adesso

Auswahl des Testautomatisierungstools das Projekt „Abcd-ef“ der Firma ABC GmbH

Anastasiia Kem, Test Engineer

Über 7 Jahre Erfahrung im Testen in verschiedenen Domänen

(Web, Embedded, Fintech, Banking) in Unternehmen wie

GlobalLogic, Raiffeisen Bank, Hermes Europe.

Also insgesamt fast 15 Jahre in der IT

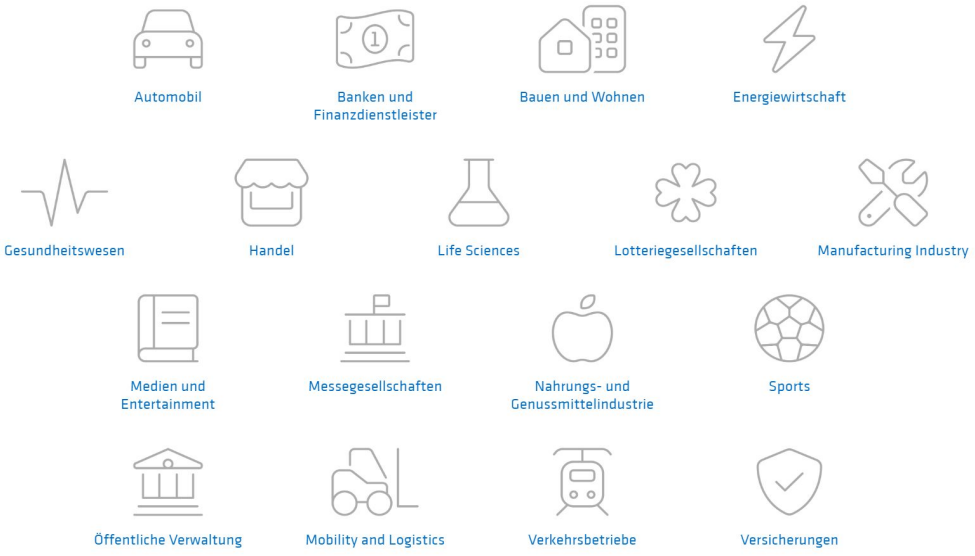
Rund
1,30
Mrd. Euro Group-Umsatz 2024

Über
10.700
Mitarbeitende in der adesso Group

Über
65
Standorte in der adesso Group

Mit einem Team von über 10.700 Mitarbeiterinnen und Mitarbeitern arbeiten wir an mehr als 60 Standorten innerhalb der adesso Group als einer der führenden IT-Dienstleister im deutschsprachigen Raum täglich daran, die Vorhaben unserer Kunden erfolgreich ans Ziel zu bringen.

Maßgeschneiderte Lösungen für:



adesso

Inhalt

1. **Produktanalyse**
Definition der Domäne, technischen Merkmale und Projektphase
Vorhandene Anforderungen und Dokumentation (Requirements, Spezifikationen, Testarten)
Identifikation von Problemstellen, Zielen und Zeitrahmen

2. **Grundlagen**
Testpyramide. Abdeckung auf API-Ebene.
Kostenpflichtige und Open-Source-Lösungen für UI-Tests

3. **Auswahl der Programmiersprache**
Backend vs. Frontend Programmiersprache

4. **Überblick über UI-Testautomatisierungstools**
Selenium, Playwright, Cypress. Analyse der führenden Frameworks

5. **Die Bedeutung von Testbarkeit und Messbarkeit**
Verbesserung der Testbarkeit und Integration der Tests in die CI/CD-Pipeline
Aufbau von Quality Gates, Definition von Reports und Metriken

6. **Schlussfolgerungen und Empfehlungen**

adesso

1. Projektanalyse

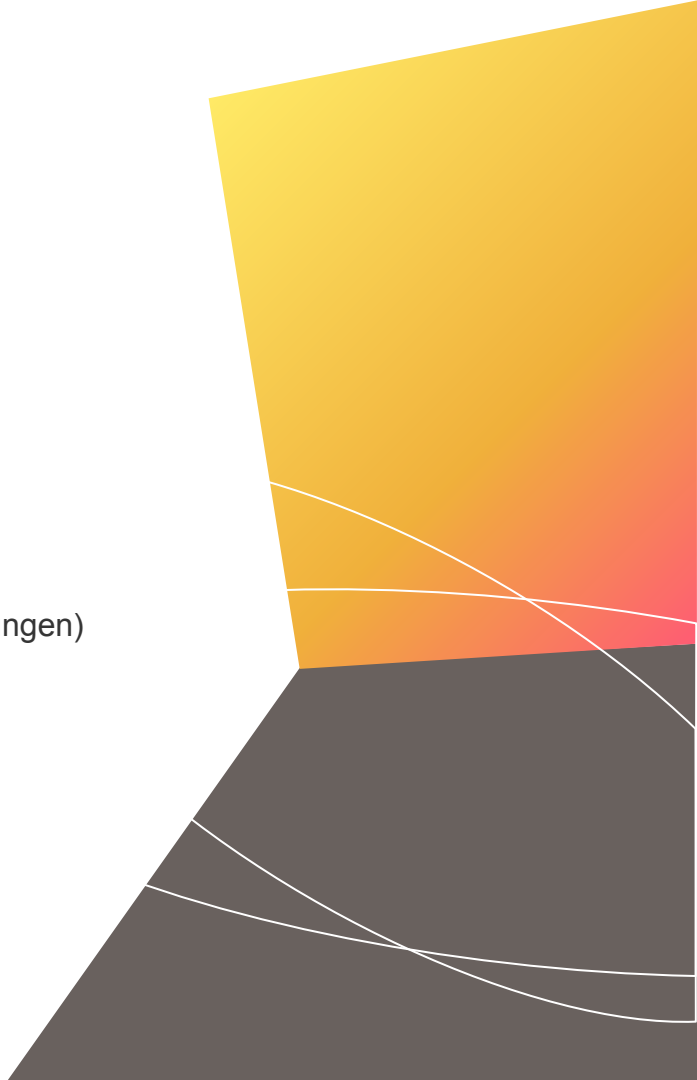
1.1 Definition der Domäne, technischen Merkmale und Projektphase

1.2 Verfügbarkeit von Anforderungen und Projektdokumentation

1.3 Vorhandensein von manuellen und automatisierten Tests

1.4 Identifikation von Problemstellen (z. B. Nutzer Beschwerden, Einschränkungen)

1.5 Definition der Ziele, Zeitrahmen und Budgets



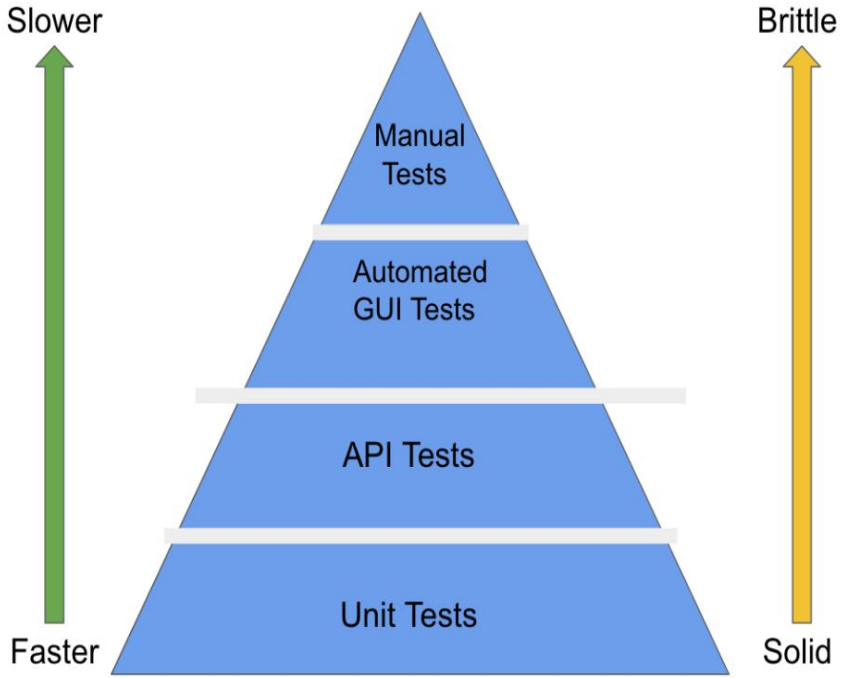
adesso

2. Grundlagen

2.1 Test Pyramide:

Unit Tests – werden von Entwicklern geschrieben, um Code auf niedriger Ebene zu prüfen.
API / Integration Tests – prüfen Datenaustausch zwischen Modulen, Anwendungen oder Services.
GUI Tests – prüfen visuelle Elemente der Anwendung.
Manuelle Tests – händische Überprüfung.

Ziel: Reduktion manueller Tests auf ein Minimum.



2.2 Automatisierte Tests:

API-Testabdeckung:
Halbautomatisch: Postman, SoapUI, JMeter
Frameworks: REST-Assured (Java), RestAssured.Net (C#), Retrofit (Java)
Kommerzielle E2E/ UI-Lösungen:Ranorex, Katalon Studio, TestComplete, etc
Open-Source E2E/ UI-Lösungen:
Selenium WebDriver (Java, C#, Python, JS),
Playwright (Java, C#, Python, JS),
Cypress (JS)

3. Wahl der Programmiersprache

3.1 Backend-Sprache verwenden (z. B. Java, C#, Python)

+ Einfache Integration von API-Tests
+ Wiederverwendung bestehender Klassen möglich
+ Stabil, weit verbreitet, große Community (Java, C#, Python)
+ Große Community von Programmierern und Testern. Es ist einfach, Entwicklers zu finden.

- UI-Tests im Browser langsamer (nicht-native Sprache)
- Höhere Einstiegshürde für Java/C#, **aber kein Problem für unsere Teams :=)**

3.2 Frontend-Sprache verwenden (JavaScript / TypeScript)

+ Native Browser-Sprachen → direkter Zugriff auf Seitenelemente
+ Playwright wurde ursprünglich in TypeScript entwickelt (Microsoft)
+ Schnell, modern, ideal für SPAs

- Kein Typisierungssystem in JavaScript → potenziell chaotischer Code
- Skriptsprache ist langsamer als kompilierte Sprachen wie Java oder C#

4. Überblick über UI-Testtools:

4.1 Selenium WebDriver - Open-Source-Tool zur Automatisierung von UI-Tests in Browsern. Bietet Cross-Browser-Unterstützung (Chrome, Firefox, Safari) und Multi-Language-Kompatibilität.

Funktionsprinzip:
Selenium Emuliert reale Benutzeraktionen (Klicks, Texteingaben, Navigation).
Arbeitet nach dem Client-Server-Modell über den JSON Wire Protocol / W3C WebDriver Standard.

Ablauf:
Testcode ruft Selenium-Methoden auf (z. B. driver.element.click()).
Befehle werden als HTTP/JSON-Anfragen an den Browser-Treiber gesendet.
Browser-Treiber (z. B. chromedriver.exe) übersetzen sie in native Browser-APIs.
Browser führt Aktion aus und sendet das Ergebnis zurück.

Fazit:

- \- Jede Aktion erzeugt einen HTTP-Request → das ist langsamer.
- \- Umfangreicher, etwas schwerfälliger als moderne Alternativen.
- \+ Selenium WebDriver erfüllt ein W3C-Standard.
- \+ Es hat große Community, Dokumentation, Entwicklern
- \+ Es gibt viele Erweiterungen (z. B. Selenide)

adesso

4.2. Playwright - ist ein modernes E2E / UI Test Framework von Microsoft.
Es ist schneller als Selenium, hat umfangreiche integrierte Funktionalität.
Unterstützt (auch wie Selenium) Chrome/Edge, Firefox und WebKit über ein einheitliches API.

Technische Merkmale:
Nutzt WebSocket-Verbindung statt HTTP → schneller Datenaustausch.
Direkter Zugriff auf Browser Prozesse („Out-of-Process“-Architektur).
Unterstützt moderne Web Features (SPA, Shadow DOM, WebSockets).
Bietet Analyse, Videoaufzeichnung, Reporting und viele andere.

Fazit

+ Playwright ist ziemlich schnell, bequem, robust
- Es ist weniger Entwickler als für Selenium,
- Ältere Projekte bleiben oft bei Selenium.

4.3 Cypress - auch bekanntes Framework für E2E-Tests mit anderer Architektur als Selenium und Playwrite.

+ Tests laufen direkt im Browser neben der getesteten Anwendung.
+ Es hat voller und schneller Zugriff auf Browser Objekte - DOM, window document, localStorage), deswegen das schnellste ist.

- Cypress wurde mit JavaScript/TypeScript Geschrieben und läuft nur unter Node.js Ekosystem. Sehr interaktiv, aber nicht universell einsetzbar

5. Testbarkeit und Messbarkeit

Integration der Autotests in die CI/CD-Pipeline ist entscheidend.

Automatisierte Tests ohne Pipeline-Einbindung sind von sehr geringem Nutzen

Quality Gate = Satz automatischer Prüfungen, die sicherstellen, dass nur qualitativ einwandfreier Code in den nächsten Entwicklungsabschnitt gelangt.

Hilft, fehlerhaften Code frühzeitig zu stoppen und Qualität messbar zu machen.

Ebenso wichtig sind die Metriken und Berichte, mit deren Hilfe die Wirksamkeit der Tests bewertet werden kann.

Wichtige Kennzahlen:

Automatisierungsgrad

Prozentsatz der Codeabdeckung durch Autotests

MTTR (Mean Time to Repair)

Defektdichte

Reporting zur Projektstatus Analyse

6. Empfehlungen

Programmiersprache: Java

Framework für API-Tests: REST-Assured

Framework für UI-Tests: Selenium WebDriver (optional mit Selenide)


Vielen Dank für Ihre
Aufmerksamkeit