



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 закоммите изменения в ветвь development.

Убедитесь, что CI успешно запустился после коммита в ветвь development и прошел все тесты. Создайте Pull Request (PR) для слияния 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

TRPIS\_PZ / .github / workflows / `main.yml` in `main`

Edit Preview ⚙ Spaces ↴ 2 ↴ No

```
46
47   jobs:
48
49     build:
50
51       strategy:
52         matrix:
53           configuration: [Debug, Release]
54
55         runs-on: windows-latest # For a list of available runner types, refer to
56                               # https://help.github.com/en/actions/reference/workflow-syntax-for-github-actions#jobsjob_idruns-on
57
58       env:
59         Solution_Name: your-solution-name          # Replace with your solution name, i.e. MyWpfApp.sln.
60         Test_Project_Path: your-test-project-path  # Replace with the path to your test project, i.e. MyWpfApp.Tests\MyWpfApp.Tests.csproj.
61         Wap_Project_Directory: your-wap-project-directory-name # Replace with the Wap project directory relative to the solution, i.e. MyWpfApp.Package.
62         Wap_Project_Path: your-wap-project-path      # Replace with the path to your Wap project, i.e. MyWpfApp.Package\MyWpfApp.Package.wappproj.
63
64       steps:
65         - name: Checkout
66           uses: actions/checkout@v4
67
```

Был выбран Workflow для .NET

И отредактирован до самого минимального вида.

Этот коммит уже стал учитываться:

1 workflow run

Event	Status	Branch	Actor
Create main.yml	now	main	In progress

.NET Build and Test #1: Commit f4c6750 pushed by Dawidan-orig

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;
    else
        return a + b;
}
```

Функция теста:

```
[Test]
Ссылок: 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()

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

В .cspoj-fайле 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	.NET Build and Test #20: Commit b6ac696 pushed by Dawidan-orig	dev	2 minutes ago 1m 41s	...			
✗ Merge branch 'dev' of https://github.com/Dawidan-orig/TRPIS_PZ into dev	.NET Build and Test #19: Commit 2d61312 pushed by Dawidan-orig	dev	15 minutes ago 1m 6s	...			
✗ Update main.yml	.NET Build and Test #18: Commit a3f5912 pushed by Dawidan-orig	dev	15 minutes ago 55s	...			
✗ Update main.yml	.NET Build and Test #17: Commit 1c38605 pushed by Dawidan-orig	dev	17 minutes ago 1m 36s	...			
✗ Update main.yml	.NET Build and Test #16: Commit c9ca987 pushed by Dawidan-orig	dev	19 minutes ago 1m 32s	...			

## Pull Request:

The screenshot shows a GitHub pull request merge interface. At the top, there's a green circular icon with a white checkmark. Below it, two green boxes indicate successful checks: "All checks have passed" (1 successful check) and "No conflicts with base branch" (Merging can be performed automatically). A large green button labeled "Merge pull request" is prominently displayed. To its right, a smaller note says "You can also merge this with the command line. [View command line instructions.](#)". At the bottom right of the merge button area, there's a link "Still in progress? Convert to draft".

The screenshot shows the comment section for a pull request. It includes a user profile picture, a placeholder "Add a comment", a rich text editor toolbar with various icons, and a text input field with placeholder "Add your comment here...". Below the input field, there are two buttons: "Markdown is supported" and "Paste, drop, or click to add files". At the bottom right, there are two buttons: "Close pull request" and a larger green "Comment" button.

The screenshot shows a message indicating the pull request has been merged and closed. It says "Pull request successfully merged and closed" and "You're all set — the dev branch can be safely deleted." There is a "Delete branch" button on the right.

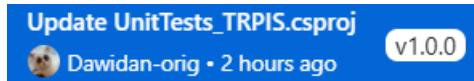
## Теперь работа с веткой Release:

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"
```

Тер отобразился:



Для генерации документации будет использован отдельный Action со следующим кодом:

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

A screenshot of a GitHub release page for version v1.0.1. It shows two download links: "zip" and "tar.gz".

Запуск проверок при Pull-Request:

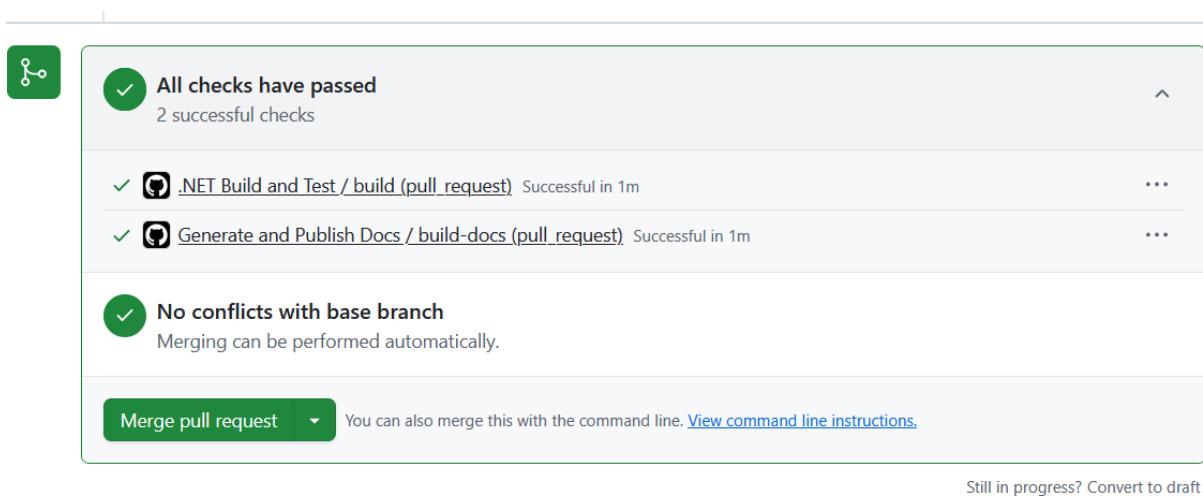
A screenshot of a GitHub pull request review interface. It shows two pending checks: ".NET Build and Test / build (pull\_request)" and "Generate and Publish Docs / build-docs (pull\_request)". Both are listed as "Queued — Waiting to run this check...". Below the checks, there is a green checkmark icon and the text "No conflicts with base branch". At the bottom, there is a "Merge pull request" button with a dropdown arrow and a note: "You can also merge this with the command line. [View command line instructions](#)".

Упс.

```
1 ► Run docfx docfx.json
2
3
4
5
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:



Появилась новая ссылка:

Deployments 1

github-pages

Которая содержит в себе следующее:

# TRPIS\_PZ

## 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 a browser-based API documentation interface. At the top, there's a navigation bar with links for 'API', 'Mathematica', 'Physics', 'Books', 'About & Translation'. Below the navigation is a header with a logo, 'TRPIS', 'Docs', and 'API'.

The main content area has a search bar labeled 'Filter by title' and a breadcrumb trail 'API / trpis2'. The page title is 'Class Class1'. It shows the namespace as 'trpis2' and the assembly as 'trpis2.dll'. A note says 'Главный Класс' (Main Class).

Under the class name, there's a code snippet:

```
public class Class1
```

It also lists inheritance: 'object' ← Class1 and inherited members: object.ToString(), object.Equals(object), object.Equals(object, object), object.ReferenceEquals(object, object), object.GetHashCode(), object.GetType(), object.MemberwiseClone().

A section titled 'Methods' contains the method 'Divide(int, int)'. The description is 'Уделение...' (Division...). The code for the method is:

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

Under 'Parameters', it shows 'a int' and 'b int'.

On the right side of the interface, there are two vertical columns of icons or links labeled 'IN' and 'OUT'. The 'IN' column includes 'Met', 'Divi', 'Gen', 'Mai', and 'Surf'. The 'OUT' column includes '</>'.

## 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](#)