Rails Workshop #5

Testing

Testbaren Code schreiben

- Schreibt SOLID Code

  - **Single responsibility:**
    Es sollte nie mehr als einen Grund dafür geben, eine Klasse zu ändern

  - **Open-closed:**
    Module sollten sowohl offen (für Erweiterungen), als auch geschlossen (für Modifikationen) sein.

  - **Liskov substitution:**
    Das liskovsche Substitutionsprinzip (LSP) oder Ersetzbarkeitsprinzip fordert, dass eine Instanz einer abgeleiteten Klasse sich so verhalten muss, dass jemand, der meint, ein Objekt der Basisklasse vor sich zu haben, nicht durch unerwartetes Verhalten überrascht wird, wenn es sich dabei tatsächlich um ein Objekt eines Subtyps handelt.

Testbaren Code schreiben

- Schreibt SOLID Code
  - **Interface segregation:**
    Clients sollten nicht dazu gezwungen werden, von Interfaces abzuhängen, die sie nicht verwenden.
  - **Dependency inversion:**
    Module hoher Ebenen sollten nicht von Modulen niedriger Ebenen abhängen. Beide sollten von Abstraktionen abhängen.
    Abstraktionen sollten nicht von Details abhängen.
    Details sollten von Abstraktionen abhängen.

RSpec

- Test Framework
- DSL um Ruby-Code zu testen
- beinhaltet Mock-Framework rspec-mocks

Testbaren Code schreiben

- Gesetz von Demeter: Objekte sollen nur mit Objekten in ihrer unmittelbaren Umgebung kommunizieren sollen.

rspec-rails

- RSpec Erweiterung um Rails-Spezifische Bestandteile

- erweitert RSpec um Testmöglichkeiten für Controller, Requests, Models, Views, Mailer und Routen.

Fixtures

- Statische Testdaten
- in Rails zu finden unter db/seeds.rb
- nur Ruby-Code um Instanzen von Modellen zu erzeugen

Factory Girl

- Alternative zu Fixtures
- Wesentlich flexibler aber auch komplexer
- stellt eine DSL zur Generierung von Testdaten zur Verfügung.
- Gute Rails Integration durch factory_girl-rails

Faker oder FFaker

- Dient dazu Fake-Daten zu generieren
- Kann Fakes von Namen, E-Mail Adressen, Adressen, Bankverbindungen etc. etc. erzeugen.
- Gut um Random Testdaten zu erzeugen.

Test Doubles, Mock Objekte, Method Stubs

- Test Doubles oder Mock Objekte sind in Tests Platzhalter Objekte, die Abhängigkeiten simulieren.

- Method Stubs sind Methoden von Test Doubles die Aufrufe und Parameter verifizieren können und vorgegebene Testdaten zurückliefern.

- Es gibt Test Doubles und Partial Test Doubles

    - Bei Einsatz von Test Doubles müssen alle benutzten Methoden der Abhängigkeit gestubt werden.

    - Wohingegen bei Partial Test Doubles grundsätzlich die originale Implementierung vorhanden ist und im Bedarfsfall Methoden gestubt werden können.

RSpec best practices

- **Abhängigkeiten, insbesondere Externe, mocken!**
  Auf die ständige Funktion von externen Schnittstellen aber auch von Klassen kann man sich nicht immer verlassen. Damit dies den Test nicht beeinflusst sollten möglichst viele Abhängigkeiten gezockt werden. Einher damit geht ein Geschwindigkeitsgewinn.
  Selbstverständlich müssen die gemockten Klassen separat getestet werden.

- **Lesbare Testbezeichnungen:**
  DrawService::TlnService import_all parses all lotto objects into draw objects
  Durch Verwendung von describe und context Blöcken um Test-Scopes und Testbedinungen zu kapseln.

- **Factory Girl: build statt create**
  Objekte müssen nicht für jeden Test in der Datenbank persistiert werden. Der Geschwindigkeitsgewinn bei Verzicht auf die Persistenz ist enorm.

Capybara

- Akzeptanztest-Framework für Rails
- Runner für RSpec Tests
- Standardmäßige Unterstützung für Rack::Test und Selenium Webdriver
- Simuliert wie echte User mit der Webseite interagieren, durch Tests in einem (Headless) Browser
- Durch Einsatz eines anderen Drivers (z.B.) Poltergeist kann auch JavaScript/AJAX getestet werden.