

# SPRAWOZDANIE PROGRAMOWANIE ZAAWANSOWANE II

Adam Zajler

# APLIKACJA ANGULAR

## 1. Inicjalizacja nowego projektu angular z domyślnymi ustawieniami

```
PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular> ng new aplikacja
? Which stylesheet format would you like to use? CSS [ https://developer.mozilla.org/docs/Web/CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
CREATE aplikacja/angular.json (2698 bytes)
CREATE aplikacja/package.json (1078 bytes)
CREATE aplikacja/README.md (1097 bytes)
CREATE aplikacja/tsconfig.json (892 bytes)
CREATE aplikacja/.editorconfig (290 bytes)
CREATE aplikacja/.gitignore (629 bytes)
CREATE aplikacja/tsconfig.app.json (277 bytes)
CREATE aplikacja/tsconfig.spec.json (287 bytes)
CREATE aplikacja/.vscode/extensions.json (134 bytes)
CREATE aplikacja/.vscode/launch.json (490 bytes)
CREATE aplikacja/.vscode/tasks.json (980 bytes)
CREATE aplikacja/src/main.ts (256 bytes)
CREATE aplikacja/src/index.html (308 bytes)
CREATE aplikacja/src/styles.css (81 bytes)
CREATE aplikacja/src/app/app.component.html (20239 bytes)
CREATE aplikacja/src/app/app.component.spec.ts (954 bytes)
CREATE aplikacja/src/app/app.component.ts (318 bytes)
CREATE aplikacja/src/app/app.component.css (0 bytes)
CREATE aplikacja/src/app/app.config.ts (318 bytes)
CREATE aplikacja/src/app/app.routes.ts (80 bytes)
CREATE aplikacja/public/favicon.ico (15086 bytes)
✓ Packages installed successfully.
  Successfully initialized git.
PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular>
```

## Tworzenie komponentu logowania

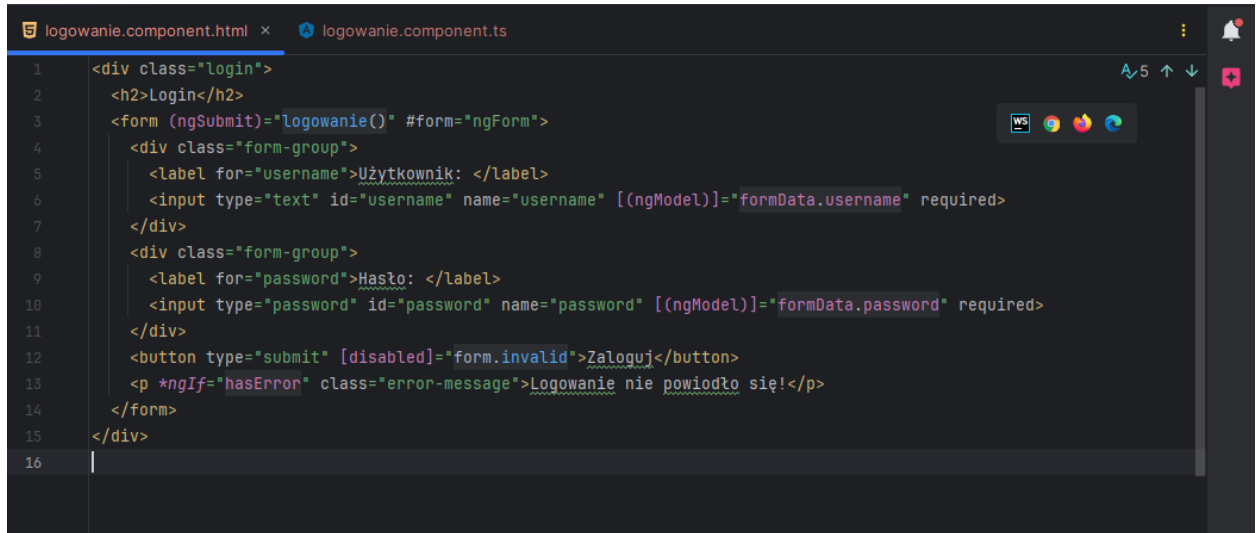
### 1. Inicjalizacja domyślnej templatki dla komponentu

```
PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular\aplikacja> ng generate component logowanie

Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

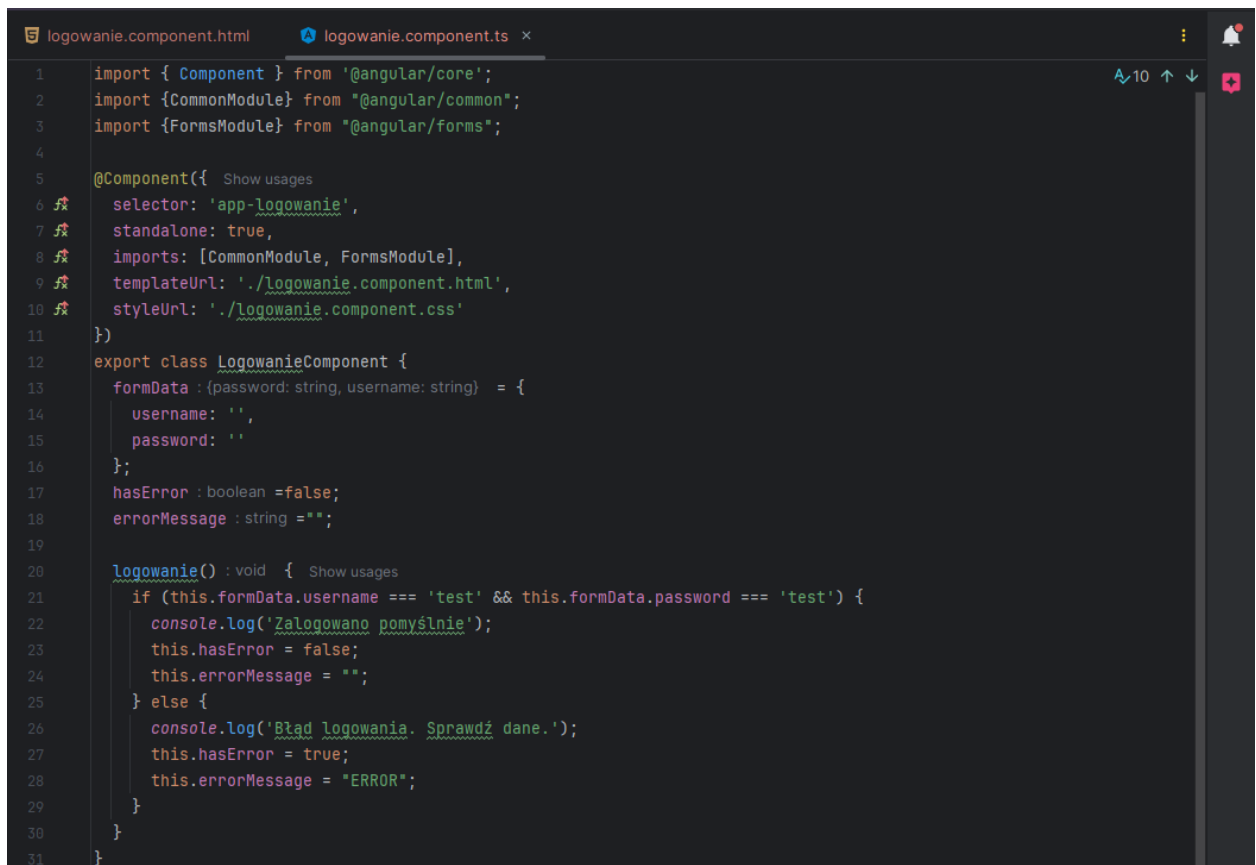
No
Global setting: enabled
Local setting: disabled
Effective status: disabled
CREATE src/app/logowanie/logowanie.component.html (25 bytes)
CREATE src/app/logowanie/logowanie.component.spec.ts (640 bytes)
CREATE src/app/logowanie/logowanie.component.ts (258 bytes)
CREATE src/app/logowanie/logowanie.component.css (0 bytes)
PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular\aplikacja>
```

2. Dodanie domyślnego pliku HTML dla komponentu logowania. Kod zawiera już brakujący paragraf do wyświetlania błędów



```
1 <div class="login">
2   <h2>Login</h2>
3   <form (ngSubmit)="logowanie()" #form="ngForm">
4     <div class="form-group">
5       <label for="username">Użytkownik: </label>
6       <input type="text" id="username" name="username" [(ngModel)]="formData.username" required>
7     </div>
8     <div class="form-group">
9       <label for="password">Hasło: </label>
10      <input type="password" id="password" name="password" [(ngModel)]="formData.password" required>
11    </div>
12    <button type="submit" [disabled]="form.invalid">Zaloguj</button>
13    <p *ngIf="hasError" class="error-message">Logowanie nie powiodło się!</p>
14  </form>
15 </div>
16
```

3. Dodanie kodu logowanie.component.ts. Należało ustawić dodatkowo (lub też nie usuwać domyślnego) ustawienia standalone



```
1 import { Component } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormsModule } from '@angular/forms';
4
5 @Component({ Show usages
6   selector: 'app-logowanie',
7   standalone: true,
8   imports: [CommonModule, FormsModule],
9   templateUrl: './logowanie.component.html',
10  styleUrls: ['./logowanie.component.css']
11 })
12 export class LogowanieComponent {
13   formData : {password: string, username: string} = {
14     username: '',
15     password: ''
16   };
17   hasError : boolean =false;
18   errorMessage : string ="";
19
20   logowanie() : void { Show usages
21     if (this.formData.username === 'test' && this.formData.password === 'test') {
22       console.log('Zalogowano pomyślnie');
23       this.hasError = false;
24       this.errorMessage = "";
25     } else {
26       console.log('Błąd logowania. Sprawdź dane.');
```

4. Zaimportowanie komponentu do app.component aby móc użyć selektora w stronie głównej

```
logowanie.component.html logowanie.component.ts app.component.ts x
1 import { Component } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import { LogowanieComponent } from './Logowanie/logowanie.component';
4
5 @Component({ Show usages Adam Zajler
6   selector: 'app-root',
7   standalone: true,
8   imports: [RouterOutlet, LogowanieComponent],
9   templateUrl: './app.component.html',
10  styleUrls: ['./app.component.css']
11 })
12 export class AppComponent {
13   title: string = 'aplikacja';
14 }
15
```

5. Dodanie selektora formularza na stronę główną

```
app.component.html x
1 <main class="main">
2   <app-logowanie></app-logowanie>
3 </main>
4
5 <router-outlet />
6
```

6. Wynik działania aplikacji dla złych danych logowania (uruchomienie **ng serve**)

**Login**

Użytkownik:

Hasło:

Logowanie nie powiodło się!

Angular is running in development mode. core.mjs:30817

Błąd logowania. Sprawdź logowanie.component.ts:26 dane.

7. Poprawne dane logowania

**Login**

Użytkownik:

Hasło:

Angular is running in development mode. core.mjs:30817

Zalogowano pomyślnie logowanie.component.ts:22

## Testy jednostkowe dla komponentu logowania

1. Dodajemy testy do pliku logowanie.component.spec.ts. Należało poprawić domyślny sposób logowania do aplikacji na odwotywanie się do

**component.formData** oraz na wcześniejszym dodaniu sekcji do wyświetlania błędu

```
1 import { ComponentFixture, TestBed } from '@angular/core/testing';
2 import { LogowanieComponent } from './Logowanie.component';
3
4 describe('LogowanieComponent', specDefinitions: () : void => {
5   let component: LogowanieComponent;
6   let fixture: ComponentFixture<LogowanieComponent>;
7
8   beforeEach(action: () : void => {
9     TestBed.configureTestingModule({
10       imports: [LogowanieComponent]
11     });
12
13     fixture = TestBed.createComponent(LogowanieComponent);
14     component = fixture.componentInstance;
15   });
16
17   it('expectation: 'should create', assertion: () : void => {
18     expect(component).toBeTruthy();
19   });
20
21   it('expectation: 'should have a form with required fields', assertion: () : void => {
22     const usernameField = fixture.debugElement.nativeElement.querySelector('input[name="username"]');
23     const passwordField = fixture.debugElement.nativeElement.querySelector('input[name="password"]');
24     const submitButton = fixture.debugElement.nativeElement.querySelector('button[type="submit"]');
25
26     expect(usernameField).toBeTruthy();
27     expect(passwordField).toBeTruthy();
28     expect(submitButton).toBeTruthy();
29     expect(usernameField.getAttribute('required')).toBe(expected: '');
30     expect(passwordField.getAttribute('required')).toBe(expected: '');
31   });
32
33   it('expectation: 'should call login() method when form is submitted', assertion: () : void => {
34     spyOn(component, method: 'logowanie');
35     const form = fixture.debugElement.nativeElement.querySelector('form');
36     form.dispatchEvent(new Event('submit'));
37     fixture.detectChanges();
38     expect(component.logowanie).toHaveBeenCalled();
39   });
40
41   it('expectation: 'should display an error message for invalid login', assertion: () : void => {
42     component.formData = {
43       username: 'zleDane',
44       password: 'zleDane',
45     }
46
47     component.logowanie();
48     fixture.detectChanges();
49
50     const errorMessage = fixture.debugElement.nativeElement.querySelector('.error-message');
51     expect(errorMessage).toBeTruthy();
52   });
53
54   it('expectation: 'should log in for valid username and password', assertion: () : void => {
55     component.formData = {
56       username: 'test',
57       password: 'test',
```

```

45     }
46
47     component.logowanie();
48     fixture.detectChanges();
49
50     const errorMessage = fixture.debugElement.nativeElement.querySelector('.error-message');
51     expect(errorMessage).toBeTruthy();
52 });
53
54 it('expectation: 'should log in for valid username and password', assertion: () : void => {
55     component.formData = {
56         username: 'test',
57         password: 'test',
58     }
59
60     component.logowanie();
61     fixture.detectChanges();
62
63     const errorMessage = fixture.debugElement.nativeElement.querySelector('.error-message');
64     expect(errorMessage).toBeFalsy();
65 });
66 });

```

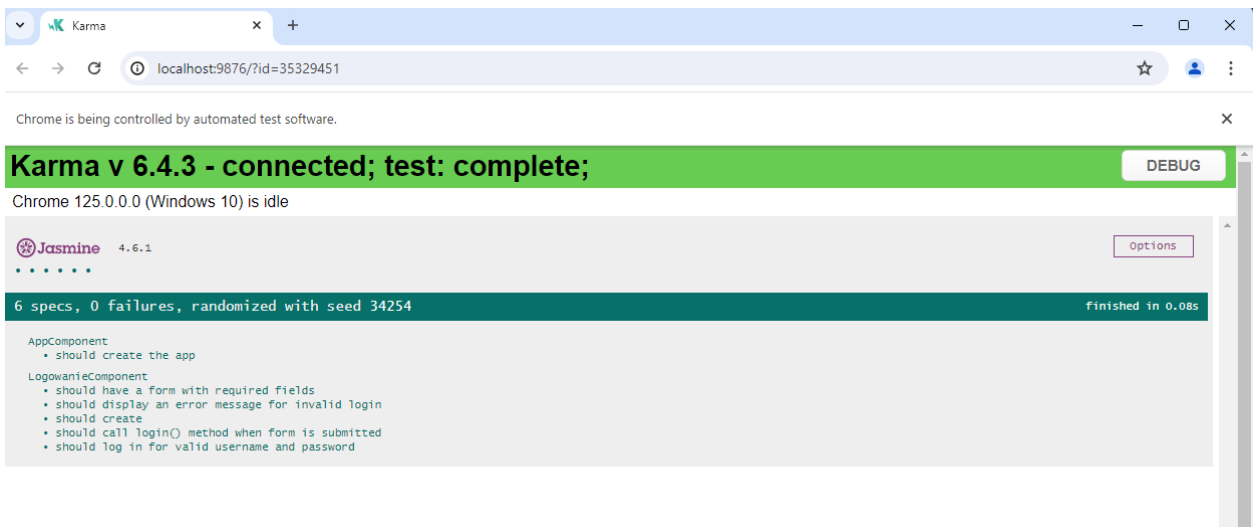
## 2. Uruchomienie testów komendą **ng test** w katalogu projektu

```

PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular\aplikacja> ng test
✓ Browser application bundle generation complete.
01 06 2024 11:36:38.071:WARN [karma]: No captured browser, open http://localhost:9876/
01 06 2024 11:36:38.091:INFO [karma-server]: Karma v6.4.3 server started at http://localhost:9876/
01 06 2024 11:36:38.092:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
01 06 2024 11:36:38.106:INFO [launcher]: Starting browser Chrome
01 06 2024 11:36:38.906:INFO [Chrome 125.0.0.0 (Windows 10)]: Connected on socket 2-LkCWazaJvttD7JAAAB with id 35329451

```


## 3. Wszystkie testy przechodzą pomyślnie



Chrome is being controlled by automated test software.

**Karma v 6.4.3 - connected; test: complete;** DEBUG

Chrome 125.0.0.0 (Windows 10) is idle

 **Jasmine** 4.6.1 Options

6 specs, 0 failures, randomized with seed 34254 finished in 0.08s

- AppComponent
  - should create the app
- LogowanieComponent
  - should have a form with required fields
  - should display an error message for invalid login
  - should create
  - should call login() method when form is submitted
  - should log in for valid username and password

## Tworzenie komponentu dashboardu wraz z testami

### 1. Inicjalizacja domyślnej templatki dla dashboard

```
PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular\aplikacja> ng generate component dashboard
CREATE src/app/dashboard/dashboard.component.html (25 bytes)
CREATE src/app/dashboard/dashboard.component.spec.ts (640 bytes)
CREATE src/app/dashboard/dashboard.component.ts (258 bytes)
CREATE src/app/dashboard/dashboard.component.css (0 bytes)
PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular\aplikacja>
```

### 2. Inicjalizacja serwisu autentyfikacji

```
PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular\aplikacja> ng generate service auth
CREATE src/app/auth.service.spec.ts (363 bytes)
CREATE src/app/auth.service.ts (142 bytes)
```

### 3. Uzupelnienie serwisu metodami logowania, wylogowywania, sprawdzania zalogowania. Oraz dodanie dodatkowo zapisu stanu w localStorage aby później naprawić błąd w testach

```
auth.service.ts x
1  import { DOCUMENT } from '@angular/common';
2  import { Inject, Injectable } from '@angular/core';
3  import { Router } from '@angular/router';
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class AuthService {
9    private localStorage;
10   constructor(@Inject(DOCUMENT) private document: Document, private router: Router) {
11     this.localStorage = document.defaultView?.localStorage;
12   }
13
14   login(): void {
15     if(this.localStorage)
16       this.localStorage.setItem('isLoggedIn', 'true');
17   }
18
19   logout(): void {
20     if(this.localStorage) {
21       this.localStorage.removeItem('isLoggedIn');
22       this.router.navigate(['']);
23     }
24   }
25
26   isLoggedIn(): boolean {
27     if(this.localStorage) {
28       let token: string | null = this.localStorage.getItem('isLoggedIn');
29       return token != null && token.length > 0;
30     }
31     return false;
32   }
33 }
34
35 }
```

### 4. Inicjalizacja domyślnej templatki komponentu menu

```
PS C:\Users\Adam\Desktop\STUDIA\programowanie zaawansowane II\Angular\aplikacja> ng generate component menu
CREATE src/app/menu/menu.component.html (20 bytes)
CREATE src/app/menu/menu.component.spec.ts (605 bytes)
CREATE src/app/menu/menu.component.ts (238 bytes)
CREATE src/app/menu/menu.component.css (0 bytes)
```

5. Uzupełnienie komponentu odpowiednim kodem

```
menu.component.html x
1 <nav *ngIf="isLoggedIn">
2   <ul>
3     <li><a routerLink="/dashboard">Dashboard</a></li>
4     <li><button (click)="logout()">Wyloguj</button></li>
5   </ul>
6 </nav>
7
```

6. Uzupełnienie menu.component.ts aby nadać funkcjonalności komponentowi. Poza domyślnym linkiem do dashboard zostało dodane wylogowywanie z AuthService

```
menu.component.html menu.component.ts x
1 import { Component } from '@angular/core';
2 import { AuthService } from '../auth.service';
3 import { CommonModule } from '@angular/common';
4
5 @Component({ Show usages
6   selector: 'app-menu',
7   standalone: true,
8   imports: [CommonModule],
9   templateUrl: './menu.component.html',
10  styleUrls: ['./menu.component.scss']
11 })
12 export class MenuComponent {
13   public isLoggedIn: boolean = false;
14
15   constructor(private authService: AuthService) { no usages
16     this.isLoggedIn = this.authService.isLoggedIn();
17   }
18
19   logout(): void { Show usages
20     this.authService.logout();
21   }
22 }
```

7. Uzupełnienie funkcjonalności komponentu dashboard.component.ts.

Najważniejsze tutaj jest zabezpieczenie przed wejściem w link dashboardu nie będąc zalogowanym

```
1 import { Component } from '@angular/core';
2 import { MenuComponent } from '../menu/menu.component';
3 import { AuthService } from '../auth.service';
4 import { Router } from '@angular/router';
5 import { CommonModule } from '@angular/common';
6
7 @Component({ Show usages
8   selector: 'app-dashboard',
9   standalone: true,
10  imports: [MenuComponent, CommonModule],
11  templateUrl: './dashboard.component.html',
12  styleUrls: ['./dashboard.component.scss']
13 })
14 export class DashboardComponent {
15   public isLoggedIn: boolean;
16   constructor(private authService: AuthService, private router: Router) { no usages
17     this.isLoggedIn = this.authService.isLoggedIn();
18
19     if(!this.isLoggedIn)
20     {
21       this.router.navigate(['']);
22     }
23   }
24 }
```



## 8. Dodanie menu do strony dashboardu

```
1 <app-menu *ngIf="isLoggedIn"></app-menu>
2
3 <p>dashboard works!</p>
4
```

## 9. Modyfikacja komponentu Logowanie.component.ts aby dodać obsługę AuthService oraz przekierowań z routera

```
1 import { Component } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormsModule } from '@angular/forms';
4 import { AuthService } from '../auth.service';
5 import { Router } from '@angular/router';
6
7 @Component({
8   selector: 'app-logowanie',
9   standalone: true,
10  imports: [CommonModule, FormsModule],
11  templateUrl: './logowanie.component.html',
12  styleUrls: ['./logowanie.component.css']
13 })
14 export class LogowanieComponent {
15   formData: {password: string, username: string} = {
16     username: '',
17     password: ''
18   };
19   hasError: boolean = false;
20   errorMessage: string = '';
21
22   constructor(private auth: AuthService, private router: Router) {}
23
24   logowanie(): void {
25     if (this.formData.username === 'test' && this.formData.password === 'test') {
26       console.log('Zalogowano pomyślnie');
27
28       this.hasError = false;
29       this.errorMessage = '';
30
31       this.auth.login();
32       this.router.navigate(['dashboard']);
33     } else {
34       console.log('Błąd logowania. Sprawdź dane.');
```

## 10. Uzupełnienie głównego pliku routingu o nowe widoki i komponenty (app.router.ts)

```
app.router.ts
1 import { Routes } from '@angular/router';
2 import { DashboardComponent } from '../dashboard/dashboard.component';
3 import { LogowanieComponent } from '../logowanie/logowanie.component';
4
5 export const routes: Routes = [ Show usages new *
6   {
7     path: '',
8     component: LogowanieComponent,
9     data: { title: 'logowanie' }
10  },
11  {
12    path: 'dashboard',
13    component: DashboardComponent,
14    data: { title: 'dashboard' }
15  }
16 ];
```

## 11. Usunięcie komponentu logowania z głównego app.component.html (teraz za wyświetlanie tego komponentu odpowiada router)

```
app.router.ts app.component.html
1 <main class="main">
2 </main>
3
4 <router-outlet />
5
```

## 12. Logowanie przy złych danych

### Login

Uzytkownik:

Hasło:

Logowanie nie powiodło się!

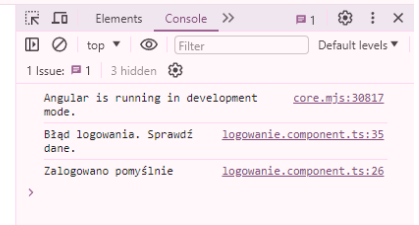
Angular is running in development mode. [core.mjs:30817](#)

Błąd logowania. Sprawdź dane. [logowanie.component.ts:35](#)

13. Poprawne logowanie (przekierowano na /dashboard). Po odświeżeniu zostajemy w tym samym widoku dzięki localStorage.

• Dashboard  
• Wyloguj

dashboard works!



14. Dodanie testów do dashboard.component.spec.ts. Należało poprawić declarations na imports oraz ustawienie w dwóch testach sztucznie component.isLoggedIn = true/false, aby uniknąć opóźnień w przetwarzaniu tej zmiennej.

```
describe('DashboardComponent', specDefinitions: () : void => {
  let component: DashboardComponent;
  let fixture: ComponentFixture<DashboardComponent>;
  let authService: AuthService;

  beforeEach( actions: () : void => {
    TestBed.configureTestingModule( moduleDef: {
      declarations: [],
      providers: [AuthService, provideRouter(routes), { provide: ComponentFixtureAutoDetect, useValue: true }],
      imports: [DashboardComponent, MenuComponent],
    }).compileComponents();

    fixture = TestBed.createComponent(DashboardComponent);
    component = fixture.componentInstance;
    authService = TestBed.inject(AuthService);
  });

  it('expectation: 'should create', assertion: () : void => {
    expect(component).toBeTruthy();
  });

  it('expectation: 'should display the menu after logging in', assertion: async() : Promise<void> => {
    component.isLoggedIn = true; //symulujemy zalogowanie się

    fixture.detectChanges(); // Zaktualizuj widok

    const menu : DebugElement = fixture.debugElement.query(By.css( selector: 'app-menu'));
    expect(menu).toBeTruthy();
  });

  it('expectation: 'should hide the menu after logging out', assertion: async() : Promise<void> => {
    component.isLoggedIn = false; //symulujemy wylogowanie

    fixture.detectChanges(); // Zaktualizuj widok

    const menu : DebugElement = fixture.debugElement.query(By.css( selector: 'app-menu'));
    expect(menu).toBeFalsy();
  });
});
```

Chrome is being controlled by automated test software.

## Karma v 6.4.3 - connected; test: complete;

DEBUG

Chrome 125.0.0.0 (Windows 10) is idle

Jasmine 4.6.1

Options

11 specs, 0 failures, randomized with seed 45087

finished in 0.104s

- LoginComponent
- should call login() method when form is submitted
  - should display an error message for invalid login
  - should create
  - should have a form with required fields
  - should log in for valid username and password

- MenuComponent
- should create

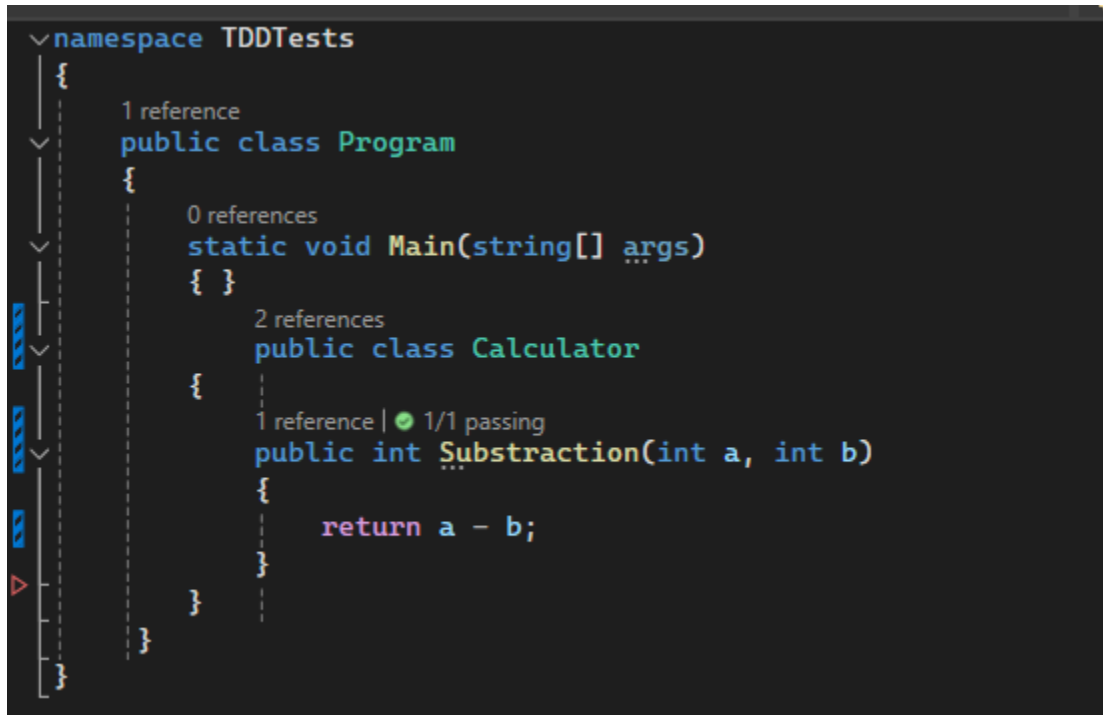
- AuthService
- should be created

- DashboardComponent
- should display the menu after logging in
  - should hide the menu after logging out
  - should create

- AppComponent
- should create the app

# Testy C# TDD

1. Zostały utworzone testy dla prostej klasy kalkulatora z odejmowaniem.
2. Kalkulator:



```
namespace TDDTests
{
    1 reference
    public class Program
    {
        0 references
        static void Main(string[] args)
        { }

        2 references
        public class Calculator
        {
            1 reference | 1/1 passing
            public int Substraction(int a, int b)
            {
                return a - b;
            }
        }
    }
}
```

3. Test dla klasy Calculator:

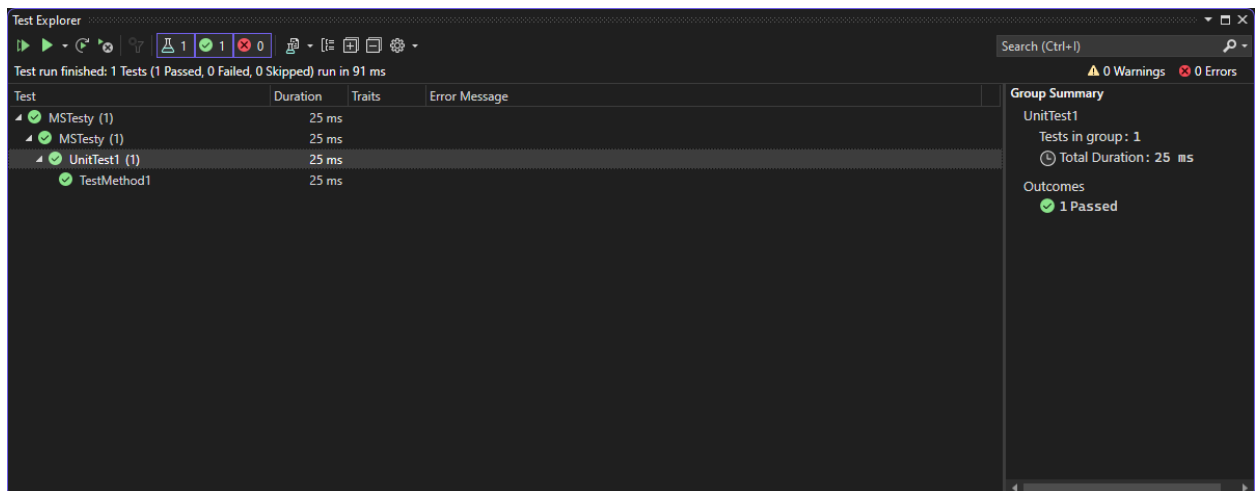
```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TDDTests;
using static TDDTests.Program;

namespace MStesty
{
    [TestClass]
    0 references
    public class UnitTest1
    {
        [TestMethod]
        ✓ | 0 references
        public void TestMethod1()
        {
            Calculator kalkulator = new Calculator();
            int a = 20;
            int b = 10;

            // Act
            int wynik = kalkulator.Substraction(a, b);

            // Assert
            Assert.AreEqual(10, wynik);
        }
    }
}
```

4. Poprawnie przeprowadzony test



The screenshot shows the Test Explorer window in Visual Studio. The test run is complete, showing 1 test passed, 0 failed, and 0 skipped. The test results are as follows:

Test	Duration	Traits	Error Message
✓ MStesty (1)	25 ms		
✓ MStesty (1)	25 ms		
✓ UnitTest1 (1)	25 ms		
✓ TestMethod1	25 ms		

The Group Summary on the right shows:

- UnitTest1
- Tests in group: 1
- Total Duration: 25 ms
- Outcomes: 1 Passed

5. Błędnie przeprowadzony test. Zmieniona zwracana wartość w metodzie

```
namespace TDDTests
{
    1 reference
    public class Program
    {
        0 references
        static void Main(string[] args)
        { }

        2 references
        public class Calculator
        {
            1 reference | 1/1 passing
            public int Substraction(int a, int b)
            {
                return 1;
            }
        }
    }
}
```

Test Explorer

Test run finished: 1 Tests (0 Passed, 1 Failed, 0 Skipped) run in 107 ms

Test	Duration	Traits	Error Message
✖ MStest1 (1)	41 ms		
✖ MStest1 (1)	41 ms		
✖ UnitTest1 (1)	41 ms		
✖ TestMethod1	41 ms		Assert.AreEqual failed. Expected:<10>. Actual:<1>.

Group Summary

UnitTest1

Tests in group: 1

Total Duration: 41 ms

Outcomes

✖ 1 Failed

# Testy C# BDD

1. Utworzenie prostej klasy kalkulatora z metodą odejmowania

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SpecFlowProject1
{
    2 references
    public class Calculator
    {
        1 reference
        public int SubtractNumbers(int number1, int number2)
        {
            return number1 - number2;
        }
    }
}
```



## 2. Utworzenie odpowiednich metod testowania w SpecFlow

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using TechTalk.SpecFlow;
using BDDTesty;
using SpecFlowProject1;
using NUnit.Framework;

namespace BDDTesty.SpecFlowProject1.StepDefinitions
{
    [Binding]
    public class CalculatorStepDefinitions
    {
        Calculator calculator;
        int number1;
        int number2;
        int result;

        [Given(@"the first number is (.*)")]
        public void GivenTheFirstNumberIs(int num1)
        {
            number1 = num1;
        }

        [Given(@"the second number is (.*)")]
        public void GivenTheSecondNumberIs(int num2)
        {
            number2 = num2;
        }

        [When(@"the two numbers are subtracted")]
        public void WhenTheTwoNumbersAreSubtracted()
        {
            calculator = new Calculator();
            result = calculator.SubtractNumbers(number1, number2);
        }

        [Then(@"the result should be (.*)")]
        public void ThenTheResultShouldBe(int p0)
        {
            Assert.AreEqual(p0, result);
        }
    }
}
```

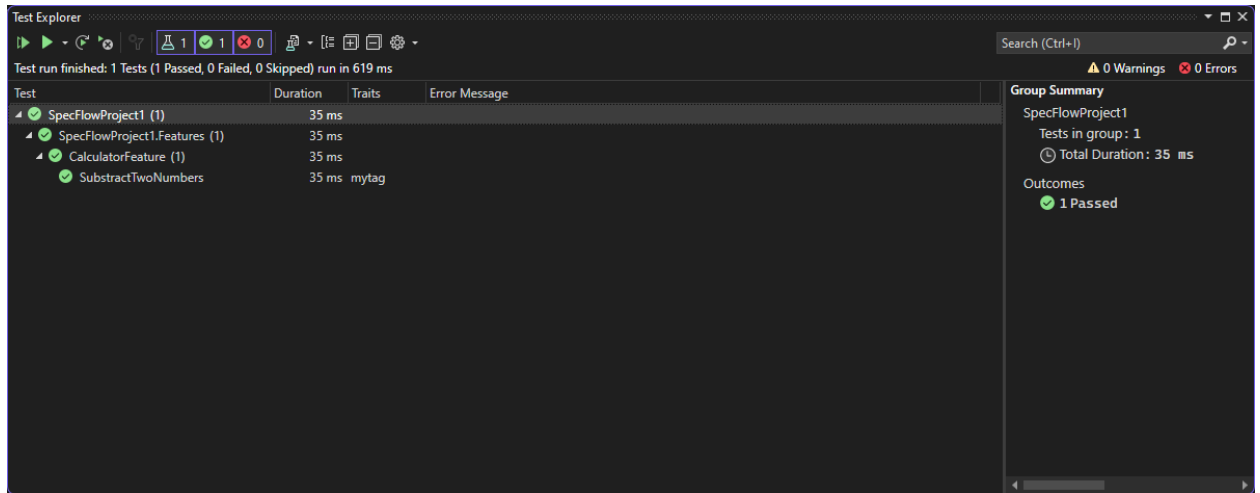
## 3. Utworzenie scenariusza

```
Feature: Calculator
  ![Calculator](https://specflow.org/wp-content/uploads/2020/09/calculator.png)
  Simple calculator for adding **two** numbers

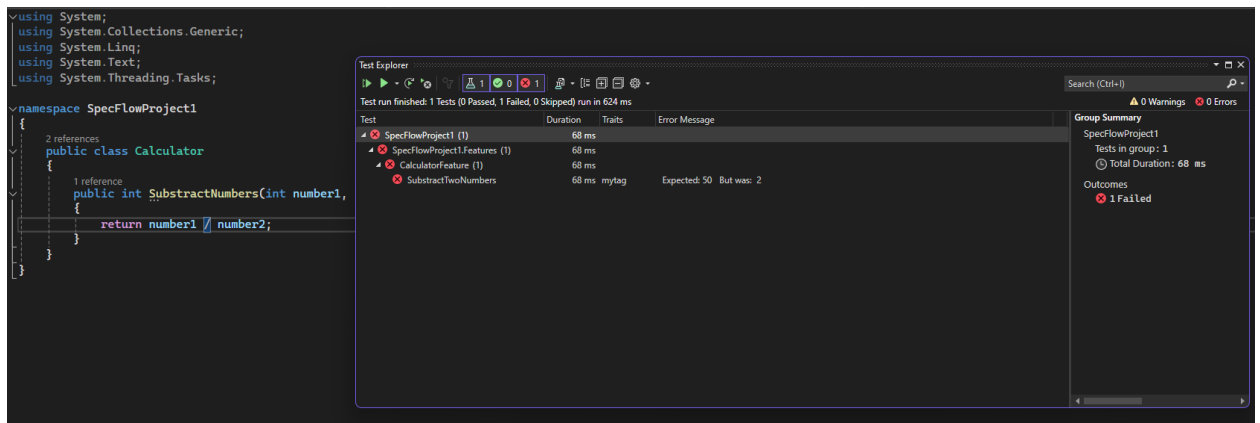
  Link to a feature: [Calculator](SpecFlowProject1/Features/Calculator.feature)
  ***Further read***: **[Learn more about how to generate Living Documentation](https://docs.specflow.org/proj

  @mytag
  Scenario: Subtract two numbers
    Given the first number is 100
    And the second number is 50
    When the two numbers are subtracted
    Then the result should be 50
```

#### 4. Poprawnie przeprowadzony test



#### 5. Źle przeprowadzony test. Funkcja kalkulatora zwraca zły wynik



# Testy C# MOQ1

1. Implementacja sztucznego zewnętrznego serwisu

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZadanieMoq.Tests
{
    4 references
    public interface IWebService
    {
        2 references
        void SendData(string data);
    }
}
```

## 2. Dodanie klasy klasy kalkulatora

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZadanieMoq.Tests
{
    3 references
    public class Calculator
    {
        private IWebService _webService;

        1 reference
        public Calculator(IWebService webService)
        {
            _webService = webService;
        }

        1 reference
        public int Add(int a, int b)
        {
            int result = a + b;
            _webService.SendData($"Add operation: {a} + {b} = {result}");
            return result;
        }
    }
}
```

### 3. Dodanie testu

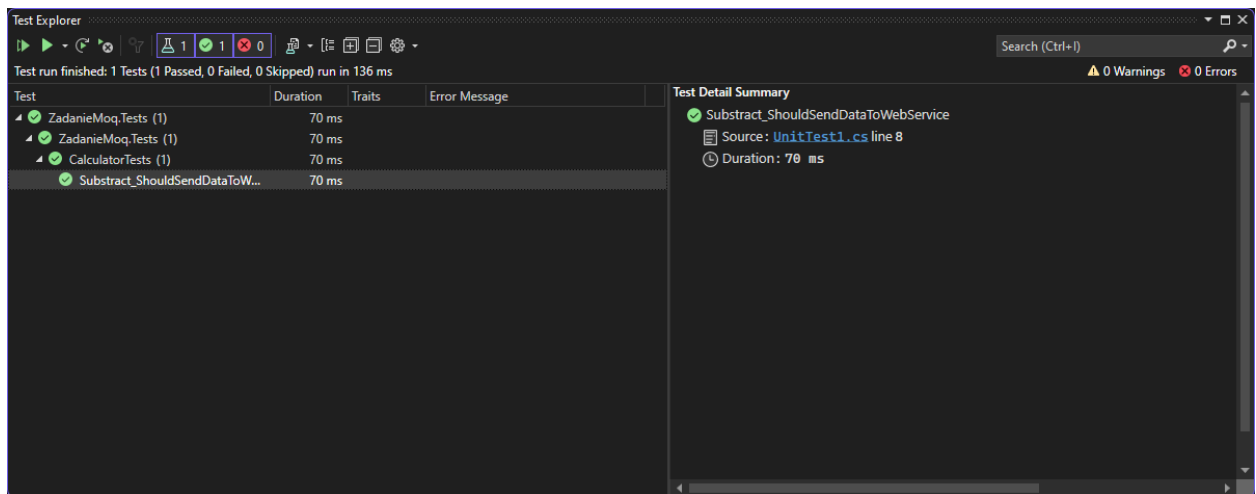
```
using Moq;
namespace ZadanieMoq.Tests
{
    [TestClass]
    0 references
    public class CalculatorTests
    {
        0 references
        public void Add_ShouldSendDataToWebService()
        {
            // Arrange
            Mock<IWebService> webServiceMock = new Mock<IWebService>();
            Calculator calculator = new Calculator(webServiceMock.Object);

            // Act
            int result = calculator.Add(3, 5);

            // Assert
            Assert.AreEqual(8, result);

            // Verify that the SendData method was called with the expected argument
            webServiceMock.Verify(ws => ws.SendData("Add operation: 3 + 5 = 8"), Times.Once);
        }
    }
}
```

### 4. Pozytywny wynik testu



Test Explorer

Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 136 ms

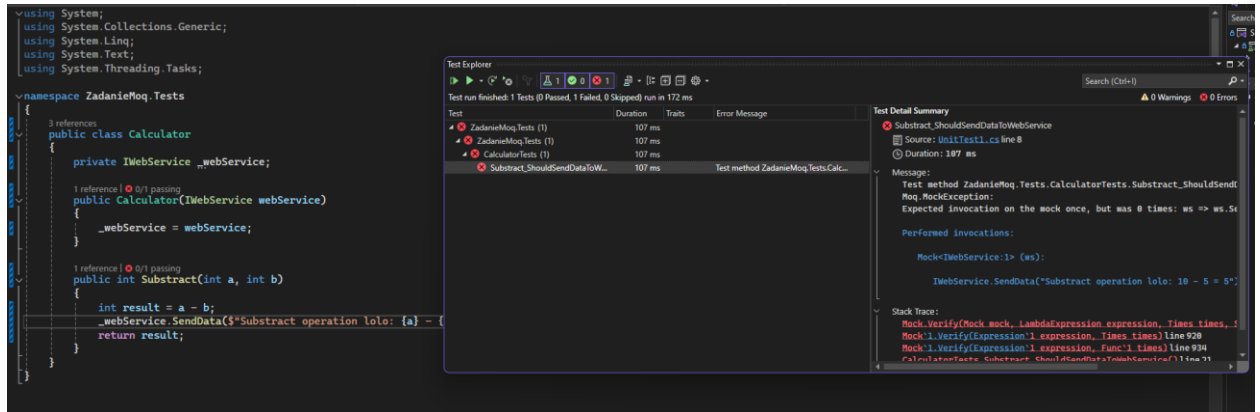
Test	Duration	Traits	Error Message
✓ ZadanieMoq.Tests (1)	70 ms		
✓ ZadanieMoq.Tests (1)	70 ms		
✓ CalculatorTests (1)	70 ms		
✓ Subtract_ShouldSendDataToW...	70 ms		

Test Detail Summary

- ✓ Subtract\_ShouldSendDataToWebService
- Source: UnitTest1.cs line 8
- Duration: 70 ms

0 Warnings 0 Errors

5. Negatywny wynik testu. Zmiana w metodzie odejmowania kalkulatora wysyłanych danych do serwisu



# Testy C# MOQ2

1. Utworzenie klasy OrderProcessor oraz interfejsu OrderService

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZadanieMoq2
{
    6 references
    public interface IOrderService
    {
        4 references | 2/2 passing
        void PlaceOrder(string product, int quantity);
    }

    5 references
    public class OrderProcessor
    {
        private IOrderService _orderService;

        2 references | 2/2 passing
        public OrderProcessor(IOrderService orderService)
        {
            _orderService = orderService;
        }

        3 references | 2/2 passing
        public void ProcessOrder(string product, int quantity)
        {
            // Przetwarzanie zamówienia
            _orderService.PlaceOrder(product, quantity);
        }
    }
}
```

## 2. Utworzenie testu

```
using Moq;
namespace ZadanieMoq2
{
    [TestClass]
    0 references
    public class OrderProcessorTests
    {
        [TestMethod]
        0 references
        public void ProcessOrder_ShouldCallPlaceOrderWithCorrectParameters()
        {
            // Arrange
            Mock<IOrderService> orderServiceMock = new Mock<IOrderService>();
            OrderProcessor orderProcessor = new OrderProcessor(orderServiceMock.Object);

            // Act
            orderProcessor.ProcessOrder("ProductABC", 3);

            // Assert
            // Sprawdzenie, czy metoda PlaceOrder została wywołana dokładnie raz z odpowiednimi parametrami
            orderServiceMock.Verify(os => os.PlaceOrder("ProductABC", 3), Times.Once);
        }

        [TestMethod]
        0 references
        public void ProcessOrder_ShouldCallPlaceOrderMultipleTimes()
        {
            // Arrange
            Mock<IOrderService> orderServiceMock = new Mock<IOrderService>();
            OrderProcessor orderProcessor = new OrderProcessor(orderServiceMock.Object);

            // Act
            orderProcessor.ProcessOrder("Product1", 2);
            orderProcessor.ProcessOrder("Product2", 5);

            // Assert
            // Sprawdzenie, czy metoda PlaceOrder została wywołana dwukrotnie z odpowiednimi parametrami
            orderServiceMock.Verify(os => os.PlaceOrder("Product1", 2), Times.Once);
            orderServiceMock.Verify(os => os.PlaceOrder("Product2", 5), Times.Once);
        }
    }
}
```

## 3. Uruchomienie testu, pozytywny wynik

Test Explorer

Test run finished: 2 Tests (2 Passed, 0 Failed, 0 Skipped) run in 216 ms

Test	Duration	Traits	Error Message
✓ ZadanieMoq2 (2)	114 ms		
✓ ZadanieMoq2 (2)	114 ms		
✓ OrderProcessorTests (2)	114 ms		
✓ ProcessOrder_ShouldCallPlace...	113 ms		
✓ ProcessOrder_ShouldCallPlace...	1 ms		

Test Detail Summary

- ✓ ProcessOrder\_ShouldCallPlaceOrderMultipleTimes
  - Source: [UnitTest1.cs](#) line 23
  - ⌚ Duration: 113 ms

0 Warnings 0 Errors