.NET MAUI

Unit en UI Testing

Waarom testen

Onder andere

- Cross-platform complexiteit
- Platform-afhankelijk gedrag
- Business logica validatie
- UI validatie
- Verhogen van betrouwbaarheid

Unit / UI testing

Unit Testing

- Business logica
- Snel uitvoerbaar
- UI onafhankelijk
- Hoge test-coverage
- Afhankelijkheden mocken

UI Testing

- Gebruikers interacties
- Langzaam uitvoerbaar
- Vereist deployment
- Specifieke tests
- Tegen omgeving

Unit Tests

Demo applicatie

Unit Testing

- xUnit
- Moq (Setup, Returns, Verify, Throws)
- InlineData
- Fact / Theory

Moq

```
using Moq;
using Xunit;

public class UserServiceTests {
    [Fact]
    public void GetUsername_ReturnsUserName_WhenUserExists() {
        // Arrange
        var mockRepo = new Mock<IUserRepository>();
        mockRepo.Setup(r => r.GetUserById(1)).Returns(new User { Name = "Alice"
});
        var service = new UserService(mockRepo.Object);

        // Act
        var result = service.GetUsername(1);

        // Assert
        Assert.Equal("Alice", result);
        mockRepo.Verify(r => r.GetUserById(1), Times.Once);
    }
}
```

Moq

```
public interface IWeatherApiClient {
    Task<int> GetTemperatureAsync(string city);
}

public class WeatherService {
    private readonly IWeatherApiClient _client;

    public WeatherService(IWeatherApiClient client) {
        _client = client;
    }

    public async Task<string> GetWeatherReportAsync(string city)
{       int temp = await _client.GetTemperatureAsync(city);
        return temp > 20 ? "Warm" : "Cold";
    }
}
```

```
using Xunit;
using Moq;
using System.Threading.Tasks;

public class WeatherServiceTests {
    [Fact]
    public async Task GetWeatherReportAsync_ReturnsWarm_WhenTempAbove20() {
        // Arrange
        var mockClient = new Mock<IWeatherApiClient>();
        mockClient.Setup(c => c.GetTemperatureAsync("Amsterdam"))
                  .ReturnsAsync(25);

        var service = new WeatherService(mockClient.Object); // DI via
constructor
        // Act
        var result = await service.GetWeatherReportAsync("Amsterdam");

        // Assert
        Assert.Equal("Warm", result);
        mockClient.Verify(c => c.GetTemperatureAsync("Amsterdam"), Times.Once);
    }
}
```

Fact / Theory

- Geen parameters
- Een enkele run
- Simpele ‘feitelijke’ tests
- Niet uitbreidbaar

```
[Fact]
public void Addition_Works()
{    int result = 2 + 2;
     Assert.Equal(4, result);
}
```

```
[Theory]
[InlineData(2, 2, 4)]
[InlineData(3, 5, 8)]
[InlineData(10, 0, 10)]
public void Addition_Works_WithManyValues(int a, int b, int expected)
{    int result = a + b;
     Assert.Equal(expected, result);
}
```

UI Tests

Demo applicatie