Titel der Arbeit  
ggf. etwas länger

[1./2. Projektarbeit / Bachelorarbeit / Seminararbeit]

vorgelegt am 30. Januar 2018

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs [Kursbezeichnung]

von

[Vorname Nachname]

|  |  |
| --- | --- |
| Betreuer in der Ausbildungsstätte: | DHBW Stuttgart: |
| [Name des Unternehmens] [Titel, Vorname und Nachname d. Betreuers] [Funktion des Betreuers] [Unterschrift des Betreuers] | [Titel, Vorname und Nachname des wissenschaftl. Betreuers/Prüfers] |

**Inhaltsverzeichnis**

[Abkürzungsverzeichnis (bei Bedarf) III](#_Toc505089007)

[Abbildungsverzeichnis (bei Bedarf) IV](#_Toc505089008)

[Tabellenverzeichnis (bei Bedarf) V](#_Toc505089009)

[1 Einleitung 1](#_Toc505089010)

[2 Hauptteil 2](#_Toc505089011)

[2.1 Eine Abschnittsüberschrift 2](#_Toc505089012)

[2.2 Zweiter Abschnitt 2](#_Toc505089013)

[2.2.1 Ein Abbildungsbeispiel 2](#_Toc505089014)

[2.2.2 Ein Tabellenbeispiel 3](#_Toc505089015)

[3 Nächstes Kapitel 4](#_Toc505089016)

[3.1 Und so weiter 4](#_Toc505089017)

[3.1.1 Und so fort 4](#_Toc505089018)

[3.2 Entsprechend 4](#_Toc505089019)

[3.2.1 Fortgesetzt 4](#_Toc505089020)

[3.2.2 Nochmal fortgesetzt 4](#_Toc505089021)

[Quellenverzeichnis 7](#_Toc505089022)

[Erklärung 8](#_Toc505089023)

# Abkürzungsverzeichnis (bei Bedarf)

ZTB = zentrale Transaktionsbestätigung

maz = meine Allianz

# Abbildungsverzeichnis (bei Bedarf)

[Abb. 1: Eine Abbildung 2](#_Toc505086739)

# Tabellenverzeichnis (bei Bedarf)

[Tab. 1: Eine Beispieltabelle 3](#_Toc505086758)

1. Einleitung

In der Einleitung sind die Problemstellung und die daraus resultierende Zielsetzung der Arbeit exakt zu formulieren und die Vorgehensweise zu begründen. Notwendige Abgrenzungen sind hier vorzunehmen. Anstelle der Überschrift „Einleitung“ kann auch eine aussagekräftigere Überschrift verwendet werden.

1. Problemstellung

Für den reibungslosen Ablauf einer Transaktion ist es zwingend notwendig, Änderungen zu überprüfen und fehlerhafte Eingaben zu erkennen. Eine effektive Vorgehensweise ist die Automatisierung der zu diesem Zweck erstellten Testroutinen. Der Vorteil von automatisierten Tests ist, dass Entwickler ihren gesamten Code automatisch gegen alle Anwendungsfälle testen können und auftretende Fehler direkt sichtbar werden. Die Analyse von Fehlern wird auch erweitert, da auf den ersten Blick ersichtlich wird, welche Tests fehlschlagen. Nach den abgeschlossenen Änderungen kann überprüft werden, ob diese Tests weiterhin fehlschlagen. Wenn dieser Ansatz nicht eingesetzt wird, sondern nur eine spärliche Teststruktur vorhanden ist, sind Entwickler gezwungen nach jeder Änderung umfangreiche manuelle Tests durchzuführen, um die Qualität des Codes sicherzustellen. Des Weiteren wird die Bearbeitung von Fehlern, die von außen ohne konkreten Hinweis gemeldet werden, unnötig erschwert. In diesen Fällen muss jeweils eine ausführliche Fehleranalyse durchgeführt werden, um den Fehler zu finden. Dieses manuelle Testen und umfangreiche Analysieren von Fehlern erhöht die Personal- und Betriebskosten. Dadurch wird ein Teil des Referatsbudgets und der Kapazität der Entwickler blockiert, die ansonsten auf andere Themen verwenden könnten. Aus diesem Grund sinkt die Produktivität des Referates (eine Bezeichnung für eine Allianz interne Unterabteilung) um ein Vielfaches.

Um diese Kosten zu verringern und die Code Qualität zu erhöhen, soll in der zentralen Transaktionsbestätigung eine umfangreiche Teststruktur und der Ansatz des automatisierten Testens realisiert werden.

1. Zentrale Transaktionsbestätigung
   1. Definition der zentralen Transaktionsbestätigung

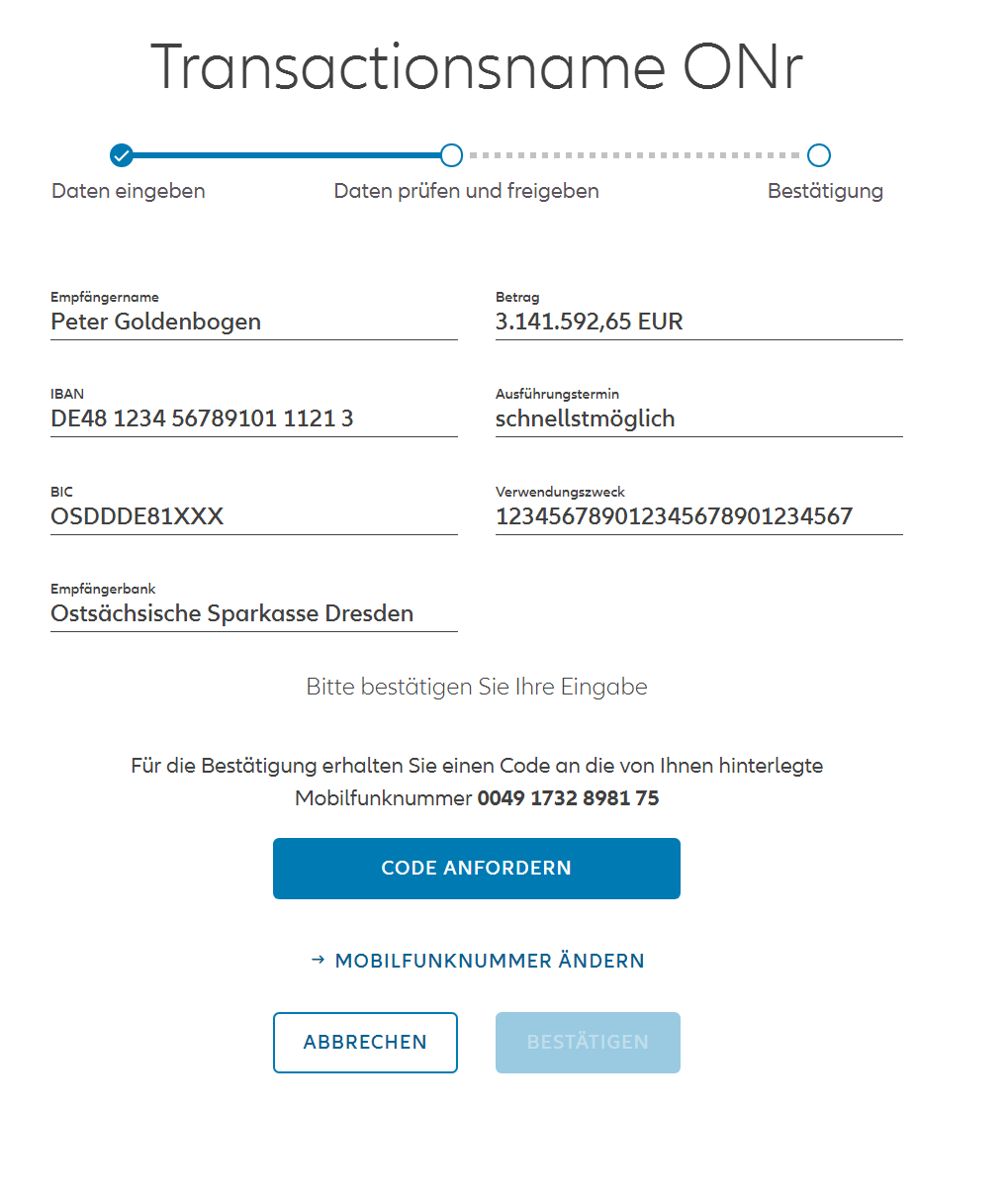
Die Aufgabe der Anwendung „zentrale Transaktionsbestätigung“ auch ZTB genannt, ist die Autorisierung von personenspezifischen Vorgängen durch eine Zwei-Faktor Authentifizierung.

Abb. : ZTB Testansicht

Die Anwendung meine Allianz, in der Nutzer ihre persönlichen Daten und Verträge einsehen und bearbeiten können, ruft im Falle eines autorisierungspflichtigen Vorgangs die Anwendung „zentrale Transaktionsbestätigung auf. Autorisierungspflichtige Vorgänge sind alle, die sicherheitsrelevante Daten, oder generell Änderungen am Vertrag betreffen.

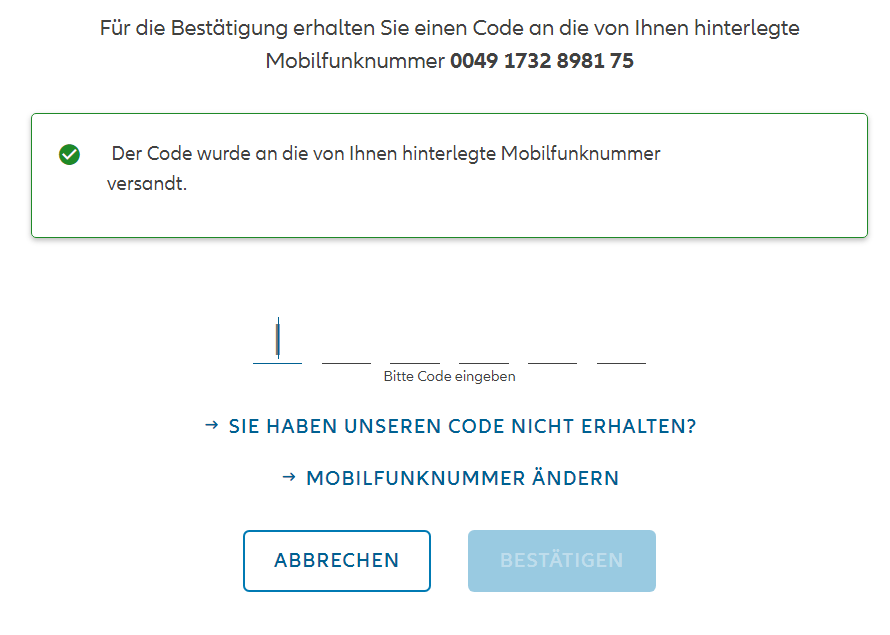
Der zentrale Transaktionsbestätigung, dargestellt in Abb. 1, werden bereits die Kundenspezifischen Daten übermittelt und dort angezeigt. Im Beispielfall heißt der Kunde Peter Goldenbogen. Um den Vorgang fortzusetzen muss der Nutzer einen Code anfordern der dann als mTan an seine eingetragene Mobilfunknummer gesendet wird. Diese mTan wird dann in das Inputfeld (Abb. 2)eingetragen und auf Echtheit überprüft. Wenn die eingegebene Tan validiert wurde wird der Vorgang autorisiert und die Autorisierung wird zurück an „Meine Allianz“ gesendet.

Abb. : ZTB mTan Eingabe

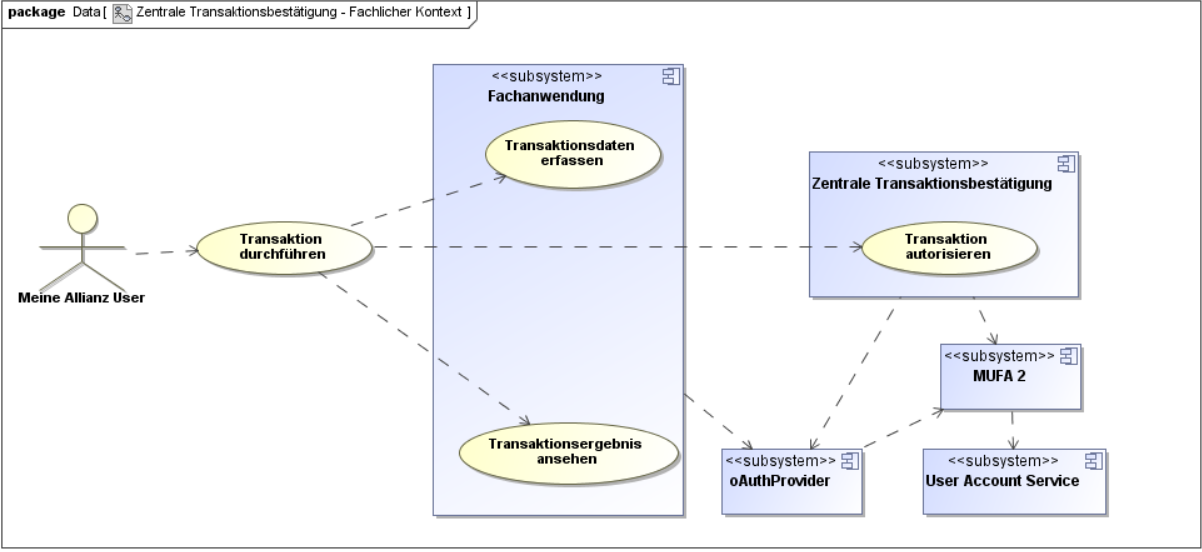


Abb. : Fachlicher Kontext der ZTB

Die technischen Hintergründe werden in dem Anwendungsfalldiagramm in Abb. 3 ausführlicher dargestellt. Der „Meine Allianz User“ ist unser Kunde der eine Transaktion durchführen möchte. Diese Transaktion muss dann wie bereits beschrieben autorisiert werden. Die Zentrale Transaktionsbestätigung überprüft direkt beim Eingang der Daten ob die eingetragene Telefonnummer und die Transaktionsnummer valide ist. Wenn dies nicht der Fall ist wird der Vorgang abgebrochen und der Kunde muss die Daten in der Anwendung „meine Allianz“ nachtragen, ansonsten wird der Vorgang wie beschrieben weiter fortgeführt. Die ZTB generiert die mTan jedoch nicht selbst sondern dies wird durch die Subsyteme Mufa 2 und den oAuthProvider durchgeführt. Nachdem die mTan versandt wurde, hat der User drei Versuche diese richtig in das Eingabefeld einzugeben.

* 1. Der IST-Zustand der End-to-End Tests in der ZTB

Die zentrale Transaktionsbestätigung verfügt aktuell bisher über sehr rudimentäre Tests, die in dem Framework Mocha geschrieben wurden. Dieses Testframework läuft mit Node.js und wurde durch den Java Script Runner von Angular (Karma) ausgeführt. Es ist bereits in der Anwendung ein Mock-Server implementiert, der das Testen ohne reale Testdaten ermöglichen soll. Zu diesem Zeitpunkt muss der Mock-Server jedoch per Hand umgestellt werden, um Fehlerfälle, die sonst nur durch Fehler in der Kommunikation mit anderen Anwendungen entstehen können, darzustellen.

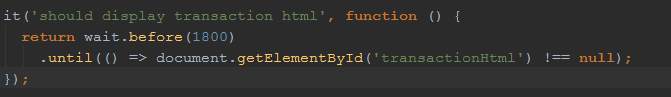


Abb. : Ist Zustand Test 1

Der erste Test aus Abbildung 2 ist sehr simpel und überprüft nur, ob die richtige HTML-Seite geladen wurde. Jedoch ist dieser Test sehr wichtig, denn wenn tatsächlich eine falsche Seite geladen wurde, schlagen alle anderen Tests automatisch ohne erkennbaren Grund fehl.

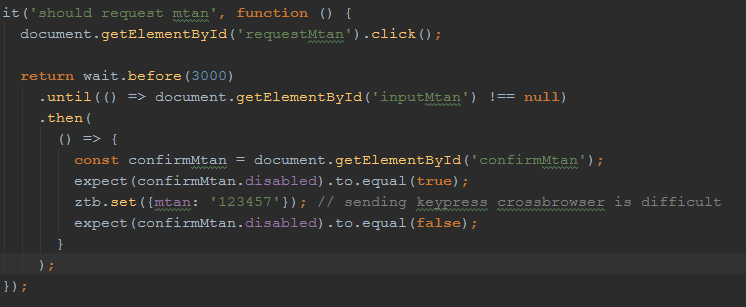


Abb. : Ist Zustand Test 2

Der nächste Test (Abbildung 3) ist etwas komplexer und überprüft, ob bei Betätigung des Buttons „requestmTan“ die Seite richtig angezeigt wird. Des Weiteren wird die Funktion der Eingabefelder und des „Bestätigen“ Buttons rudimentär getestet. Bei fehlerhafter Eingabe, zum Beispiel nicht vollständig ausgefüllte Felder, könnte es zu unerwarteten Fehlern im Backend kommen, da dort andere Eingaben vorausgesetzt werden.

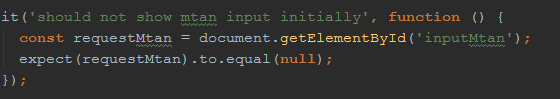


Abb. : Ist Zustand Test 3

Der dritte Test (Abbildung 4) bezieht sich ebenfalls auf die Eingabefelder. In diesem Fall wird überprüft, ob diese direkt beim Aufrufen der Startseite zu sehen sind oder ob sie tatsächlich wie gewünscht erst nach Betätigen des Buttons „requestmTan“ erscheinen.

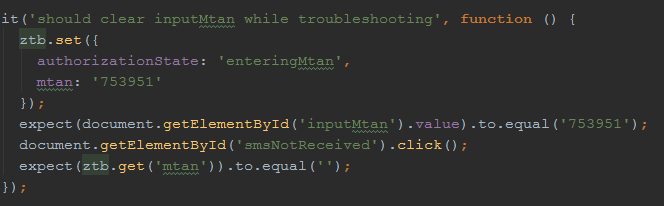


Abb. : Ist Zustand Test 4

Der Test in Abbildung 5 beschreibt das Troubleshooting. Wenn in das Eingabefeld der mTan Zahlen eingegeben wurden und der User auf den Link „smsNotReceived“ klickt, sollte er eine neue SMS bekommen und das Eingabefeld sollte geleert worden sein.

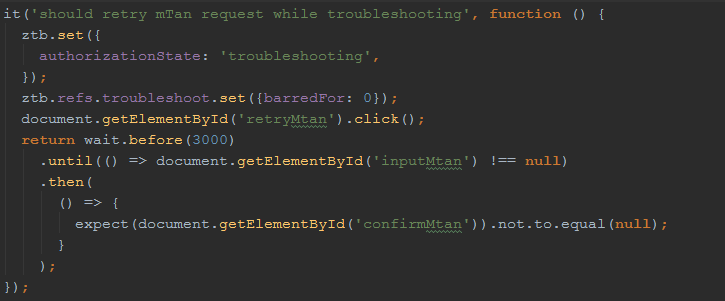


Abb. : Ist Zustand Test 5

In diesem Test (Abbildung 6) wird überprüft, ob der Button, mit dem eine neue mTan anfordert wird, nach Ablauf des Counters, der zu Testzwecken auf 0 gesetzt wird, wieder aktiv ist.

In den beschriebenen Tests ist zu erkennen, dass teilweise wichtige logische Funktionen der Anwendung bereits getestet wurden. Problematisch ist jedoch, dass immer nur einige Teilaspekte getestet wurden und der größte Teil der Anwendung weiterhin ungetestet bleibt.

* 1. Der Soll-Zustand der End-to-End Tests in der ZTB

Die End-to-End Tests in der zentralen Transaktionsbestätigung sollen alle Anwendungsfälle im Frontend-Bereich umfassen und nicht nur bisher einen Bruchteil. Die Testfälle sollen so verlässlich sein, dass die Entwickler davon ausgehen können, dass im Frontend keine Fehler auftreten, wenn alle Tests fehlerfrei durchlaufen werden. Diese Testfälle sollen alle automatisch beim Einchecken des Codes ausgeführt werden, um direkt dem Entwickler als Feedback zu dienen, ob die Anwendung fehlerfrei ist. Die Testfälle sollen alle eindeutig beschrieben sein, um sicherzustellen. dass die Entwickler, wenn ein Test fehlschlägt, direkt erkennen können in welchem Abschnitt des Codes der Fehler auftritt.

Der Wunsch des Referates ist es, dass diese Tests sofern möglich mit einem neuen Framework namens Cypress.io (siehe Kapitel …) geschrieben werden.

1. Tests
   1. Motivation fürs Testen

* Warum
* Nutzen
* CodeQualität
* Veränderung in der Programmiertechnik (Zeitablauf/agil usw)
* TDD
* Fehlersuche
* Zeitaufwand
* Direktes Feedback
* BO Testet
* Aussagekräftige Fehlermeldungen
* Continuous deployment/integration
  1. Testarten/Stages
* Test stages
* Testarten (unittests, regressionstests, Pentests, integrationstest, Useracceptance tests, „End to End tests )
* Pyramide (Optimalfall verteilung)
  1. Art meiner Tests vorgestellt

1. Verwendeten Tools (rein sachlich Entscheidung bei Lösungsansatz)
   1. Cypress.io

* Selenium runs remote commands through the network while Cypress runs in the same run-loop as the application
* Focused on end-to-end testing
* Any front-end framework
* Cypress tests are all written in JavaScript
* Shortcuts
  + Ability to take shortcuts and programmatically log in (cy.request())
* Integration tests

Cypress test runner is open source

Public projects are for free aswell

Private Projects need a license

* 1. Svelte
  2. Weitere
* Npm
* IntelliJ

1. Lösungsansatz
   1. Anforderungen
   2. Erwartete Probleme
   3. Thesenaufstellung
2. Durchführung
   1. Was hab ich gemacht
   2. Was für Probleme hatte ich und wie habe ich sie gelöst
   3. Noch offene Anforderungen
3. Ausblick
4. Zusammenfassung

**Anhang (bei Bedarf)**

[Anhang 1: Beispielüberschrift 5](#_Toc505081892)

[Anhang 2/1: Teil 1 des Anhangs 2 5](#_Toc505081893)

[Anhang 2/2: Teil 2 des Anhangs 2 6](#_Toc505081894)

Anhang 1: Beispielüberschrift

Hier können Sie dann Unterlagen einfügen (Empfehlung: Im Text auf den Anhang verweisen). Die Überschrift des Anhangs bitte über Start – Formatvorlagen – Anhang formatieren; nur dann kann ein korrektes Anhangsinhaltsverzeichnis erstellt werden, das nicht im Inhaltsverzeichnis vorne erscheint.

Anhang 2/1: Teil 1 des Anhangs 2

Bei Anhängen, die nicht auf eine Seite passen, können Sie den Anhang ggf. aufteilen.

Anhang 2/2: Teil 2 des Anhangs 2

Bei Anhängen, die nicht auf eine Seite passen, können Sie den Anhang ggf. aufteilen.

# Quellenverzeichnis

**Literaturverzeichnis**

**Preiß, N. (2007):** Entwurf und Verarbeitung relationaler Datenbanken, München/Wien: Oldenbourg

**Steger, J. (2006):** Kosten- und Leistungsrechnung, 4. Aufl., München/Wien: Oldenbourg

**Stoi, R. (2003):** Management und Controlling von Intangibles, in: Studium & Praxis, 4. Jg., Nr. 1, S. 34-46

**Gesprächsverzeichnis (bei Bedarf)**

**Musterfrau, E. (2018):** Bereichsleiterin Sales, Fantasieunternehmen GmbH, Stuttgart, persönliches Gespräch am 25.01.2018 in Waiblingen

# Erklärung

Ich versichere hiermit, dass ich meine [Projektarbeit / Bachelorarbeit / Seminararbeit] mit dem Thema: [ Thema einfügen ] selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Unterschrift)