Изпит – Unit тестове с InMemory база данни и Mock обекти (LibraryManagement)

Профил: Приложен програмист

Клас: 12

Тема на изпита:

Тестване на ASP.NET Core MVC приложение „LibraryManagement“, което използва Entity Framework Core за работа с база данни, чрез:
• InMemory база данни (Microsoft.EntityFrameworkCore.InMemory)
• Mock обекти (Moq) за външни зависимости

Описание на готовия проект:

Проектът представлява система за управление на библиотека и съдържа следните основни елементи:
• Модели (Entities): Book, Member, Loan
• DbContext: LibraryDbContext
• Сервизни класове: BookService, MemberService, LoanService
• Интерфейс за уведомления: INotificationService, имплементиран от NotificationService
• ASP.NET Core MVC приложение с минимален HomeController

Задача 1. Създаване/проверка на тестовия проект (5 т.)

1. Уверете се, че съществува NUnit Test Project с име “LibraryManagement.Tests”. Ако липсва – създайте такъв.
2. Проверете, че към тестовия проект е добавена референция към основния проект “LibraryManagement”.
3. Уверете се, че в тестовия проект са инсталирани следните NuGet пакети:
   • Microsoft.EntityFrameworkCore.InMemory
   • Moq
   • NUnit3TestAdapter
   • Microsoft.NET.Test.Sdk

Задача 2. Unit тестове за BookService (10 т.)

Използвайки InMemory база данни, създайте и/или допълнете тестове за класа BookService:

1. Тест, който проверява, че методът ChangeAvailableCopiesAsync връща false, когато книгата не съществува.
2. Тест, който проверява, че ChangeAvailableCopiesAsync не позволява AvailableCopies да стане по-малко от 0 или по-голямо от TotalCopies.
3. Тест, който проверява, че при валидни данни ChangeAvailableCopiesAsync успешно променя броя на AvailableCopies.
4. (По избор) Тест, който проверява, че GetAllAsync връща книгите сортирани по заглавие.

Задача 3. Unit тестове за LoanService с InMemory и Moq (15 т.)

Използвайки InMemory база данни и mock на INotificationService (Moq), създайте тестове за класа LoanService:

1. Тест, който проверява, че CreateLoanAsync връща false, когато членът не е активен (IsActive == false).
2. Тест, който проверява, че CreateLoanAsync връща false, когато няма налични бройки от дадена книга (AvailableCopies == 0).
3. Тест, който проверява, че при успешно CreateLoanAsync:
   • се създава нов запис в таблицата Loans в InMemory базата;
   • AvailableCopies на съответната книга намалява с 1;
   • методът INotificationService.SendAsync се извиква точно веднъж.
4. Тест, който проверява, че ReturnBookAsync връща false, когато заемът (Loan) не е намерен.
5. Тест, който проверява, че при успешно ReturnBookAsync:
   • се задава стойност на ReturnDate;
   • AvailableCopies на съответната книга се увеличава с 1;
   • методът INotificationService.SendAsync се извиква точно веднъж.

Задача 4. Допълнителни тестове (10 т.)

Добавете поне два допълнителни теста по ваш избор за някой от следните случаи:

• Негативни сценарии (невалидни Id, опит за връщане на вече върната книга, липсващи записи и др.).
• Тестове за MemberService – напр. правилна подредба по фамилия и име, създаване/изтриване на членове.
• Гранични случаи при заемане и връщане на книги.
• Други ситуации, които смятате за важни за надеждността на системата.

Оценяване:

Максимален брой точки: 40
30–40 т. – Отличен (6)
24–29 т. – Много добър (5)
18–23 т. – Добър (4)
12–17 т. – Среден (3)
0–11 т. – Слаб (2)

Указания:

• Всеки тест трябва да има говорещо име и да следва структура Arrange–Act–Assert.
• Използвайте отделна InMemory база данни за всеки тест (различно databaseName), за да няма влияние между тестовете.
• При използване на Moq проверявайте броя на извикванията на mock-натите методи (Times.Never, Times.Once и т.н.).
• След приключване на задачите, стартирайте всички тестове и се уверете, че минават успешно.