Testautomatisierung

Unit-Tests

Es sind im Repository-Layer (Data-Access-Layer) zwei Unit-Tests vorhanden, die die Klasse „KategorieMapper“ testen.

Verwendetes Testframework: NUnit – erlaubt nahtlose Integration in Visual Studio Testrunner.

Verwendete NUnit-spezifische Attribute:

- [TestFixture] = Repräsentiert eine Test-Klasse
- [SetUp] = Setup-Methode, die vor jeder Test-Methode einer Test-Klasse durchlaufen wird
- [Test] = Markiert Methoden als Test-Methoden

Das Namensschema der Unit-Tests ist wie folgt:
<Methodenname>_<Eingabe>_<erwartetes Verhalten>

Beispiel:
MapToDomainModel_ValidKategorie_KategorieBisPropertiesSetCorrectly

Die Unit-Tests folgen dem Arrange-Act-Assert-Muster.

- Arrange → Methodenspezifische Testvoraussetzungen werden erfüllt
- Act → Aufruf der getesteten Methode
- Assert → Prüfung, ob das erwartete Verhalten eingetreten ist.

```
//Arrange
var mapper = new KategorieMapper();
var expectedId = _sampleKategorieBiz.KategorieId;
var expectedName = _sampleKategorieBiz.Name;

//Act
var mappedKategorie =
mapper.MapToDomainModel(_sampleKategorie);

//Assert
Assert.AreEqual(expectedId, mappedKategorie.KategorieId);
Assert.AreEqual(expectedName, mappedKategorie.Name);
```

Probleme bei der Erstellung von Unit-Tests in Bezug auf das Projekt “Larrybook”

- Die einzelnen Layers des Backend-Projekts sollten unabhängig voneinander getestet werden.
- Hierfür ist Dependency-Injection notwendig um Mocks der einzelnen Layers zu erstellen.
- Es wurden daher nur die Mapper getestet, da diese nicht von Objekten aus anderen Layers abhängig sind.
- Da die Anwendung als Ergebnis des Hochschulprojektes einen prototypischen Charakter aufweist (erste Gehversuche mit verschiedenen Frameworks), wäre eine ausführliche Testinfrastruktur, wie sie oben beschrieben, ist nur mit unverhältnismäßig hohem Aufwand realisierbar.

Aus diesen Gründen wurden die Unit-Tests auch nicht in ein separates Unit-Test-Projekt verschoben.