Java Entwickler / Software Developer (m/w/d)

Bücher eröffnen Welten – wir tun es ihnen gleich. Als Deutschlands größter Sortimentsbuchhändler verknüpfen wir echte Kundennähe mit richtungsweisender Digitalinnovation. Wir denken das Buch als Erlebnis – ob vor Ort, online, als App oder über den e-Reader. Und versammeln rund 4000 Expertinnen und Experten mit einer Mission: Ein Kulturgut auf allen Kanälen erfahrbar zu machen.

Für unsere Zentrale in Münster suchen wir ab sofort einen

Java Entwickler / Software Developer (m/w/d)

Welche Aufgaben erwarten dich:

- Du betreibst die eigenverantwortliche Weiterentwicklung unserer **eCommerce-Plattform** mit Spaß an Software-Engineering und Clean Code in einem agilen Produktteam
- Du betreust als **Full-Stack-Entwickler** die ganze Kette vom Datenbankzugriff/ Serviceaufruf bis zur Gestaltung und Auslieferung des Frontends

- Du sicherst die Qualität durch **Code Reviews**, Unit-/Integrations-/End-To-End-Tests sowie die Auslieferung über **Jenkins** und **Git**
- Du misst anhand von **Kennzahlen** die Produktivität, Performance und Qualität eurer Software, um sie stetig zu verbessern
- Du entwickelst zusammen mit dem Product Owner die beste Lösung für unsere **Kunden**

Was bringst du mit:

- Mehrjährige Berufserfahrung in der Software-Entwicklung mit Java, gerne im Bereich **eCommerce**
- Sehr gute Kenntnisse aktueller Frameworks und Konzepte (z.B. **Spring** MVC/Spring Boot, JPA, Hibernate, REST API, Microservices, Self-contained Systems) und Entwicklungswerkzeuge (Git, Maven, Jenkins)
- Know How im Umgang mit dem **Apache Tomcat**, Apache Webserver, Linux, Docker und Co
- Erfahrungen mit **Datenbanksystemen** (MySQL, Postgres) und **Frontend-Technologien** (HTML5, JavaScript, CSS3)
- DDD, TDD, BDD – die Abkürzungen sind Dir nicht unbekannt und idealerweise hast Du hier bereits praktische Erfahrung gesammelt
- Konzeptionelle Stärke, analytisches Denken, Kommunikationsfähigkeit und **sehr gute Deutschkenntnisse**

Diese Benefits bieten wir dir:

- Ein **motiviertes und selbstorganisiertes Produktteam**, dem ein tolles Miteinander und Freude an der Arbeit wichtig ist
- Ein modernes Arbeitsumfeld und die Möglichkeit neue Technologien oder Vorgehensmodelle einzuführen
- Ein **aktueller Technologie Stack** (Java, Apache Tomcat, Spring, Spring Boot, JPA, Maven, Git, Jenkins, Nexus, Gitlab, Sonar, Jira, Eclipse, IntelliJ, Postgres, MySQL, RabbitMQ, Apache Kafka, Handlebars, Grafana, Graylog)
- Besuch und Mitgestaltung von **Fachkonferenzen, Hackathons** und die Möglichkeit auf dem **Thalia Tech Blog** zu veröffentlichen

- Ein familienfreundliches Unternehmen mit **flexiblen Arbeitszeiten** und 30 Tagen Urlaub im Jahr sowie eine Vielzahl an Sozialleistungen

Haben wir dein Interesse geweckt?

Dann bewirb dich unter Angabe deines Gehaltsrahmens und deines frühestmöglichen Eintrittstermins über unser Online-Portal oder über bewerbungen@thalia.de!

Thalia Bücher GmbH | Deine Ansprechpartnerin: Stefanie Klein

Akzeptanztests bei Thalia

Im Team „Kaufen“ und im Team „Kunde im Mittelpunkt“ setzen wir seit über einem Jahr automatisierte Akzeptanztests zur Sicherung unserer Qualität ein. Beide Teams sind begeistert von diesem Vorgehen. Höchste Zeit für einen Beitrag in unserem Techblog

Was ist ein Akzeptanztest?

Ein Akzeptanztest ist ein funktionaler Test, der das Benutzerverhalten beschreibt, um User Story und Akzeptanzkriterien zu überprüfen.
Dabei steht die Sicht des Benutzers im Vordergrund. Es ist wichtig nicht irgendwas zu testen, sondern genau das, was wir mit einer User Story umsetzen wollen. Durch die Akzeptanztests wird sichergestellt, dass die Software aus Sicht des Benutzers wie gewünscht funktioniert.

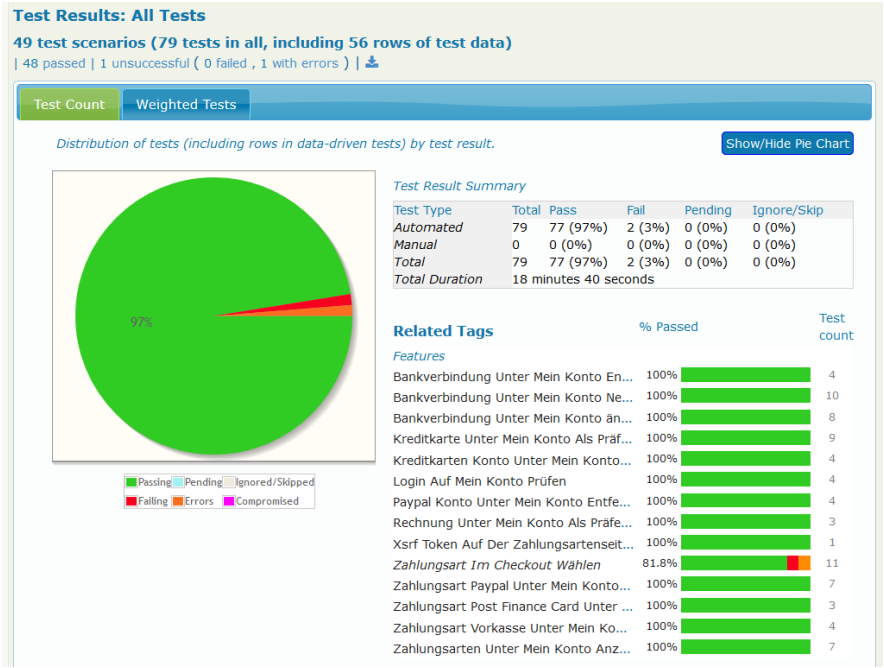
Ein Akzeptanztest besteht aus drei Teilen: Vorbedingungen (Angenommen), Aktionen die durchgeführt werden (Wenn) und erwartete Ergebnisse (Dann). Diese sogenannten Szenarien werden später zu automatisierten Tests auf verschiedenen Ebenen der Testpyramide. Zur Definition der Szenarien verwenden wir in unseren Teams das Framework Cucumber.

Beispiel eines einfachen Szenarios:

```
Szenario: Anzeige der Zahlungsarten unter mein Konto mit Rechnung als präferierte Zahlungsart
  Angenommen es existiert ein angemeldeter Kunde bei Thalia.de
   Angenommen der Kunde hat die präferierte Zahlungsart Rechnung
  Wenn dieser Kunde die Seite Mein Konto Zahlungsarten aufruft
  Dann ist das Akkordeon an der Stelle Rechnung aufgeklappt
```

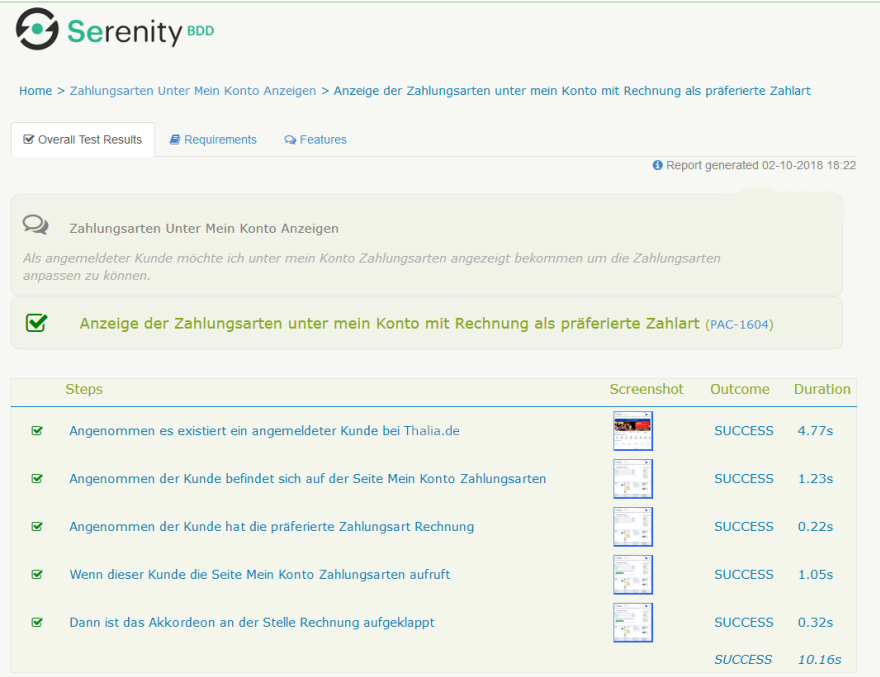
Testreport mittels Serenity

Bei jedem Testdurchlauf wird automatisch mittels Serenity ein Bericht mit den Testergebnissen erstellt.



Serenity Report – Übersicht

Zu jedem Szenario können die einzelnen Testschritte inklusive Screenshots betrachtet werden. Im Fehlerfall sieht man somit direkt, welcher Testschritt fehlgeschlagen ist.



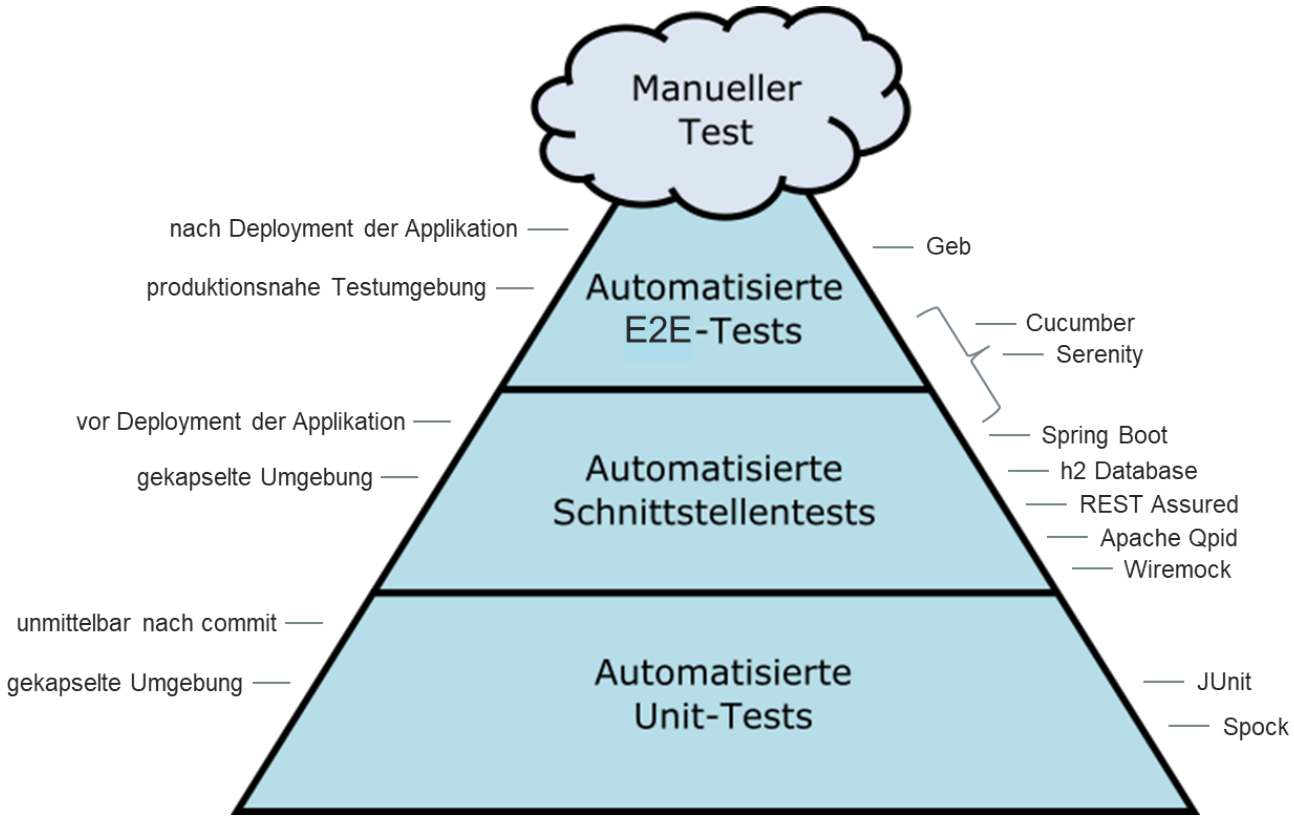
Serenity Report – Übersicht

Zielbild Testpyramide

Unser Zielbild bei der Qualitätssicherung im Team „Kaufen“ orientiert sich an der klassischen Testpyramide. Wir wollen unsere Fehler natürlich möglichst früh und möglichst vor dem End-to-End-Test finden. Da wir allerdings gerade dabei sind unser Frontend abzulösen und unseren Checkout neu zu gestalten, schreiben wir im Moment relativ viele End-to-End-Tests.
Um dem entgegen zu wirken, begeben wir uns als nächstes auf die Suche nach einem geeigneten JavaScript-Test-Framework. Wir erhoffen uns dadurch mögliche Fehler im Frontend bereits auf Unit-Test-Ebene aufzudecken und ggf. sogar einzelne End-to-End-Tests ablösen zu können.

Welche Frameworks verwenden wir und wann werden die Tests ausgeführt?

Die gleichen Szenarien nutzen wir sowohl für die automatisierten End-to-End-Tests, als auch für die Schnittstellentests.



Die automatisierten End-to-End-Tests erfolgen nach dem Deployment in einer produktionsnahen Testumgebung. Dabei verwenden wir das Framework Geb, welches den Selenium WebDriver nutzt und der Browser-Automatisierung dient. Die Tests werden hierbei in der Programmiersprache groovy geschrieben.

Durch die Schnittstellentests soll die jeweilige Applikation inklusive Datenbank- und RabbitMQ-Prüfungen (wo notwendig) zur Laufzeit getestet werden. Um möglichst unabhängig zu sein, erfolgt der Test in einer gekapselten Umgebung. Dazu wird die Applikation mittels h2 Database, Apache Qpid und Spring Boot gestartet, so dass die Schnittstellentests zwar zur Laufzeit , aber noch vor dem Deployment ausgeführt werden können. Fremdsysteme werden dabei mit Wiremock simuliert. REST Assured wird zur Durchführung von HTTP Requests und Validierung der HTTP Response verwendet.

Welche Vorteile bringen uns die Akzeptanztests?

Die Akzeptanztests helfen uns offene Punkte und Ungereimtheiten aufzudecken und mit dem Product Owner zu besprechen.

Aktuell werden im Team „Kaufen“ die Szenarien meistens durch eine Person aus

dem Dev-Team kurz vor oder zeitgleich zur Implementierung geschrieben. Dies kann, muss aber nicht, die QAlerin des Teams sein. Wie beim Code-Review erfolgt auch hier ein Review durch eine zweite Person des Dev-Teams.

Noch besser wäre es die Szenarien bereits vor dem Start der Software-Implementierung zu schreiben. Optimalerweise sogar in der „3 Amigos“ Konstellation, welche aus Product Owner, QAlerin/Analyst und Entwickler besteht, um die Anforderung aus unterschiedlichen Blickwinkeln zu betrachten. Dadurch können Ungereimtheiten und Komplexitätstreiber bereits während der Anforderungsanalyse aufgedeckt und besprochen werden. Bisher fehlte uns im Team „Kaufen“ leider die Zeit / der Vorlauf diese Variante auszuprobieren. Wir sind aber zuversichtlich, dass es dazu eine Gelegenheit geben wird

Im Team „Kunde im Mittelpunkt“ wird bei der Erstellung der Szenarien bereits mit der Product Ownerin und der Fachseite zusammen gearbeitet. Dabei wurden bisher sehr gute Erfahrungen gemacht, da anschließend alle relevanten Personen den gleichen Wissenstand auf einer formalen Ebene haben. Siehe auch hybride Testkonzeption.

Es gibt weitere Vorteile, die uns als Dev-Team zugutekommen.
Wir haben es geschafft die Teamsicht auf unsere Tests zu verbessern. Alle im Dev-Team schreiben Szenarien und implementieren die daraus resultierenden automatisierten Tests. Die Akzeptanztests sind Teil des Repositories der jeweiligen Applikation, werden in der Entwicklungsumgebung geschrieben und bei Bedarf lokal ausgeführt. Durch die Verknüpfung zwischen Fachlichkeit und automatisierten Tests in Form der Szenarien, sind die Auswirkungen bei fehlgeschlagenen Testfällen außerdem direkt sichtbar und helfen uns so bei GoLive-Entscheidungen.

BDD + DDD = eine hybride

Testkonzeption

Das agile Produktteam „KiM“, verwendet in Bezug auf die Qualitätssicherung seiner Produkte zwei wesentliche Ansätze, die wir im Rahmen dieses Blogs kurz skizzieren wollen. Dabei geben wir nicht nur einen konzeptionellen Einblick, sondern stellen auch aus unserer Sicht die Vorteile für die Anwendung eines hybriden Ansatzes heraus.

Behaviour Driven Development

Beim Behaviour Driven Development steht das erwünschte Verhalten der Software aus Sicht des Kunden im Vordergrund. In der Anforderungsanalyse werden die Ziele, Aufgaben und Ergebnisse der Software in Textform als Szenarien festgehalten. Die Szenarien werden unter Verwendung des „Given-When-Then“ – Konzepts erstellt (siehe auch: Akzeptanztests bei Thalia) und dienen als Basis für die Entwicklung automatisierter Tests. Somit wird bei den automatisierten Tests stark die Perspektive des Kunden eingenommen.

Dieser Ansatz bietet nicht nur den Vorteil der einfachen Lesbarkeit, sondern auch die Möglichkeit alle beteiligten Akteure (wie bspw. den Product Owner, die Fachseite, etc.) während der Testfallerstellung mit einzubeziehen und dadurch ein gemeinsames Verständnis vom System bzw. der Software aufzubauen.
Ein mögliches Beispiel für ein Szenario:

```
Szenario: Benutzer meldet sich über die Mobile-App an
Angenommen "Max Mustermann" benutzt die Mobile-App
Wenn "Max Mustermann" sich während des Bestellvorgangs erfolgreich anmeldet
Dann soll die Checkout-Seite angezeigt werden
Und "Max Mustermann" ist in der Mobile-App angemeldet
```

Ein weiteres Element, welches beim BDD-Konzept das gemeinsame Verständnis stärkt, ist der Einsatz einer „ubiquitous language“.
Die „ubiquitous language“ dient als gemeinsam definiertes und genutztes Vokabular, wodurch Fehlinterpretationen bei der Verwendung von Begriffen minimiert werden. Beispielsweise definieren Fachbereiche wie das CRM oder die Softwareentwicklung Begriffe wie „Kunde“ oder „Auftrag“ unterschiedlich. Der

Kunde könnte zum einen als Kunde interpretiert werden, sobald er sich im Webshop eingeloggt hat oder zum anderen wenn er mindestens einen Auftrag durchgeführt hat. Dieses heterogene Verständnis kann zu Problemen bei der Testfallerstellung führen, beispielsweise durch fehlende oder kontroverse Testfälle.

Die eingenommene Perspektive beim BDD-Ansatz offenbart allerdings auch einige Schwächen, die im folgenden beschrieben werden und weshalb sich das Team KiM zu einem Einsatz eines weiteren Konzepts entschieden hat: Dem Domain-Driven Design.

Domain-Driven-Design in der Praxis

Um den eingenommenen Blickwinkel des Endkunden durch den BDD Ansatz im Softwareentwicklungsprozess zu erweitern und mögliche Grenzen zwischen IT und Business aufzuweichen, kombiniert das Team KiM während der Anforderungsanalyse den Behaviour-Driven-Development mit dem Domain-Driven-Design (DDD) Ansatz. Ausgehend vom Business eines Unternehmens, bis hin zu einer fachlichen Domäne (wie bspw. Vertrieb, Buchhaltung, CRM, etc.) bietet das strategic Design des DDD Ansätze zur Strukturierung und Aufteilung der Domäne. Das strategic Design, als einer- von vier Bestandteilen des DDD-Konzeptes, steht bei Team KiM im besonderen Fokus, da sich hier sehr schnell ein Mehrwert generieren lässt. Aufbauend auf der ubiquitous language bietet das strategic Design u.a. folgende Techniken bzw. Konzepte:

- Event Storming
- bounded Context
- Context Map

Event Storming

Nach der Bestimmung der Domäne können anhand des Event Storming die jeweilig relevanten Events innerhalb einer Domäne identifiziert und zugeordnet werden.
Am Beispiel eines sog. „Newsletter“-Projektes“ hat das Team KiM in enger Zusammenarbeit mit Product-Owner und dem Fachbereich die Events in der

Domäne ermittelt und das Ergebnis festgehalten:

Event Storming – Newsletter

bounded Context

Die im Event Storming ermittelten Events lassen sich in bounded Contexts (sog. eigenständige Modelle) clustern. Folgende bounded Contexts wurden festgehalten:

- Kontaktverwaltung
- Registrierung
- Newsletter (An/-Abmeldung)
- Gewinnspiel
- Präferenzen

Der bounded Context (s. Bild unten) bietet leider noch keinen brauchbaren Systemüberblick unter Berücksichtigung des Kommunikations-, Daten- und Modellflusses. Diesen Part übernimmt die Context map, die den Zusammenhang zwischen Model und dem Kontext skizziert.

Context Map

Die Context Map bietet einen guten Ausgangspunkt für die nachfolgende Softwareentwicklung und die Testfallerstellung. Am Beispiel des Newsletter-Projektes hat das Team KiM folgende context map erstellt:

Context Map – Newsletter

Die Domain Events aus dem Event-Storming werden den einzelnen bounded contexts (blaue Post-its) zugeordnet. Darüber hinaus ist der Daten- und Modellfluss anhand von Pfeilen (rot + schwarz) skizziert.
Die Abkürzungen CF, OHS, SUP und CUS stellen sogenannte context map patterns dar, mit der Bedeutung:

- CF = Conformist (Das Downstream Team übernimmt das Modell des Upstream Teams)
- OHS = Open / Host Service (der Host bietet einen vordefinierten Satz an Services an)
- SuP = Supplier (Supplier-Customer i.S.v. Kunde-Lieferantenbeziehung)
- CuS = Customer (Supplier-Customer i.S.v. Kunde-Lieferantenbeziehung)

Die erstellte Context Map bündelt nicht nur die Ergebnisse der hier vorgestellten Techniken, sondern unterstützt zugleich die Testfallentwicklung. Hierbei hilft nicht nur die Visualisierung des Systems, sondern auch die Gruppierung der Events und die Modellzuordnung. Dadurch lassen sich präzisere bzw. eindeutig definierte Testfälle erstellen.

Vorteile

Den hier skizzierten hybriden Ansatz hat das Team KiM bereits erfolgreich in den Softwareentwicklungsprozess integriert und aufgrund folgender Vorteile zu schätzen gelernt:

- Berücksichtigung mehrerer Perspektiven (Business und Endkunde)
- Integration der Testfallentwicklung

- Beteiligung aller relevanten Akteure am Entwicklungsprozess
- Reduzierung von Fehlinterpretationen bei Sachverhalten/Begrifflichkeiten (ubiquitous langauge)
- Reduzierung des Testaufwandes durch Testautomatisierung

Abschließend zu diesem Blog stellen sich dem Team die Fragen:

1. Wie ist eure Meinung zu dem o.g. Vorgehen ?
2. Welche Erfahrungen habt ihr mit dem BDD- bzw dem DDD-Ansatz gemacht?