BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

Datenschutz/-sicherheit

und die Notwendigkeit von Tests

von Kevin Böhme und Rico Ukro

- Fragen und Aufgaben -

Übungsfragen

Was ist ein Unit-Test?

1. Ein Test, der die gesamte Anwendung überprüft.
2. Ein Test, der einzelne Funktionen, Methoden oder Klassen überprüft.
3. Ein Test, der die Benutzeroberfläche überprüft.
4. Ein Test, der die Datenbank überprüft.

Die richtige Antwort ist: **Ein Test, der einzelne Funktionen, Methoden oder Klassen überprüft.** Unit-Tests prüfen kleine, isolierte Teile der Software, um sicherzustellen, dass diese korrekt funktionieren.

Welche der folgenden Punkte gehören zu den Zielen eines Software-Tests?

1. Verbesserung der Performance
2. Sicherstellen, dass eine Software korrekt funktioniert
3. Verhindern, dass die Software benutzt wird
4. Senken der Entwicklungszeit

Die richtigen Antworten sind: **Sicherstellen, dass eine Software korrekt funktioniert** und **Verbesserung der Performance.**

Welche der folgenden Testmethoden wird üblicherweise zuerst in der Entwicklungsphase durchgeführt?

1. Integrationstest
2. Penetrationstest
3. Unit-Test
4. Systemtest

Die richtige Antwort ist: **Unit-Test**. Diese Tests werden früh durchgeführt, um die kleinsten Bausteine der Software zu testen.

Wie unterstützt das Privacy by Design Prinzip die Durchführung von Softwaretests, um Datenschutzrisiken zu minimieren?

Privacy by Design bedeutet, dass Datenschutz schon während der Entwicklung berücksichtigt wird. Softwaretests helfen dabei, Schwachstellen zu identifizieren, bevor die Software in Produktion geht, was das Risiko von Datenschutzverletzungen reduziert.

Erklären Sie das Konzept der testgetriebenen Entwicklung (TDD).

TDD ist eine Methodik, bei der Tests vor dem Schreiben der eigentlichen Funktionalität erstellt werden.

Welche Vorteile hat es im Vergleich zu herkömmlichen Ansätzen?

Der Vorteil ist, dass die Software kontinuierlich gegen die Tests validiert wird, was zu weniger Fehlern und besserer Codequalität führt.

Welche Art von Test zielt darauf ab, zufällige Eingaben zu verwenden, um Schwachstellen in der Software zu finden?

1. Unit-Test
2. Fuzz-Test
3. Regressionstest
4. Systemtest

Die richtige Antwort ist: **Fuzz-Test**. Er verwendet zufällige Eingaben, um Schwachstellen oder unerwartetes Verhalten in der Software zu entdecken.

Was ist "Mocking" im Kontext von Softwaretests?

Mocking ist das Simulieren von Abhängigkeiten wie Datenbanken oder APIs, um zu testen, wie die zu testende Funktion damit interagiert.

Warum wird es oft in Unit-Tests verwendet?

Es wird in Unit-Tests verwendet, um unabhängige und isolierte Tests durchzuführen.

Warum ist es wichtig, Testdaten zu anonymisieren oder zu pseudonymisieren?

Testdaten sollten anonymisiert oder pseudonymisiert werden, um sicherzustellen, dass keine echten personenbezogenen Daten ungeschützt in Testumgebungen verwendet werden. Dies minimiert das Risiko von Datenschutzverletzungen und entspricht den Anforderungen der DSGVO.

Welcher Test wird üblicherweise durchgeführt, um sicherzustellen, dass neue Codeänderungen keine Fehler in bestehender Funktionalität einführen?

1. Smoke-Test
2. Regressionstest
3. Penetrationstest
4. Performance-Test

Die richtige Antwort ist: **Regressionstest**. Diese Tests prüfen, ob bestehende Funktionen nach Codeänderungen weiterhin korrekt funktionieren.

Aufgaben

Aufgabe 1 - Unit tests - Schaltjahr

Ihr Kunde „Karl“ hat für sein Produkt „Karls Karlender“, bei Ihnen angefragt, die fehlende Implementierung für die Berechnung der Schaltjahre zu übernehmen:

1. Entwickeln Sie Unit-Tests für eine Funktion `is_leap_year(year: int) -> bool`, die überprüft, ob ein gegebenes Jahr ein Schaltjahr ist
2. Erstellen Sie mindestens 5 Testfälle, die verschiedene Szenarien abdecken (z.B. reguläre Jahre, Schaltjahre, Grenzfälle).
   Hinweise: Nutzen Sie die Bibliothek `unittest`
3. Implementieren Sie die Funktion `is_leap_year(year: int) -> bool`

Negative Kalenderjahre sind nicht vorgesehen.

Aufgabe 1 - Unit tests - Schaltjahr

Als kleine Starthilfe:

- *Template*: `test_leapyear.py`
- *Template*: `leapyear.py`

Aufgabe 2 - Unit/Integration tests - Sockenversand

Für einen renommierten Online-Sockenversand, dessen CEO Mark Sockerberg ist, soll die bisherige stark veraltete und fehleranfällige MS-SQL-Server Implementierung durch ein modernes Python Backend ersetzt werden.

Ihre Aufgabe ist es:

1. Unit-Tests für die Funktionalität des Sockenversands zu entwickeln
2. Implementierung der Funktionalität des Sockenversands zu erstellen
3. Integrationstests für die Funktionalität des Sockenversands zu entwickeln

Aufgabe 2 - Unit/Integration tests - Sockenversand

Anforderungen:

- Klasse `SockStore`: Verwaltet den Bestand an Socken und stellt die folgenden Schnittstellen bereit
- Methode `search(self, color: str) -> int`: Gibt die Anzahl der Socken einer bestimmten Farbe zurück
- Methode `buy_sock(self, color)`: Kauft ein Paar Socken einer bestimmten Farbe und gibt die gekaufte Farbe zurück
- Methode `add_sock(self, color: str, quantity: int)`: Fügt Socken einer bestimmten Farbe und Menge hinzu

Aufgabe 2 - Unit/Integration tests - Sockenversand

Als kleine Starthilfe:

- *Template*: `test_sockstore_unit.py`
- *Template*: `sockstore.py`
- *Template*: `test_sockstore_integration.py`

Aufgabe 3 - System tests - Blog

Einer Ihrer Kunden, die Firma „Bloggify“, hat Sie beauftragt, die System tests für ihre Blogging-Plattform zu entwickeln. Das Plattform-Backend ist mittels REST-API über https://jsonplaceholder.typicode.com/posts erreichbar.

Ihre Aufgabe ist es:

1. System tests für die Blogging-Plattform zu entwickeln
2. Schwachstellen durch System tests in der Blogging-Plattform zu finden, um diese dem Kunden zu melden

Aufgabe 3 - System tests - Blog

Guide für die API:

- https://jsonplaceholder.typicode.com/guide

Als kleine Starthilfe:

- *Template*: `test_blog_system.py`

Aufgabe 4 - Unit tests - Krypto-Sicherheit

Im Chef hat Sie beauftragt, den Zufallsgenerator, den der MI-Student „F. Triplequestion“ entwickelt hat, zu validieren und zu verifizieren, da dieser in einem Kundenprojekt eingesetzt werden soll.

Ihre Aufgabe ist es:

1. Entwickeln Sie Unit Tests, um die `Funktion bestRndGenEver() -> string` zu validieren. Diese Methode soll zufällige Zahlen generieren.
2. Notieren Sie die Schwachstellen, die Sie in der Funktion durch Ihre Tests gefunden haben.
3. Implementieren Sie die Funktion bei Bedarf neu, um die Schwachstellen zu beheben.

Zur Erinnerung: In der Kryptographie ist es absolut unverzichtbar, dass Pseudo-Zufallsgeneratoren niemals die gleichen Zufallszahlen liefern dürfen.

Aufgabe 4 - Unit tests - Krypto-Sicherheit

Als kleine Starthilfe:

- *Zufallsgenerator*: `bestRndGenEver.py`