запускался при каждом новом коммите в ваш репозиторий.

Добавьте скрипты для тестирования вашего проекта (например, unit-тесты, интеграционные тесты) и убедитесь, что они успешно проходят при запуске через CI.

Создайте ветвь development для разработки новых функций и исправлений.

Разработайте новую функцию или исправление ошибки в вашем проекте и закомитите изменения в ветвь development.

Убедитесь, что CI успешно запустился после коммита в ветвь development и прошел все тесты. Создайте Pull Request (PR) для вливания изменений из ветви development в основную ветвь (например, main).

Настройте автоматическое тестирование и слияние PR при успешном прохождении всех тестов.

Проверьте, что изменения из ветви development успешно влились в основную ветвь с помощью CI/CD.

Создайте ветвь release для подготовки к выпуску новой версии вашего проекта.

Подготовьте все необходимые изменения (например, обновление версии, исправление багов) в ветвь release.

Убедитесь, что CI успешно запустился после коммита в ветвь release и все тесты прошли успешно.

Создайте новый тег для версии вашего проекта и опубликуйте его на платформе Git.

Создайте документацию к вашему проекту и добавьте ее в репозиторий.

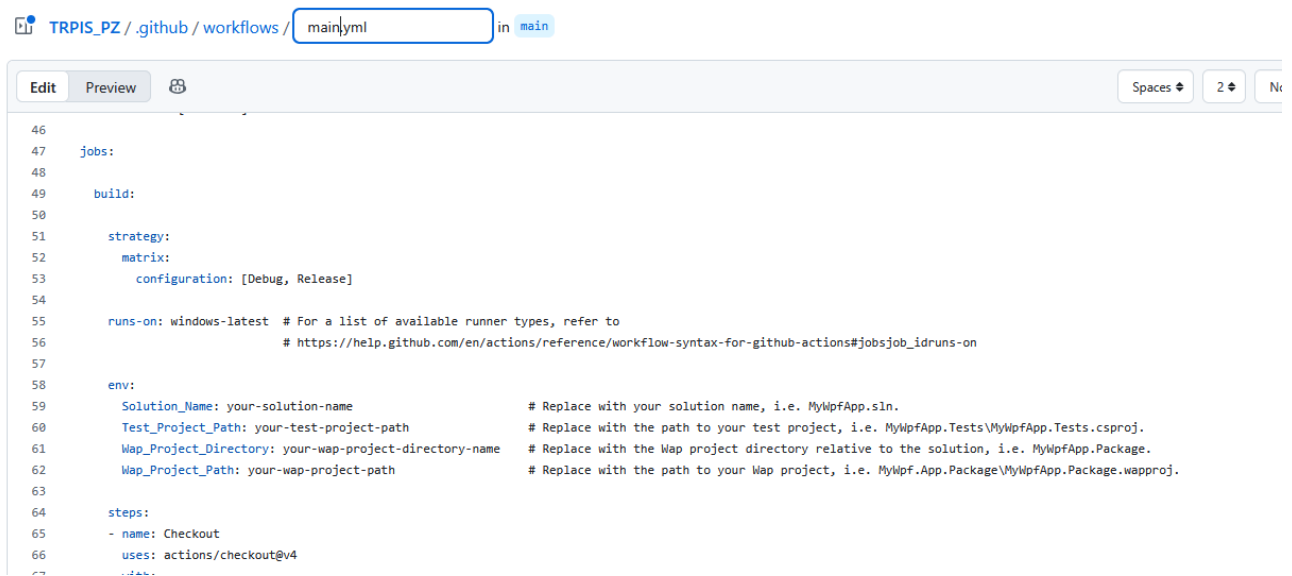
Настройте автоматическую генерацию документации при каждом обновлении кода через

CI/CD.

Проверьте, что документация успешно обновляется при каждом изменении кода.

Заключите текущую версию проекта и завершите работу над лабораторной работой.

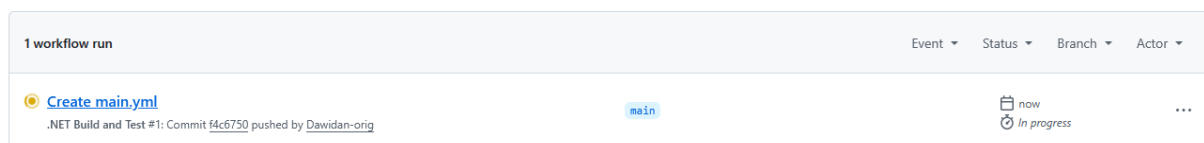
# Выполнение



Был выбран Workflow для .NET

И отредактирован до самого минимального вида.

Этот коммит уже стал учитываться:



Для контроля каждого коммита используется следующее, сразу настроил для dev:

```
on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main, dev ]
```

Что тестируется:

```
public static int Sum(int a, int b)
{
    if (a == 1)
        return b;

    return a + b;
}
```

Функция теста:

```
[Test]
Ссылки: 0
public void Test1()
{
    int[] a = { 1, 2, 3, 4 };
    int[] b = { 4, 3, 2, 1 };

    for (int i = 0; i < a.Length; i++)
        if (Program.Sum(a[i], b[i]) != a[i] + b[i])
            Assert.Fail();

    Assert.Pass();
}
```

Тест не прошёл, исправление ошибки...

Тестирование	Длительн...	Признаки	Сообщени
▲ ❌ UnitTests_TRPIS (1)	57 мс		
▲ ❌ UnitTests_TRPIS (1)	57 мс		
▲ ❌ Tests (1)	57 мс		
❌ Test1	57 мс		

И после её исправления:

Тестирование	Длительн...	Призн
▲ ✅ UnitTests_TRPIS (1)	43 мс	
▲ ✅ UnitTests_TRPIS (1)	43 мс	
▲ ✅ Tests (1)	43 мс	
✅ Test1	43 мс	

Ветвь dev была создана ещё в предыдущей работе, использую её.

Делаю коммит изменений, а потом ещё коммит с использованием функции Sum в Main()

<b>Fix (Sum) : Usage in main</b> Dawidan-orig • just now
<b>Feat : Sum added</b> Dawidan-orig • 2 minutes ago

В .csproj-файле была ошибка автоматической генерации, и она очень долго обнаруживалась. Visual Studio самостоятельно сделал просто Reference на .dll, хотя надо было на проект целиком. Когда проблема была решена, Action стал успешным.

Произошла ошибка с указанием путей, в результате чего итоговый файл (После добавления проекта с тестами) выглядит следующим образом:

```
name: .NET Build and Test

on:
  push:
    branches: [ main, dev ] //На коммиты в ветви
  pull_request:
    branches: [ main, dev ] //На Pull-request'ы в ветви

jobs:
  build:
    runs-on: windows-latest

    steps:
      - uses: actions/checkout@v3
      - name: Setup .NET SDK
        uses: actions/setup-dotnet@v3
        with:
          dotnet-version: '8.0.x'
      - name: Restore dependencies
        run: dotnet restore person1/ProjectTRPIS/ProjectTRPIS.sln
      - name: Restore dependencies for tests
        run: dotnet restore person1/UnitTests_TRPIS/UnitTests_TRPIS.sln


      - name: Build
        run: dotnet build person1/ProjectTRPIS/ProjectTRPIS.sln --no-restore --configuration Debug
      - name: Test
        run: dotnet test person1/UnitTests_TRPIS/UnitTests_TRPIS.sln --no-restore --verbosity
normal
```

19 workflow runs			Event ▾	Status ▾	Branch ▾	Actor ▾
✔	Update UnitTests_TRPIS.csproj	dev		2 minutes ago	...	
	.NET Build and Test #20: Commit <a href="#">b6ac696</a> pushed by <a href="#">Dawidan-orig</a>					1m 41s
✖	Merge branch 'dev' of https://github.com/Dawidan-orig/TRPIS_PZ into dev	dev		15 minutes ago	...	
	.NET Build and Test #19: Commit <a href="#">2d61312</a> pushed by <a href="#">Dawidan-orig</a>					1m 6s
✖	Update main.yml	dev		15 minutes ago	...	
	.NET Build and Test #18: Commit <a href="#">a3f5912</a> pushed by <a href="#">Dawidan-orig</a>					55s
✖	Update main.yml	dev		17 minutes ago	...	
	.NET Build and Test #17: Commit <a href="#">1c38605</a> pushed by <a href="#">Dawidan-orig</a>					1m 36s
✖	Update main.yml	dev		19 minutes ago	...	
	.NET Build and Test #16: Commit <a href="#">c9ca987</a> pushed by <a href="#">Dawidan-orig</a>					1m 32s

### Pull Request:
















A screenshot of the GitHub pull request status bar. It features a green icon with a white branching diagram on the left. The status bar contains two green checkmark icons. The first status is 'All checks have passed' with a subtext '1 successful check'. The second status is 'No conflicts with base branch' with a subtext 'Merging can be performed automatically.' At the bottom, there is a green button labeled 'Merge pull request' with a dropdown arrow, followed by the text 'You can also merge this with the command line.' and a blue link 'View command line instructions.'

Still in progress? [Convert to draft](#)





Add a comment


WritePreview




Add your comment here...

 Markdown is supported

 Paste, drop, or click to add files

 Close pull request

Comment

 Pull request successfully merged and closed  
You're all set — the `dev` branch can be safely deleted.

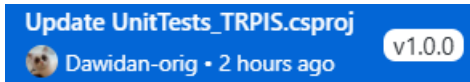
## Теперь работа с веткой Release:

Надо добавить в конфигурацию данные об этой ветви, чтобы всё прошло успешно. То-есть просто обозначить ветвь `release` кроме ветвей `main` и `dev`.

### Создание тега версии:

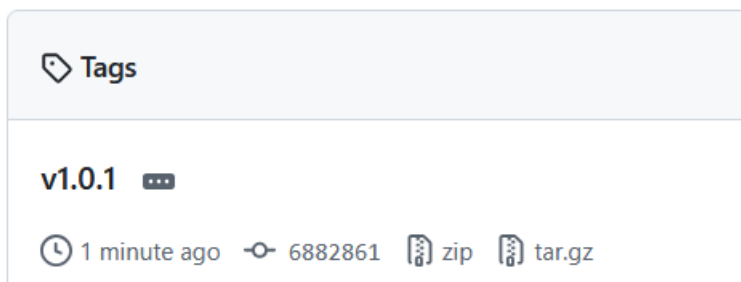
```
PS C:\Users\Dawidan> cd A:\Browser_Download\TRPIS_PZ\TRPIS_PZ
PS A:\Browser_Download\TRPIS_PZ\TRPIS_PZ> git tag -a v1.0.0 -m "Release version 1.0.0"
```

Тег отобразился:

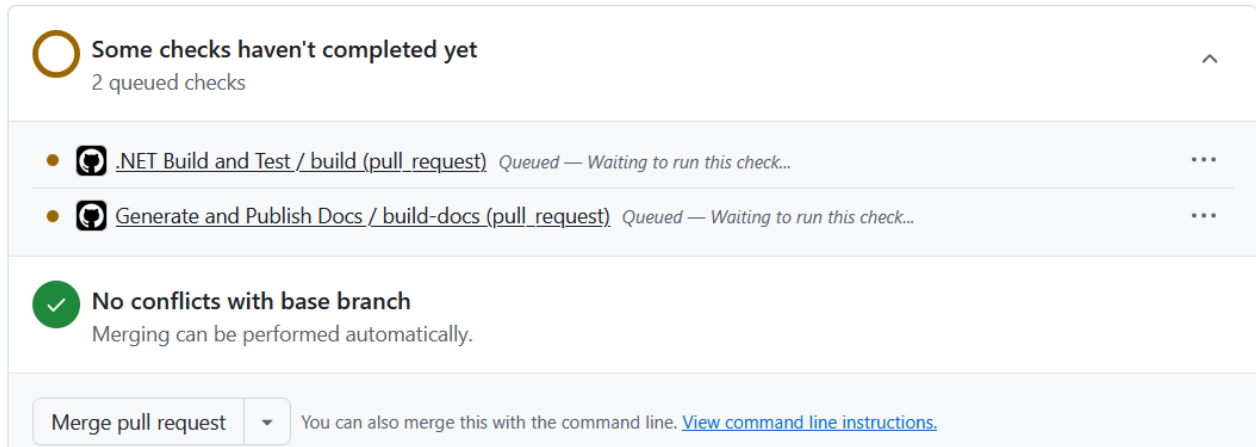


Для генерации документации будет использован отдельный Action со следующим кодом:

После добавления тега и ветки Release, он был успешно добавлен:



Запуск проверок при Pull-Request:



Упс.



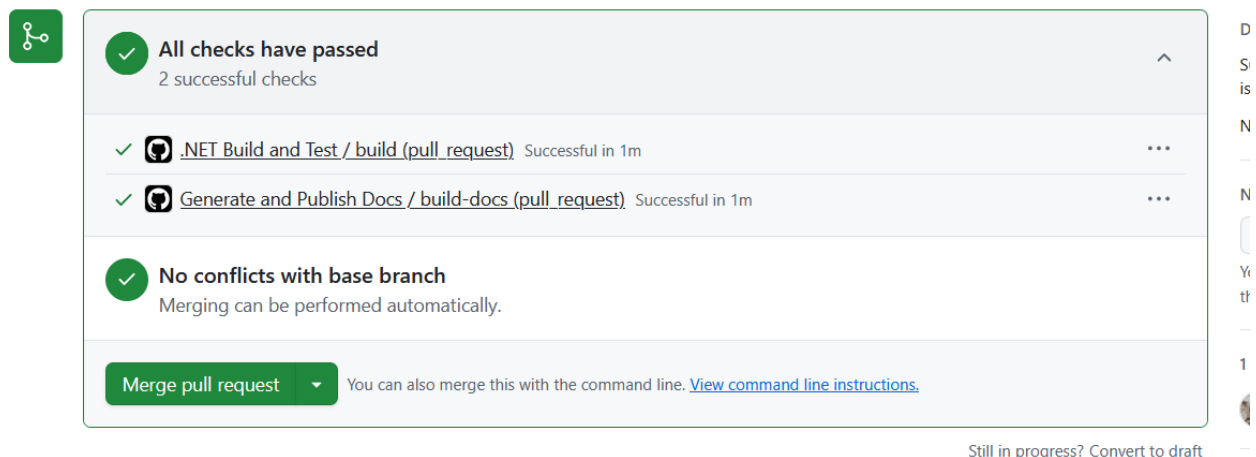
```

1 ▶ Run docfx docfx.json
6
6 FileNotFoundException: Cannot find config file D:\a\TRPIS_PZ\TRPIS_PZ\docfx.json
7   at (DocfxConfig , string configDirectory) GetConfig(string configFile) in
8     Docset.cs:73
9   at void <Execute>b__0() in DefaultCommand.cs:39
10  at int Run(LogOptions options, Action run) in CommandHelper.cs:48
11  at int Execute(CommandContext context, Options options, CancellationToken
12    cancellationToken) in DefaultCommand.cs:31
13  at int Execute(CommandContext context, TSettings settings) in
14    CancellableCommandBase.cs:24
15  at Task<int> Execute(CommandContext context, CommandSettings settings) in
16    CommandOfT.cs:40
17  at async Task<int> Execute(CommandTree leaf, CommandTree tree, CommandContext
18    context, ITypeResolver resolver, IConfiguration configuration) in
19    CommandExecutor.cs:259
20 Error: Process completed with exit code 1.

```

Для генерации документа надо было ещё дать Write разрешение, кроме самой настройки docfx.

В конечном итоге успех:



The screenshot shows a GitHub Actions workflow status panel. At the top, a green checkmark icon is next to the text "All checks have passed" and "2 successful checks". Below this, two workflow runs are listed, both with green checkmarks and "Successful" status:

- .NET Build and Test / build (pull request) Successful in 1m
- Generate and Publish Docs / build-docs (pull request) Successful in 1m

Below the workflow runs, another green checkmark icon is next to the text "No conflicts with base branch" and "Merging can be performed automatically." At the bottom, there is a green button labeled "Merge pull request" and a link to "View command line instructions." On the right side of the panel, there is a vertical sidebar with the text "D", "S", "is", "N", "N", "Y", "t", "1".

Появилась новая ссылка:

**Deployments** 1

[github-pages](#)

Которая содержит в себе следующее:

## This is the HOMEPAGE.

Refer to [Markdown](#) for how to write markdown files.

## Quick Start Notes:

1. Add images to the *images* folder if the file is referencing an image.

Наконец, после настройки docfx:

The screenshot shows the TRPIS API documentation page for Class1. The page is titled "Class Class1" and is located in the namespace "trpis2". It is the main class of the assembly "trpis2.dll". The page includes a sidebar with a search filter and a list of classes. The main content area shows the class definition, inheritance, inherited members, and methods. The class is defined as a public class. The inheritance section shows that Class1 inherits from object. The inherited members section lists several methods from the object class. The methods section shows the Divide method, which is a static method that takes two integers as parameters and returns an integer.

API / trpis2

### Class Class1

Namespace: [trpis2](#)  
Assembly: trpis2.dll

Главный Класс

```
public class Class1
```

**Inheritance**  
[object](#) ← Class1

**Inherited Members**  
[object.ToString\(\)](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.ReferenceEquals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#), [object.MemberwiseClone\(\)](#)

### Methods

#### Divide(int, int)

Уделение...

```
public static int Divide(int a, int b)
```

**Parameters**  
**a** [int](#)  
**b** [int](#)

## Generation()

</>

Просто текст

```
public static string Generation()
```

Returns

[string](#)

## Main(string[])

</>

Главная функция!

```
public static void Main(string[] args)
```

Parameters

**args** [string](#)[]

## Sum(int, int)

</>

Сумма!

```
public static int Sum(int a, int b)
```

Parameters

**a** [int](#)

**b** [int](#)

Returns

[int](#)

Изменение:

## Generation() #

Этот текст возвращает просьбу, чтобы docfx заработал наконец уже

```
public static string Generation()
```

Returns

[string](#)