

Synesthesia Web Tests

Structura testelor

```
Synesthesia.Web.Tests/
└── ProfilePageModelTests.cs      # Teste pentru pagina de profil
└── AuthenticationTests.cs        # Teste pentru login/register/logout
└── StudioModelUploadTests.cs     # Teste pentru upload audio (existent)
└── StudioSaveTests.cs            # Teste pentru salvare proiecte
└── ModelTests.cs                 # Teste pentru models și EF Core
└── UnitTest1.cs                  # Template (poate fi șters)
```

Rularea testelor

Metoda 1: Visual Studio

1. Deschide Synesthesia.Web.sln
2. Build solution (Ctrl+Shift+B)
3. Test Explorer (Ctrl+E, T)
4. Run All Tests

Metoda 2: Command Line

```
cd Synesthesia.Web.Tests
dotnet test
```

Metoda 3: Cu Coverage (PowerShell)

```
cd Synesthesia.Web.Tests
.\RunTests.ps1
```

Acest script va:

- Rula toate teste
- Genera raport de coverage
- Deschide raportul HTML în browser

Test Coverage

Current Coverage by File:

- ✓ StudioModel.OnPostUploadAsync - **95%** (6 teste)
- ✓ ProfileModel - **90%** (4 teste + 4 noi)
- ✓ LoginModel - **85%** (3 teste)
- ✓ RegisterModel - **85%** (3 teste)

- ✓LogoutModel - **100%** (2 teste)
- ✓StudioModel.OnPostSaveToProfileAsync - **90%** (7 teste)
- ✓Models (AudioFile, FractalProject, etc.) - **80%** (12 teste)

Target Coverage: 85%+

Test Driven Development (TDD)

Exemplu: Adăugarea unei funcționalități noi

Scenariul: Vrem să adăugăm posibilitatea de a șterge un AudioFile direct (nu doar prin proiect)

Pas 1: Scrie testul ÎNAINTE (RED)

```
[Fact]
public async Task OnPostDeleteAudioAsync_ValidAudio_DeletesSuccessfully()
{
    // Arrange
    var db = GetInMemoryDb();
    var audioFile = new AudioFile
    {
        Id = Guid.NewGuid(),
        UserId = "user-123",
        FileName = "test.mp3",
        FilePath = "/uploads/audio/test.mp3",
        Format = "mp3"
    };
    await db.AudioFiles.AddAsync(audioFile);
    await db.SaveChangesAsync();

    // Act
    var result = await profileModel.OnPostDeleteAudioAsync(audioFile.Id);

    // Assert
    Assert.IsType<RedirectToPageResult>(result);
    Assert.Equal(0, await db.AudioFiles.CountAsync());
}
```

Rulează testul → **VĂ EȘUA** (funcția nu există încă) ✗

Pas 2: Implementeaază funcția (GREEN)

```
// În ProfileModel.cs
public async Task<IActionResult> OnPostDeleteAudioAsync(Guid audioId)
{
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    var audioFile = await _db.AudioFiles
        .FirstOrDefaultAsync(a => a.Id == audioId && a.UserId == userId);

    if (audioFile == null)
    {
```

```

        TempData["Error"] = "Audio not found.";
        return RedirectToPage();
    }

    _db.AudioFiles.Remove(audioFile);
    await _db.SaveChangesAsync();

    TempData["Success"] = "Audio deleted.";
    return RedirectToPage();
}

```

Rulează testul → **VA TRECE** ✓

Pas 3: Refactorizează (REFACTOR)

- Adaugă validări
- Șterge fișierul fizic
- Adaugă logging
- Îmbunătățește testele

Best Practices

1. AAA Pattern (Arrange, Act, Assert)

```

[Fact]
public async Task Example_Test()
{
    // Arrange - Setup
    var db = GetInMemoryDb();
    var model = new ProfileModel(...);

    // Act - Execute
    var result = await model.OnGetAsync();

    // Assert - Verify
    Assert.NotNull(model.Username);
}

```

2. Naming Convention

MethodName_Scenario_ExpectedResult

Exemplu:

- OnPostUploadAsync_ValidMp3_SavesFileAndDbRecord
- OnGetAsync_UserNotAuthenticated_ReturnsEarly

3. Test Isolation

Fiecare test trebuie să fie independent:

```

private ApplicationDbContext GetInMemoryDb()
{
    var options = new DbContextOptionsBuilder<ApplicationDbContext>()
        .UseInMemoryDatabase($"TestDb_{Guid.NewGuid()}"") // Unique DB per
test!
        .Options;
    return new ApplicationDbContext(options);
}

```

4. Cleanup

```

[Fact]
public async Task Test_WithTempFiles()
{
    var tempDir = Path.Combine(Path.GetTempPath(),
Guid.NewGuid().ToString());
    try
    {
        // Test code here
    }
    finally
    {
        if (Directory.Exists(tempDir))
            Directory.Delete(tempDir, true);
    }
}

```

5. Mock Dependencies

```

var userManager = new Mock<UserManager<AppUser>>(...);
userManager.Setup(um => um.GetUserAsync(It.IsAny<ClaimsPrincipal>()))
    .ReturnsAsync(testUser);

```

Debugging Tests

Visual Studio

1. Set breakpoint in test
2. Right-click test → Debug Test
3. Step through (F10, F11)

Console Output

```

[Fact]
public void Test()
{
    var result = CalculateSomething();
    _output.WriteLine($"Result: {result}");
    Assert.Equal(42, result);
}

```

Resurse

- [xUnit Documentation](#)
- [Moq Documentation](#)
- [EF Core Testing](#)
- [ASP.NET Core Testing](#)

Next Steps

1. **Adaugă teste pentru:**
 - ChangePasswordModel
 - IndexModel (Manage/Index)
 - Validări custom
 - Error handling
 2. **Integrare CI/CD:**
 - GitHub Actions
 - Azure DevOps
 - Automatic coverage reports
 3. **Performance tests:**
 - Load testing
 - Stress testing
 - Memory profiling
-

Coverage goal: 85%+