Hochschule für Telekommunikation Leipzig

MiZe Dokumentation

Projektdokumentation des Softwareentwicklungsprojektes „HfTL Mitfahrzentrale – MiZe“ im Zeitraum vom 18.03.2015 bis 25.09.2015.

J. Dümig, T. Kilian, B. Schmitz, F. Seidel, D. Tonn, C. Wiegel
BWI13

Impressum

Herausgegeben von

Softwareentwicklungsprojekt HfTL Mitfahrzentrale – MiZe
J. Dümig, T. Kilian, B. Schmitz, F. Seidel, D. Tonn, C. Wiegel
Hochschule für Telekommunikation Leipzig, Gustav-Freytag-Straße 43-45, 04277 Leipzig

Dateiname	**Dokumentnummer**	**Dokumentname**
MiZe_Dokumentation.pdf	1	MiZe Dokumentation
Version	**Zuletzt geprüft**	**Status**
1.0	17.09.2015	Final
Autoren	**Inhaltlich geprüft von**	**Veröffentlicht von**
MiZe Projektteam	MiZe Projektteam	D. Tonn
Ansprechpartner	**Telefon / FAX**	**E-Mail**
D. Tonn	ausschließlich per E-Mail	s134325@hft-leipzig.de

Kurzbeschreibung

Dieses Dokument dient als Projektdokumentation für das Softwareentwicklungsprojekt „HfTL Mitfahrzentrale – MiZe“, das im Rahmen des Moduls Softwareengineering durch die oben genannten Studenten der Hochschule für Telekommunikation Leipzig im Sommersemester 2015 erbracht wurde.

Inhaltsverzeichnis

1. Einleitung ..... - 1 -
   1.1. Teamvorstellung ..... - 1 -
   1.2. Aufgabenstellung ..... - 1 -
   1.3. Idee / Motivation ..... - 2 -
2. Architektur ..... - 3 -
3. MiZe - Systembeschreibung ..... - 5 -
4. Realisierungsphase ..... - 11 -
   4.1. Vorgehensmodell ..... - 11 -
   4.2. Versionsverwaltung ..... - 12 -
   4.3. Besonderheiten während der Realisierungsphase ..... - 13 -
5. Installation ..... - 23 -
   5.1. Systemvoraussetzungen ..... - 23 -
   5.2. Installation der Software ..... - 23 -
   5.3. Konfiguration der Software ..... - 24 -
   5.4. Starten und Stoppen der Applikationen ..... - 25 -
   5.5. Backupstrategien ..... - 26 -
6. Betrieb ..... - 26 -
7. Test ..... - 27 -
8. Wirkbetriebsüberführung ..... - 30 -

A. Anhang i

Abbildungsverzeichnis

Abbildung 1: MiZe Frontend ..... - 3 -
Abbildung 2: MiZe Datenbankmodell ..... - 4 -
Abbildung 3: MiZe Klassendiagramm 01 ..... - 5 -
Abbildung 4: MiZe Klassendiagramm 02 ..... - 6 -
Abbildung 5: MiZe Klassendiagramm 03 ..... - 7 -
Abbildung 6: MiZe Klassendiagramm 04 ..... - 7 -
Abbildung 7: MiZe Klassendiagramm 05 ..... - 8 -
Abbildung 8: MiZe Aktivitätsdiagramm ..... - 9 -
Abbildung 9: MiZe Anwendungsfalldiagramm ..... - 10 -
Abbildung 10: MiZe Sequenzdiagramm ..... - 10 -
Abbildung 11: MiZe GitHub ..... - 12 -
Abbildung 12: Beispiel Anwendungsfalldiagramm ..... - 21 -
Abbildung 13: Beispiel Aktivitätsdiagramm ..... - 21 -
Abbildung 14: Beispiel Klassendiagramm ..... - 21 -
Abbildung 15: Beispiel Sequenzdiagramm ..... - 22 -
Abbildung 16: Beispiel Zustandsdiagramm ..... - 22 -
Abbildung 17: MiZe SoapUI ..... - 27 -
Abbildung 18: MiZe RAW Request ..... - 28 -
Abbildung 19: MiZe JSON Inhalt ..... - 28 -
Abbildung 20: MiZe RAW Response ..... - 28 -
Abbildung 21: MiZe JSON Response ..... - 29 -
Abbildung 22: MiZe Validierung ..... - 29 -

Tabellenverzeichnis

Tabelle 1: MiZe Weekly Übersicht ..... - 11 -
Tabelle 2: Vergleich der UML-Diagramme ..... - 20 -

1. Einleitung

Dieses Dokument dient als Projektdokumentation des Softwareentwicklungsprojektes „HfTL Mitfahrzentrale – MiZe“, das im Rahmen des Moduls Softwareengineering der berufsbegleitenden Wirtschaftsinformatiker als alternative Prüfungsleistung abgelegt wurde.

Ziel des Dokumentes ist die Beschreibung des Softwaresystems auf Basis von UML Diagrammen, die Erklärung der Vorgehensweise während der Entwicklung, sowie die detaillierte Erläuterung der eigentlichen Installation. Zusätzlich gibt das Dokument einen Ausblick über mögliche Migrationsoptionen in einen späteren Produktivbetrieb.

1.1. Teamvorstellung

Das MiZe Projektteam bestand während der kompletten Entwicklungsphase von März bis September 2015 aus den folgenden, berufsbegleitenden Wirtschaftsinformatikstudenten des Matrikel 13:

- Dümig, Johannes - s134306 (Deutsche Telekom Technik GmbH – Bamberg)
- Kilian, Tobias - s134312 (T-Systems International GmbH – Berlin)
- Schmitz, Björn - s134321 (T-Systems International GmbH – Darmstadt)
- Seidel, Florian - s134322 (Deutsche Telekom Technik GmbH – Nürnberg)
- Tonn, Dustin - s134325 (T-Systems International GmbH – Berlin)
- Wiegel, Christoph - s134329 (T-Systems International GmbH – Magdeburg)

Während der gesamten Projektlaufzeit fungierte D. Tonn als offizieller Ansprechpartner bzw. Projektverantwortlicher.

1.2. Aufgabenstellung

„Das Praktikum hat zum Ziel, den Studierenden anhand eines fiktiven Szenarios einen Einblick in ein Softwareentwicklungsprojekt zu geben. Dabei steht die Arbeit mit dem UML - Standard sowie der Umgang mit aktuellen Entwicklungswerkzeugen im Vordergrund.
Das Praktikum ist aufgeteilt in Vorbereitung, Durchführung und Nachbereitung. Mit der Lösung der vorbereitenden Aufgaben erarbeiten Sie sich die Voraussetzungen für den Durchführungsteil, welcher in einem Computerpool an der HfTL, bzw. im Selbststudium gemeinsam bearbeitet wird. Als Software kommt das Programm Enterprise Architect von Sparx Systems zum Einsatz. Zusammen mit Antworten auf die Fragen der Nachbereitung bildet die erfolgreiche Bearbeitung aller Teile die Prüfungsvorleistung.“

Quelle: S. Wieland – „Tauschbörse.pdf“

Basierend auf dieser Aufgabenstellung wurden im Vorfeld mit der verantwortlichen Professorin des Moduls die Grundlagen und das weitere Vorgehen für ein eigens gewähltes Projekt diskutiert und besprochen. Das daraus resultierende Grobkonzept (siehe Anhang) diente in der Vorbereitungsphase als erster Lösungsansatz und wurde im weiteren Verlauf der Entwicklung für die Prozessgestaltung und Realisierung herangezogen.

1.3. Idee / Motivation

Während der Vorstellung der Aufgabenstellung und möglichen Beispiele wurde dem Projektteam schnell klar, dass es bei der Bearbeitung der alternativen Prüfungsleistung auf ein selbstgewähltes Szenario hinauslaufen wird.

Nach mehrfacher, interner Diskussion über mögliche Szenarien fiel die Entscheidung hinsichtlich einer Mitfahrzentrale für die Studenten der Hochschule für Telekommunikation Leipzig eindeutig aus. Da das Projektteam selbst aus den unterschiedlichsten Regionen Deutschlands stammt und für die Anreise zu jeder Präsenzphase auf das eigene Fahrzeug bzw. öffentliche Verkehrsmittel angewiesen ist, konnten somit auch die eigenen Interessen vertreten werden.

Zudem war allen bewusst, dass eine hochschulinterne Mitfahrzentrale neben den üblichen Vorteilen vergleichbarer Institutionen (Attraktiver, günstiger, schneller gegenüber Fernbussen bzw. der Bahn; umweltschonender) auch das Miteinander der Studenten fördern sollte, da beispielsweise Lerninhalte und Wissen während der An- und/oder Abreise ausgetauscht werden könnten.
Folgende E-Mail wurde der verantwortlichen Professorin am 08.04.2015 als Projektvorschlag zugesandt (Tobias Kilian zu diesem Zeitpunkt noch nicht im Team):

Sehr geehrte Frau Prof. Dr.-Ing. Wieland,

die nachfolgend aufgelisteten Studenten möchten im Rahmen des Moduls Softwareengineering eine webbasierte Mitfahrgelegenheitszentrale für die Studenten der HfTL entwickeln:

** Björn Schmitz (s134321)*
** Christoph Wiegel (s134329)*
** Florian Seidel (s134322)*
** Johannes Dümig (s134306)*
** Dustin Tonn (s134325)*

Anbei ein erster Entwurf der Szenariobeschreibung:

"... gesucht ist eine Software zur Bereitstellung einer Online Mitfahrgelegenheitszentrale für nicht ortsansässige Studenten der HfTL. Die Nutzer der Zentrale können nach Angabe ihres persönlichen Profils selbstständig Fahrten zu konkreten Terminen bzw. Präsenzphasen anbieten (inkl. möglicher Zwischenstopps auf dem Weg zur HfTL). Ebenso können sich die Nutzer für freie Plätze als Bei- bzw. Mitfahrer bei anderen Nutzern melden. Nutzer, die Fahrten anbieten, geben an, zu welchen Preisen sie andere Nutzer bzw. Kommilitonen mitnehmen wollen/können. Stimmt ein Nutzer dem Angebot eines anderen Nutzers zu, kommt das Geschäft zustande. Beide Parteien bekommen im Anschluss die jeweiligen Kontaktdaten des anderen Nutzers für eine abschließende Abstimmung übermittelt ..."

Könnten Sie bitte prüfen, ob unsere Projektidee Ihren Vorstellungen bzw. Kriterien entspricht? Über eine zeitnahe Zu- bzw. Absage des Projektvorschlages würden wir uns folglich sehr freuen.
Im Namen aller erwähnten Studenten, herzlichen Dank im Voraus!

Mit besten Grüßen
Dustin Tonn

2. Architektur

Die Architektur der Mitfahrzentrale basiert auf einem Multi-Tier Ansatz und besteht folglich aus mehreren Ebenen bzw. Schichten. Konkret handelt es sich um eine 3-Tier Architektur bestehend aus den folgenden Ebenen:

1. Frontend

Das Frontend von MiZe dient im klassischen Sinne als Präsentationsschicht und ist somit für die Repräsentation der Daten, Benutzereingaben und als Schnittstelle verantwortlich.

Folgende Technologien wurden im Rahmen der Umsetzung von MiZe eingesetzt:

- HTML5 – (Hypertext Markup Language)
- jQuery (AJAX) – (Asynchronous JavaScript and XML)
- CSS3 – (Cascading Style Sheets)

Die nachfolgende Grafik zeigt eine frühe, nicht finale Version des Frontend:

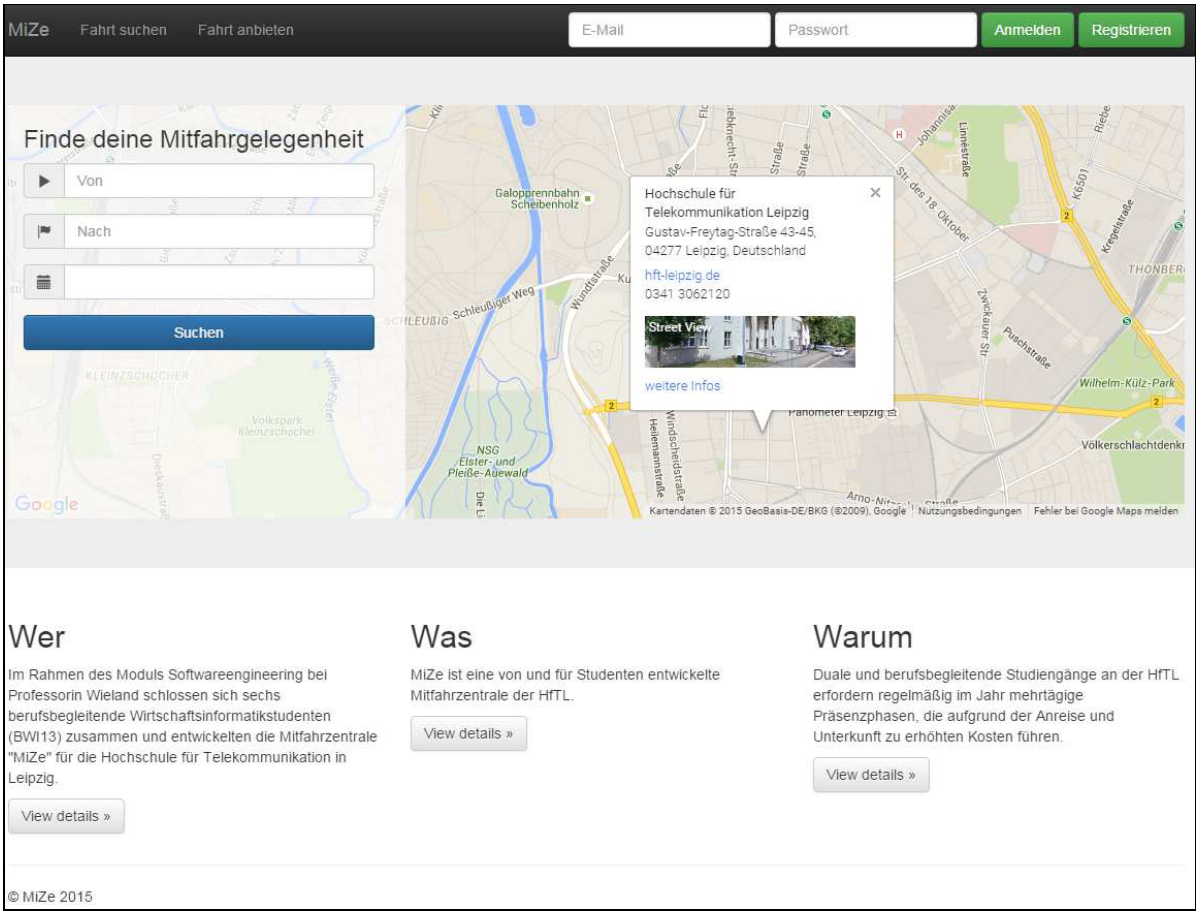


Abbildung 1: MiZe Frontend

2. Backend

Das Backend von MiZe beinhaltet sämtliche Verarbeitungsmechanismen bzw. die eigentliche Anwendungslogik. Bei der Implementierung wurde verstärkt auf folgende Technologien gesetzt:

- JAX-RS – (Java API for RESTful Web Services)
- Swagger / Swagger-UI
- POJO – (Plain Old Java Object)

Detaillierte Informationen zu Swagger-UI können dem Anhang entnommen werden.

3. Datenbank

MiZe nutzt eine MySQL Instanz zur Vorhaltung der erforderlichen, persistenten Daten.

Folgende Grafik zeigt das vollständige Datenbankmodell der Mitfahrzentrale:

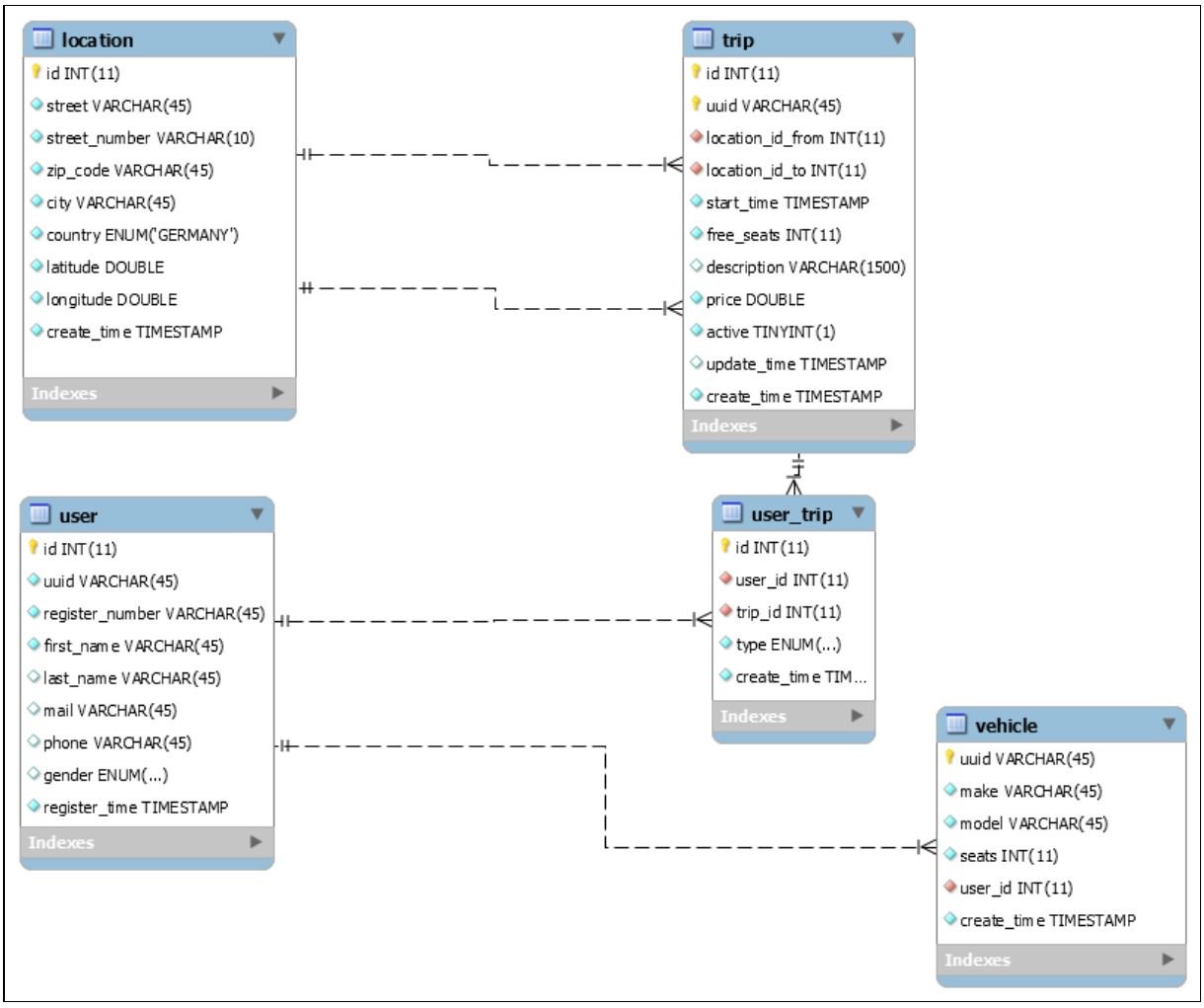


Abbildung 2: MiZe Datenbankmodell

Die Vorteile dieser 3-Tier Architektur können wie folgt zusammengefasst werden:

- gute Skalierbarkeit, da die unterschiedlichen Schichten logisch voneinander entkoppelt sind
- Möglichkeit zum Austausch der Komponenten der einzelnen Schichten durch andere Technologien gegeben
- Aufteilung begünstigt die Wartung des Systems

3. MiZe - Systembeschreibung

Die Mitfahrzentrale „MiZe“ der Hochschule für Telekommunikation Leipzig besteht aus den nachfolgend aufgelisteten Java Klassen. Als Veranschaulichung dienen Auszüge aus dem automatisch erstellten Klassendiagramm:

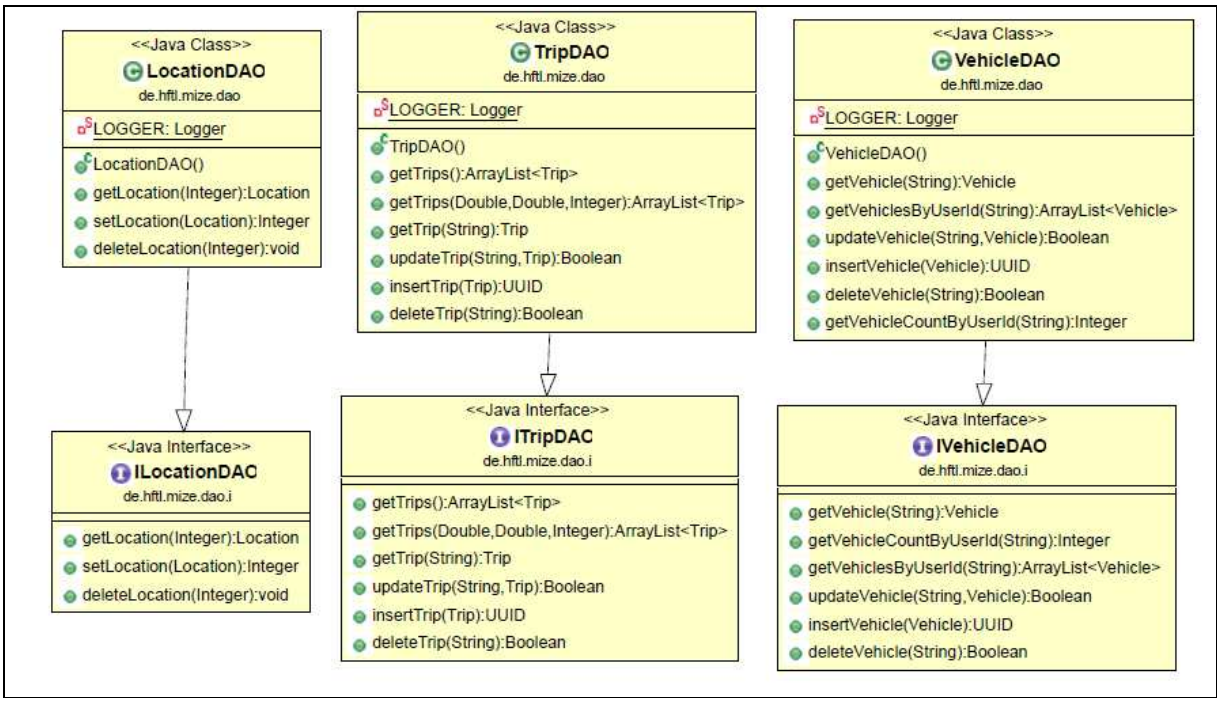


Abbildung 3: MiZe Klassendiagramm 01

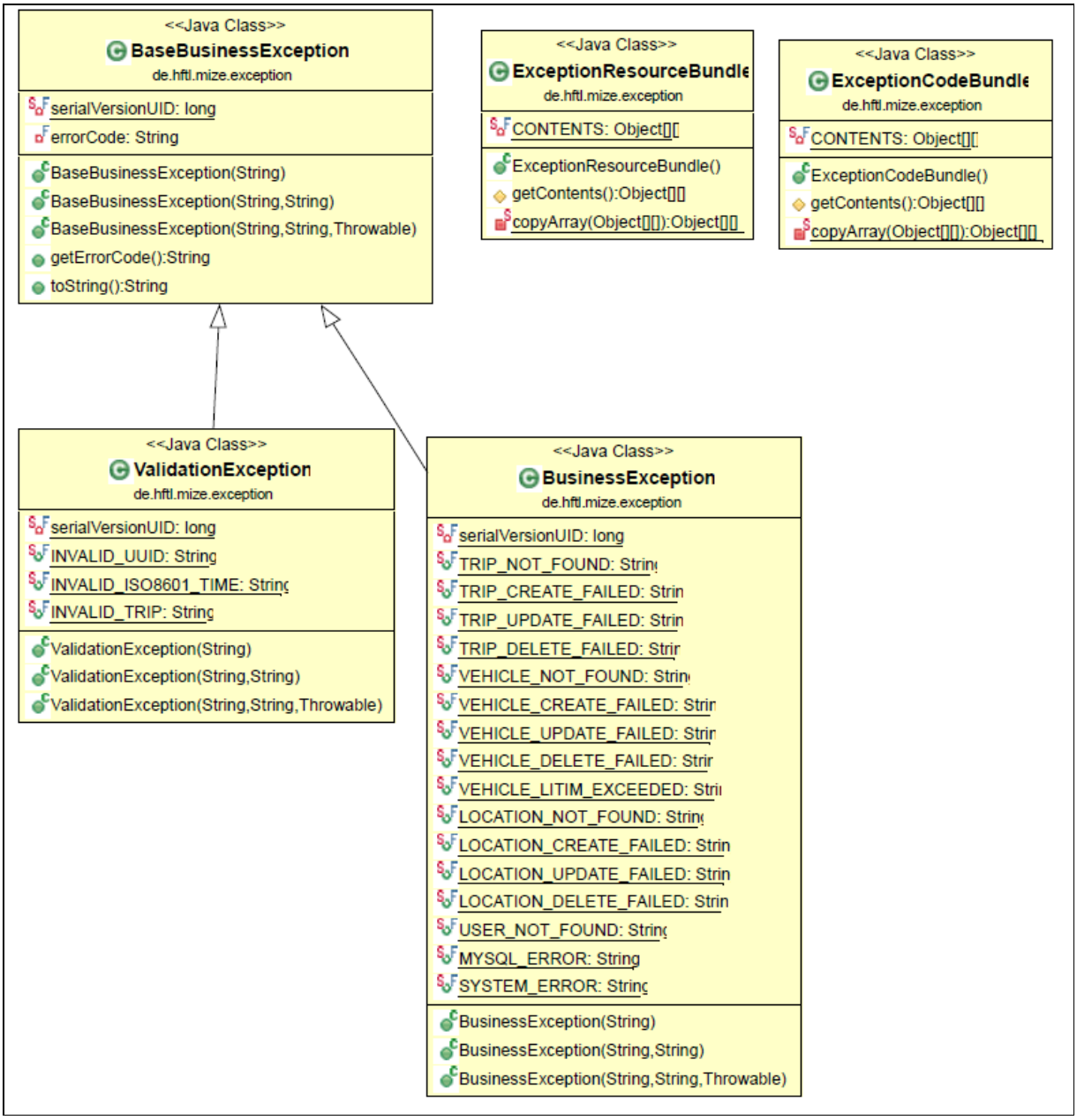


Abbildung 4: MiZe Klassendiagramm 02

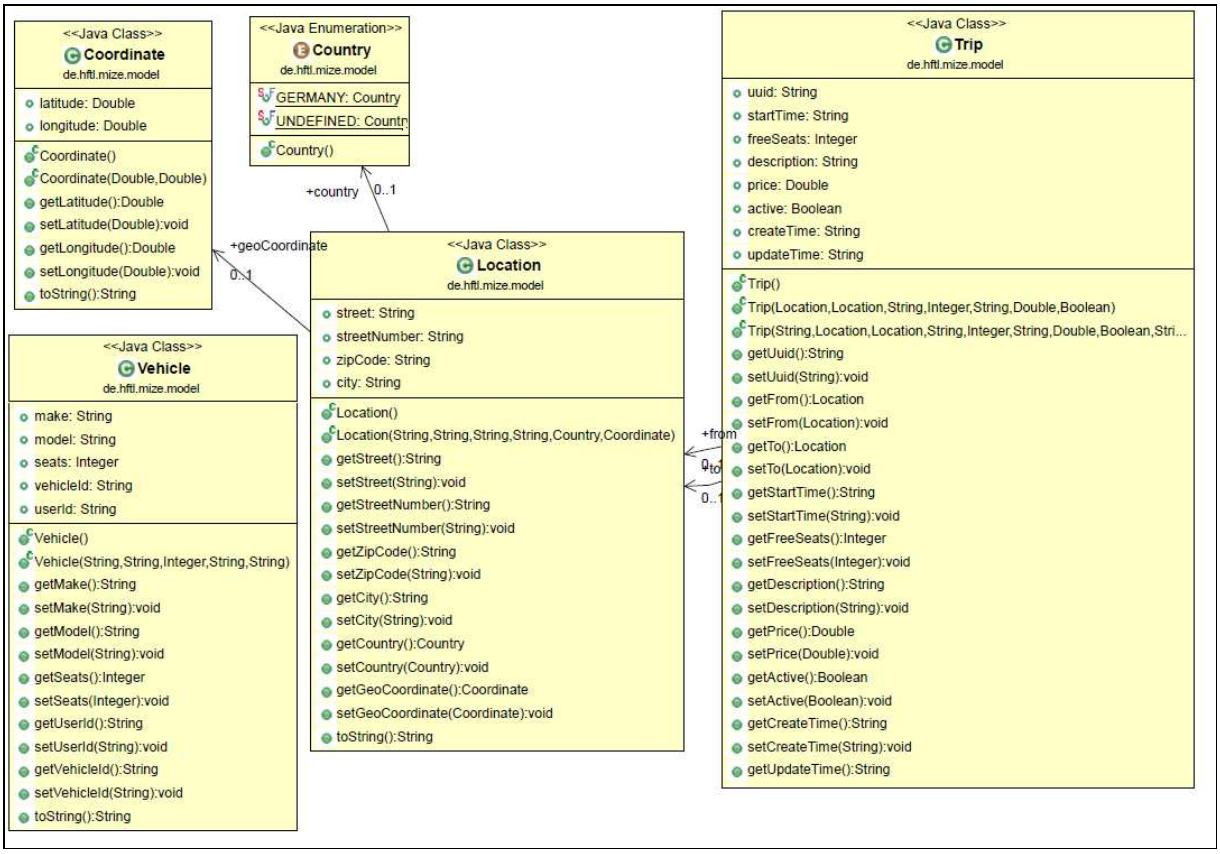


Abbildung 5: MiZe Klassendiagramm 03

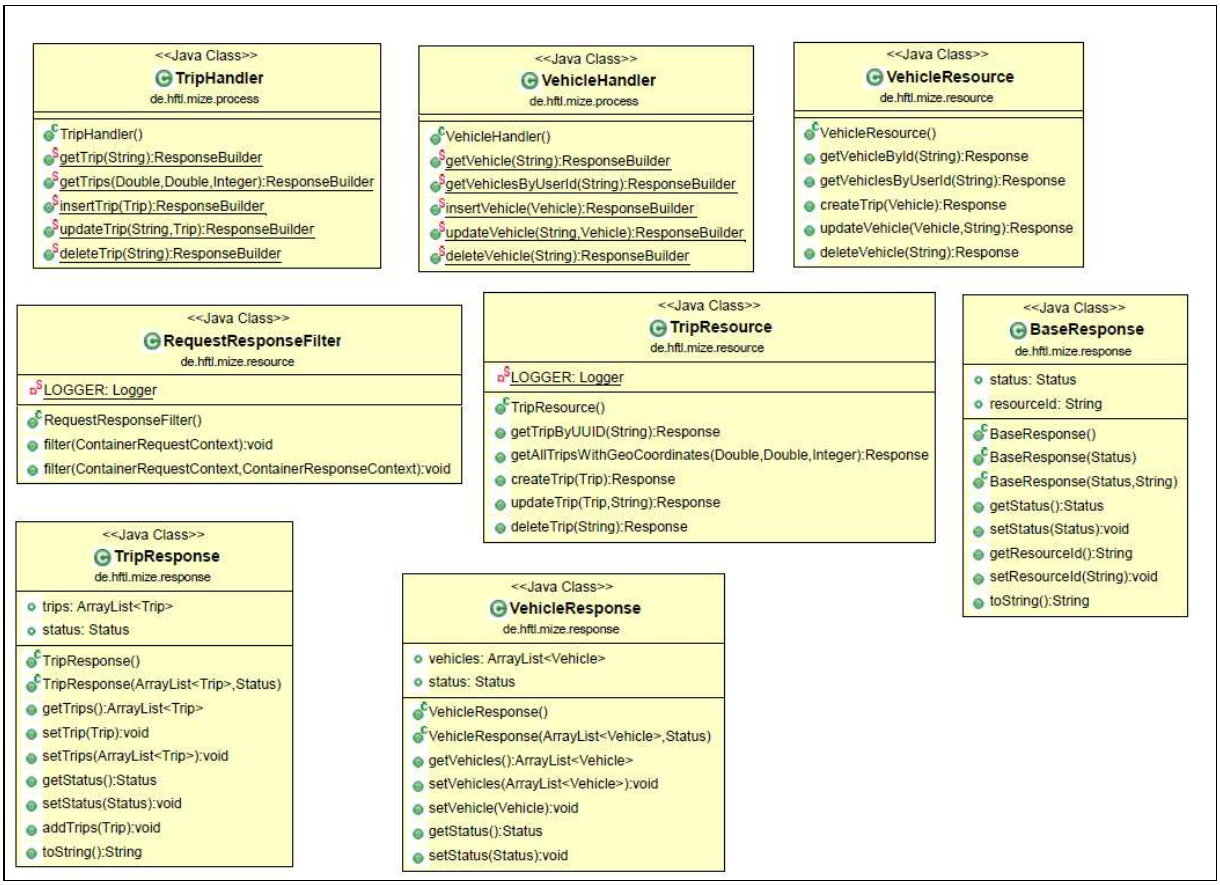


Abbildung 6: MiZe Klassendiagramm 04

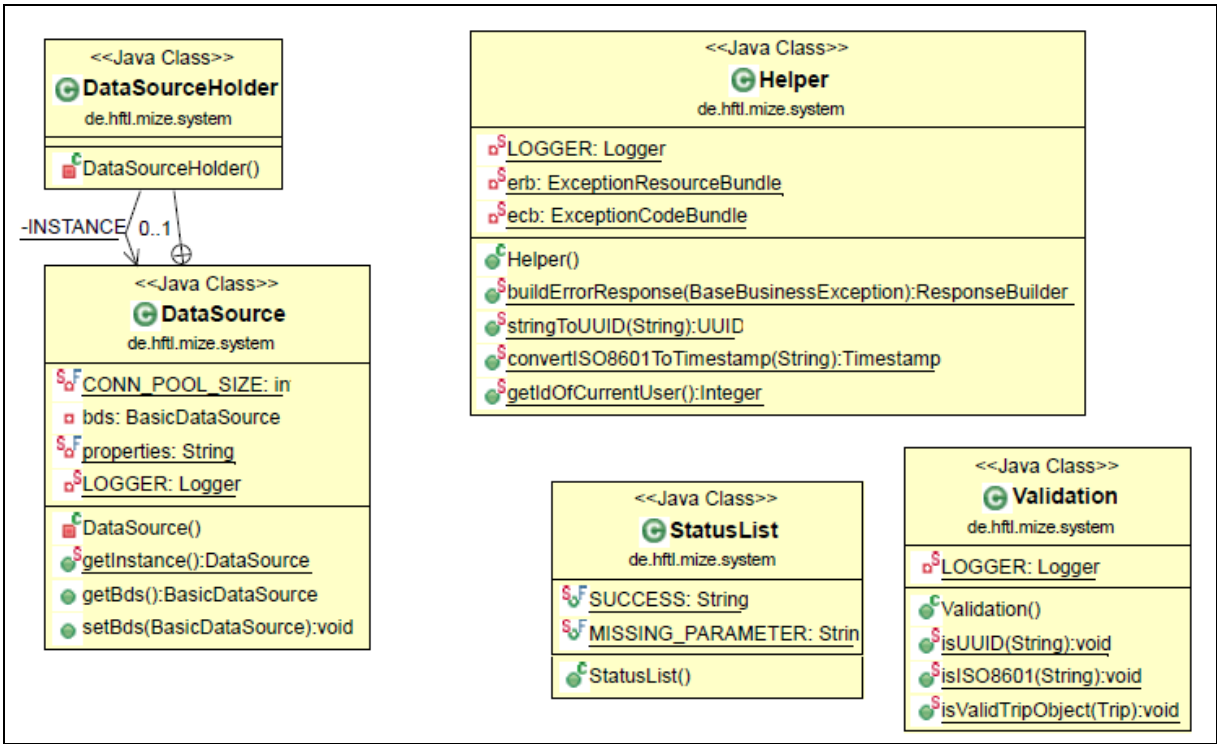


Abbildung 7: MiZe Klassendiagramm 05

Das folgende Aktivitätsdiagramm gibt einen Überblick über die möglichen Szenarien nach der erfolgreichen Nutzeranmeldung:

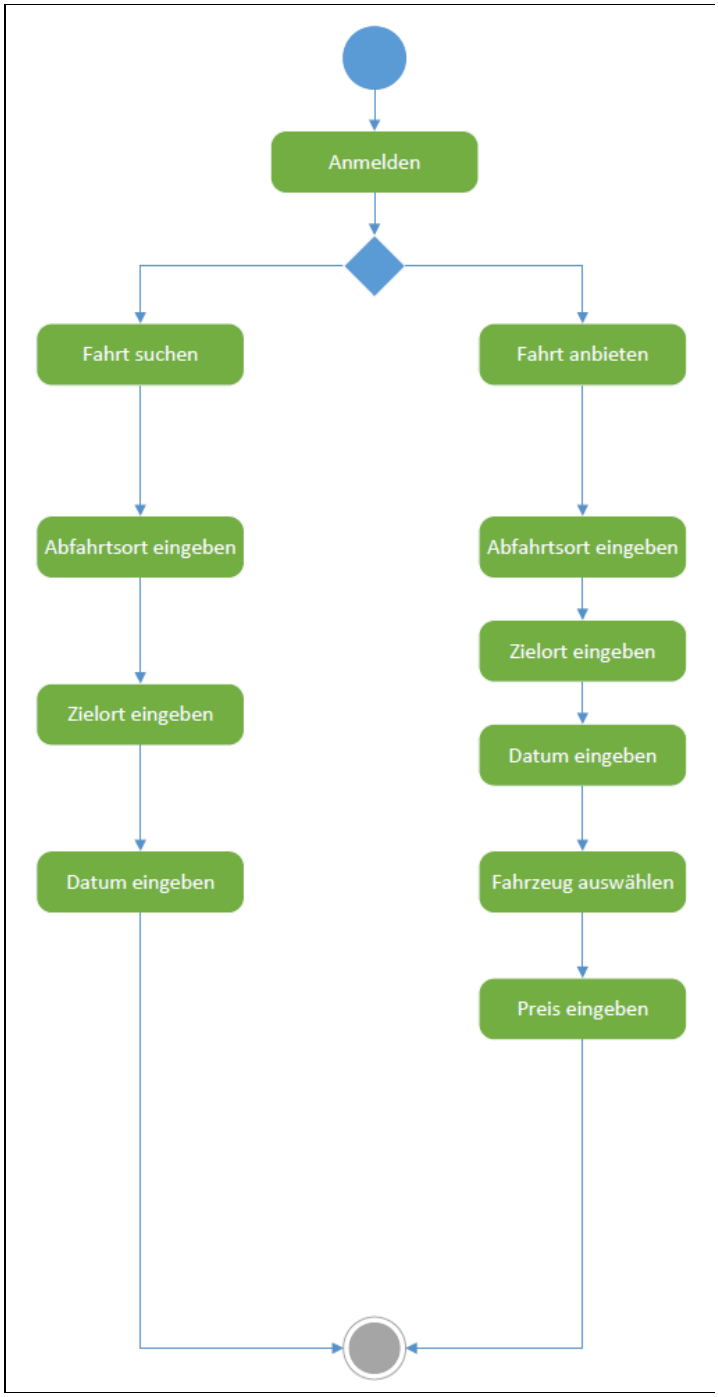


Abbildung 8: MiZe Aktivitätsdiagramm

Weitere Detaillierungen, auch hinsichtlich der vorkommenden Akteure und angebundenen Systeme, können dem Anwendungs- und Sequenzdiagramm entnommen werden:

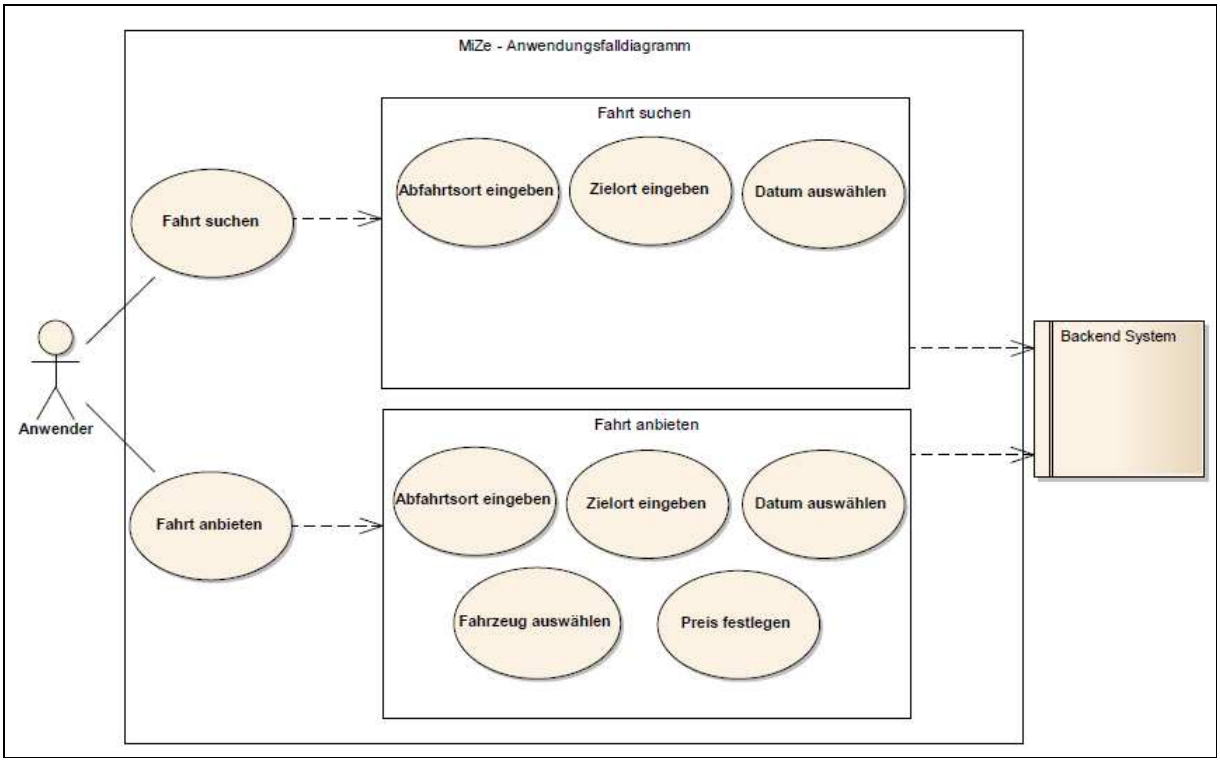


Abbildung 9: MiZe Anwendungsfalldiagramm

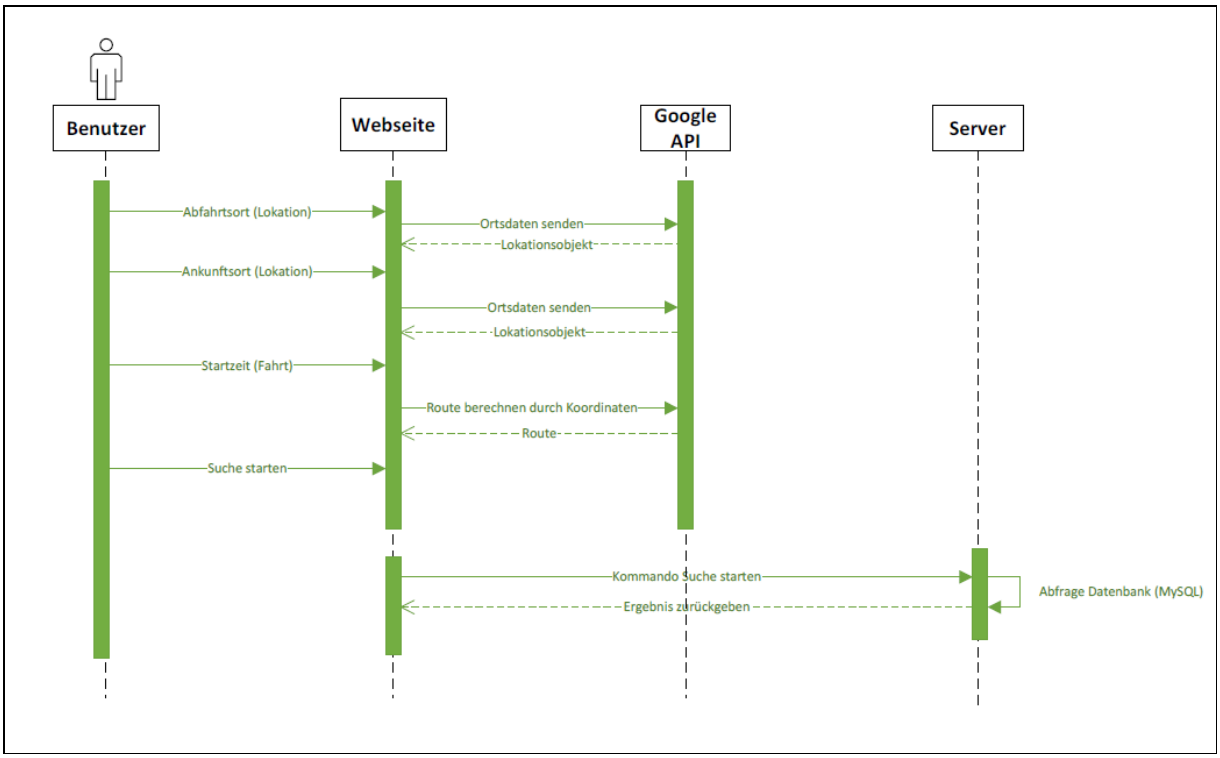


Abbildung 10: MiZe Sequenzdiagramm

4. Realisierungsphase

Dieses Kapitel dient der Beschreibung des verwendeten Vorgehensmodells, der eingesetzten Versionsverwaltung, sowie der Besonderheiten, die während der Entwicklungsphase aufgetreten sind. Zudem erfolgt die Beantwortung der zusätzlichen Aufgabenstellungen.

4.1. Vorgehensmodell

Aufgrund bereits existierender Vorerfahrungen der einzelnen Projektmitglieder fiel die Entscheidung hinsichtlich eines geeigneten Vorgehensmodells für die Realisierung des Projektes auf SCRUM.

SCRUM punktet vor allem durch den agilen Ansatz und stellt funktionierende Software und die Zusammenarbeit mit Kunden bzw. im Team besonders in den Vordergrund. Fest definierte Entwicklungszyklen sorgen für Transparenz, Qualität und Sicherheit. Mit Hilfe der regelmäßig stattfindenden Meetings können Hindernisse und Herausforderungen schnell publiziert und angegangen werden. Der Produkt- bzw. Projekterfolg wird mit SCRUM dadurch erreicht, dass es geänderte Anforderungen (vorhandene werden geändert, neue kommen hinzu) zulässt und in kurzen Entwicklungszyklen / Sprints darauf reagiert werden kann.

Während der Realisierungsphase wurde das Team in folgende Rollen aufgeteilt:

- Dümig, Johannes – SCRUM Team (Infrastruktur & Deployment)
- Kilian, Tobias – SCRUM Team (Architektur & Entwicklung)
- Schmitz, Björn – SCRUM Team (Test)
- Seidel, Florian – SCRUM Team (Prozessdesgin & technische Redaktion)
- Tonn, Dustin – SCRUM Master / Product Owner
- Wiegel, Christoph – SCRUM Team (Prozessdesign & technische Redaktion)
- (Professorin Wieland – höheres Management)

Die nachfolgende Tabelle gibt einen Überblick über die geplanten und stattgefundenen, wöchentlichen Besprechungen (Weekly):

MiZe – Weekly Übersicht

Weekly Nr.	Stattgefunden am:	Weekly Nr.	Stattgefunden am:
1	19.04.2015	11	05.07.2015
2	26.04.2015	12	19.07.2015
3	03.05.2015	13	26.07.2015
4	10.05.2015	14	02.08.2015
5	17.05.2015	15	09.08.2015
6	31.05.2015	16	16.08.2015
7	07.06.2015	17	23.08.2015
8	14.06.2015	18	30.08.2015
9	21.06.2015	19	06.09.2015
10	28.06.2015	20	13.09.2015

Tabelle 1: MiZe Weekly Übersicht

Für die Zusammenschaltung des geografisch getrennten Projektteams wurde das von Google vertriebene Konferenztool „Hangouts“ verwendet.

Aufgrund der Besonderheiten des berufsbegleitenden Studiums ist es nachvollziehbar, dass tägliche Meetings (Daily) nicht, wie in SCRUM beschrieben, stattfinden konnten. Um dennoch agil auf Hindernisse und Herausforderungen eingehen zu können, wurde vom Projektteam eine separate Chatgruppe mit Hilfe von WhatsApp eingerichtet, in der dann täglich in „Daily-Manier“ die aktuellen Stände abgefragt wurden.

4.2. Versionsverwaltung

Wie in der Aufgabenstellung beschrieben, wurde zur Realisierung der Versionsverwaltung eine auf MiZe abgezielte GitHub Instanz eingerichtet:

https://github.com/s134325/HfTL-Mitfahrgelegenheit

Alle Projektmitglieder erhielten entsprechenden Zugriff, um die jeweiligen Entwicklungsstände zu den festgelegten Sprint-Terminen hochladen zu können. Folgende Grafik zeigt einen Stand der „Commits“ der jeweiligen Projektmitglieder:

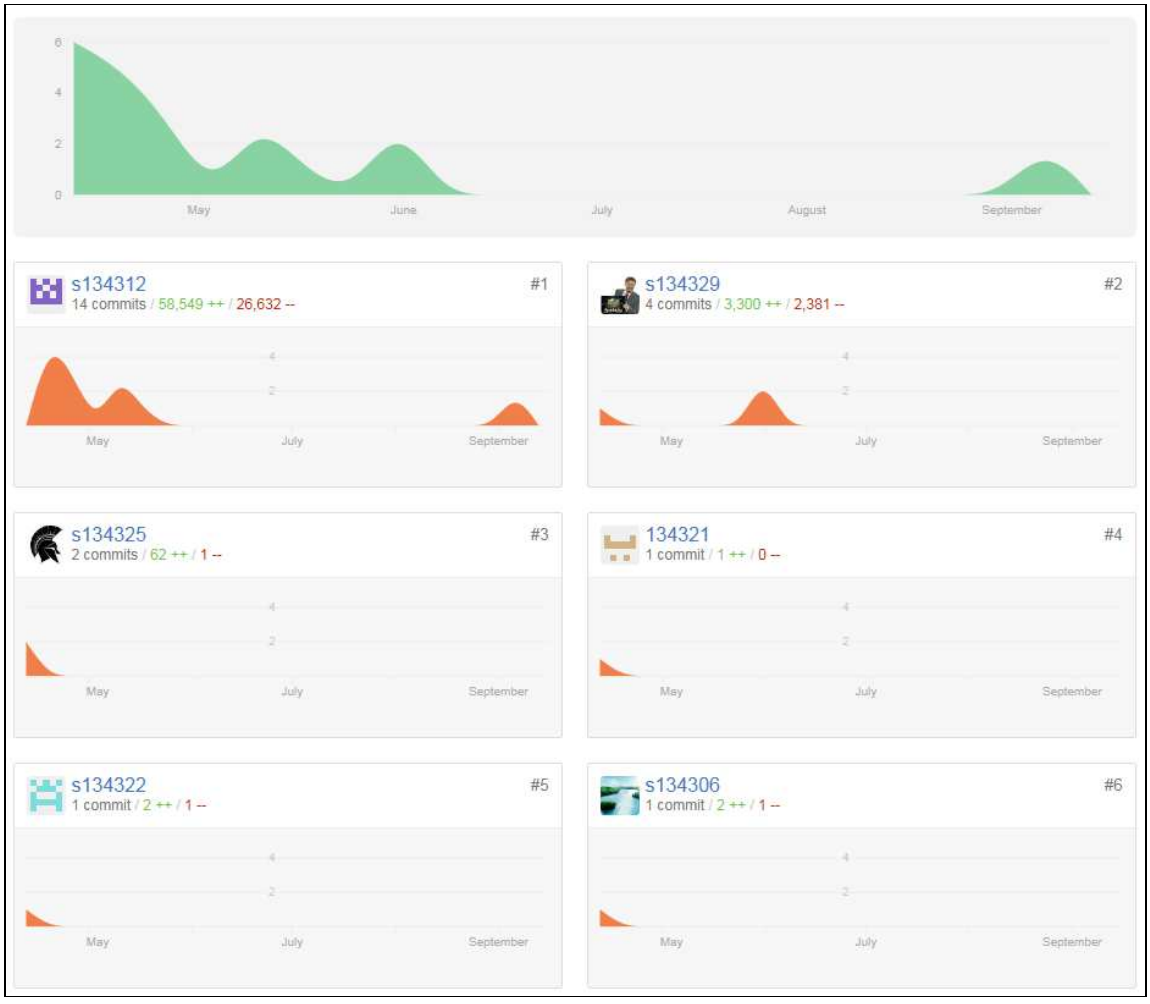


Abbildung 11: MiZe GitHub

Hinweis: Diese Grafik wurde vor dem eigentlichen Projektende im Rahmen der Dokumentationsarbeiten erzeugt und spiegelt daher nicht den finalen Stand wider! Die finale Statistik kann nach Absprache mit dem Projektverantwortlichen eingesehen werden. Eine Freischaltung auf die GitHub Instanz ist ebenfalls möglich. Aufgrund der unterschiedlichen, im Vorfeld festgelegten, Teamrollen ist es ebenfalls verständlich, dass unterschiedliche Commit-Angaben zustande kommen.

Als zusätzliche Unterstützung wurde vom Team eine privat organisierte ownCloud Instanz für die primäre Verwaltung und Vorhaltung von Daten und Dokumenten verwendet. Die Adresse dieser Instanz wird aus privaten Gründen nicht in der Dokumentation aufgeführt, kann aber bei Bedarf angefragt werden.

4.3. Besonderheiten während der Realisierungsphase

Im Zuge der Prozessmodellierungs- und Entwicklungsphase wurde häufig mit dem Modellierungstool „Enterprise Architect“ der Firma Sparx Systems gearbeitet. Folglich lässt sich die nachfolgende Frage hinsichtlich der Unterstützung durch den EA relativ einfach beantworten:

„*Wie unterstützt der EA bei der Erstellung der UML Diagramme? Nennen Sie mindestens drei Beispiele, die Ihnen besonders aufgefallen sind (positiv oder negativ)!*“

Zusammenfassend lässt sich feststellen, dass Enterprise Architect zur Erstellung von UML Diagrammen nur bedingt zu empfehlen ist, da es für Neueinsteiger bzw. Laien eine umständliche Softwarelösung darstellt. Mit alternativen Tools, wie Microsoft Visio oder die Open Source Software „Dia“ ist es dagegen leicht möglich, ein Diagramm zu erstellen und dies zeitnah fertigzustellen. Enterprise Architect bietet zwar viele Funktionen, die aber ohne gründliche Einarbeitung mittels Unterlagen oder Tutorials schwer zu erahnen bzw. entdecken sind. Der dafür notwendige, zeitliche Sonderaufwand hätte sich durchaus negativ auf das Projekt auswirken und zu Verzögerungen führen können. Während der Erstellung der Diagramme konnten keine wesentlichen Vorteile gegenüber den anderen, erwähnten Tools festgestellt werden. Folgende Punkte sind zudem negativ in Erscheinung getreten:

- Starten der Software: die EA Lizenz funktioniert nur, wenn eine Verbindung mit dem HfTL Campusnetz hergestellt werden konnte. Für die Verbindung muss zwingend ein VPN Tunnel eingerichtet werden. Die erwähnten Alternativtools punkten hier aufgrund des Zeitfaktors und der Anwenderfreundlichkeit.
- Verwendung der Software: Zunächst muss das eigentliche Softwareprojekt initial im EA angelegt werden. Anschließend kann mit der Modellierung der Prozesse bzw. Erstellung der Diagramme begonnen werden. Hinderlich ist, dass die Diagrammoptionen erst durch aufwändiges Suchen zum Vorschein kommen und unterschiedliche Diagrammtypen in komplizierten Menüstrukturen versteckt sind. Auch hier punkten die Alternativtools hinsichtlich der Menüführung und Übersichtlichkeit.
- Exportfunktion: Beim Erstellen einer PDF Datei aus dem EA heraus erscheinen häufig Darstellungsfehler, die im Nachhinein nicht mehr korrigiert werden können.
- Hilfefunktion: Der EA bietet nach eigenen Erfahrungen keine zufriedenstellende Hilfefunktion an. Es erfolgt lediglich ein Verweis auf die Webseite des Herstellers.

Ziel des Praktikums bzw. Projektes war es, einen Überblick in die Softwareentwicklung zu bekommen und mit dem UML Standard zu arbeiten. Folglich wurden während der Realisierung unterschiedliche Diagrammtypen betrachtet, die in der nachfolgenden Tabelle erfasst und gegenübergestellt wurden.

„Stellen Sie die Eigenschaften der Diagramme gegenüber! Gehen Sie dabei darauf ein, was in den Diagrammen dargestellt wird und welche Aspekte in Bezug auf die anderen Diagramme nicht ersichtlich sind! Nutzen Sie dazu eine Tabelle!“

Diagrammtyp	Anwendungsfalldiagramm	Aktivitätsdiagramm	Klassendiagramm	Sequenzdiagramm	Zustandsdiagramm
Einordnung	• Verhaltensdiagramm • Betrachtungsweise: Requirements (Anforderungen an das System) • Anwendungsfalldiagramm → Akteure, Szenarien	• Verhaltensdiagramm • Betrachtungsweise: dynamische Sicht (Interaktion, Abläufe im System) • Aktivitätsdiagramm → Ablaufmöglichkeiten	• Strukturdiagramm • Betrachtungsweise: statische Sicht (logischer Aufbau des Systems) • Klassendiagramm → Klassen, Beziehungen	• Verhaltensdiagramm • Betrachtungsweise: dynamische Sicht (Interaktionen, Abläufe im System) • Sequenzdiagramm → Objekte, Interaktionen	• Verhaltensdiagramm • Betrachtungsweise: dynamische Sicht (Interaktionen, Abläufe im System) • Zustandsdiagramm → internes Verhalten von Objekten
Eigenschaften	• Anwendungsfalldiagramme (engl. Use-Case Diagram) beschreiben das Zusammenwirken von Personen (Akteuren) mit einem System --> stellt Akteure, Anwendungsfälle und deren Beziehungen dar • als Akteur werden außerhalb des betrachteten Systems liegende Personen bzw. andere technische Systeme bezeichnet, welche mit dem System zum Erreichen eines konkreten Ziels interagieren • durch einen Anwendungsfall wird eine abgeschlossene Teilfunktionalität des Anwendungssystems	• Aktivitätsdiagramme beschreiben die Objekte eines Programms mittels der Aktivitäten, die sie während des Programmablaufes vollführen • alle Szenarien eines Anwendungsfalls werden in einem Aktivitätsdiagramm beschrieben • Anwendungsfälle können durch eine Zielaussage des Akteurs gefunden werden oder über eine Geschäftsprozessanalyse, bei der der Arbeitsablauf des Akteurs untersucht wird • eine Aktivität ist ein einzelner Schritt innerhalb eines	• Klassendiagramme dienen der grafischen Darstellung der existierenden Klassen und deren Beziehungen untereinander • beschrieben die statische Struktur von Objekten in einem System und ihre Beziehungen untereinander → zentraler Bestandteil der UML und auch zahlreicher objektorientierter Methoden • die UML beschränkt sich auf die Beschreibung der Notation und Semantik • zwei wesentliche Elemente: • Objekt: ein Konzept, eine Abstraktion oder ein	• das Sequenzdiagramm visualisiert Nachrichten, welche eine begrenzte Menge von beteiligten Akteuren oder Objekten in einer zeitlich begrenzten Situation austauscht, beschreibt die zeitliche Abfolge von Interaktionen zwischen einer Menge von Objekten innerhalb eines zeitlich begrenzten Kontextes • im Vordergrund steht der zeitliche Verlauf der Nachrichten • Sequenzdiagramm sind verschachtelbar • eine Menge von verschiedenen Abläufen (Aktivitäten) wird in verschiedene Sequenzen zerlegt	• Zustandsdiagramme visualisieren die unterschiedlichen Zustände, die Objekte in ihrem Leben annehmen können • werden auch als Zustandsübergangsdiagramme (engl. State Transition Diagram) bezeichnet, da sie auch die Funktionen beschreiben, die zu Änderungen des Zustandes eines Objektes führen • beschreibt eine hypothetische Maschine, die sich zu jedem Zeitpunkt in einer Menge endlicher Zustände befindet, sie besteht aus: • einem Anfangszustand

	beschrieben, welche für alle beteiligten Akteure ein erkennbares Ereignis liefert • es werden typische Handlungen beschrieben, die ein Benutzer mit dem System ausführt, z.B. "Versicherung abschließen" • die Anwendungsfälle können durch Aktivitätsdiagramme, Oberflächenprototypen, Sequenzdiagramme und weitere Dokumente (wie Word oder Excel) näher beschrieben werden • Anwendungsfälle werden durch Szenarien beschrieben (Normalfälle + Problemfälle) • Anwendungsfälle dienen als Grundlage für die Erstellung des Systems und müssen daher vollständig sein • darüber hinaus bilden sie die Grundlage für das Testen des Systems nach der Erstellung	Programmablaufes, der eine oder mehrere von ihm ausgehende Transitionen enthält • jede Aktivität kann in mehrere Teilaktivitäten gegliedert werden, die in Subaktivitätsdiagrammen dargestellt werden • gehen mehrere Transitionen von der Aktivität aus, so müssen diese mittels Bedingungen voneinander zu unterscheiden werden • somit gilt ein Aktivitätsdiagramm als Sonderform eines Zustandsdiagramms, dessen Zustände der Modellelemente in der Mehrzahl als Aktivitäten definiert sind	Gegenstand mit klarer Abgrenzung und präziser Bedeutung z.B. der "rote Apfel" • Klasse: eine Gruppe von Objekten mit ähnlichen Eigenschaften, z.B. "Äpfel"	• mit einem Sequenzdiagramm können die Szenarien von Anwendungsfällen beschrieben werden • Sequenzdiagramme stellen die einzelnen Objekte und ihre Interaktion (auf zeitlichen Ablauf bezogen) dar • Interaktionsdiagramme (wie Sequenzdiagramme) werden ausgehend von Fallbeispielen - den Szenarios - erstellt • beim Erstellen der Diagramme konzentriert man sich auf die wichtigsten Fälle (primäre Szenarios) • anschließend werden die Sonderfälle mit einbezogen und eventuell weitere Diagramme erstellt (sekundäre Szenarios)	• einer endlichen Menge von Zuständen • einer endlichen Menge von Ereignissen • einer endlichen Anzahl von Transitionen, die den Übergang des Objektes von einem zum nächsten Zustand beschreiben • einem oder mehreren Endzuständen

	• die textuelle Beschreibung des Szenarios ist in jedem Fall erforderlich und in den meisten Fällen hinreichend • das Diagramm kann darüber hinaus dazu dienen, die Zusammenhänge zu verdeutlichen				
Was wird dargestellt?	• Darstellung der Zusammenhänge zwischen Anwendungsfällen und Akteuren • Darstellung der Interaktion mit dem System • Anwendungsfälle werden als Ellipsen dargestellt • Anwendungsfälle können beliebig kompliziert und umfangreich sein • Verbindungen zwischen den Anwendungsfällen und den Akteuren werden durch Linien hergestellt • damit wird angezeigt, welche Akteure an dem	• Darstellung der Vernetzung von elementaren Aktionen und deren Verbindungen mit Kontroll- und Datenflüssen • jedes Aktivitätsdiagramm sollte genau einen Start- und mindestens einen Endzustand haben • die Knoten im Aktivitätsdiagramm sind die ablaufenden Aktivitäten, die Kanten sind die Transitionen • Transitionen können mit Bedingungen versehen werden; Transitionen ohne Bezeichnung stellen den „else“-Zweig oder den Standard-	• das zentrale Element im Klassendiagramm sind Klassen • Klassen werden als Rechtecke dargestellt, die den Namen der Klasse und/oder die Attribute und Operationen enthalten • Klassennamen, Attribute und Operationen werden durch eine horizontale Linie getrennt • der Klassenname steht im Singular und beginnt mit einem Großbuchstaben • Attribute können näher beschrieben werden, z.B. durch ihren Typ, einen Initialwert und	• das Sequenzdiagramm beschreibt die zeitliche Abfolge von Interaktionen zwischen einer Menge von Objekten innerhalb eines zeitlich begrenzten Kontextes • die Zeitlinie verläuft senkrecht von oben nach unten, die Objekte werden durch senkrechte Lebenslinien beschrieben und die gesendeten Nachrichten waagerecht entsprechend ihres zeitlichen Auftretens eingetragen • Objekte werden oben durch das entsprechende	• Modell des endlichen Zustandsautomaten (engl. finite state machine - FSM) • Darstellung eine begrenzten und nicht leeren Menge von Zuständen • Darstellung der Start- und Endzustände • Darstellung einer begrenzten Menge, nicht leere Menge von Ereignissen • die Zustände, die ein Objekt im Laufe seiner Existenz annehmen kann • die Ereignisse (empfangene Botschaften), die zu einem Zustandswechsel führen und welche

	entsprechenden Anwendungsfall beteiligt sind • Akteure werden nach ihrem Rollenverhalten in Bezug auf das System und nicht nach ihrer Identität unterschieden	Zweig dar • Transitionen können mit Aktionen verknüpft sein, die beim Übergang von einer Aktivität zur Nachfolgenden ausgeführt werden • im Aktivitätsdiagramm können durch Balken parallele Abläufe dargestellt werden • mit einem weiteren Balken können die parallelen Abläufe wieder synchronisiert werden • in einer swimlane (Schwimmbahn) werden die Aktivitäten eines Akteurs dargestellt • swimlanes können mit Akteuren, Klassen und Packages verknüpft werden • eine Aktivität wird durch ein "Rechteck" mit konvex abgerundeten Seiten visualisiert, sie enthalten eine Beschreibung der internen Aktion • von der Aktion gehen die Transitionen aus, die den	Zusicherungen, aber mindestens mit ihrem Namen aufgeführt • Operationen können ebenfalls durch Parameter, Initialwerte, Zusicherungen usw. beschrieben werden, aber mindestens mit ihrem Namen aufgeführt • die gefundenen Klassen werden durch Linien miteinander verbunden • diese Linien stellen die Elemente Assoziation, Aggregation, Komposition und Vererbung dar • eine besondere Assoziation ist die Aggregation, die durch eine Raute an der Linie dargestellt wird • sie gibt an, dass eine Klasse in einer anderen Klasse "enthalten" ist (Ist-Teil-von-Beziehung) • die Komposition ist eine stärkere Form der Aggregation, die durch eine ausgefüllte Raute dargestellt wird • Komposition gibt an,	Objektsymbol bzw. als Rechteck und einer senkrechten, gestrichelten Linie dargestellt • Objektkonstruktion kann durch ein Kreuz am Ende der Lebenslinie dargestellt werden • Objektzustände werden auf der Lebenslinie dargestellt • Objektkonstruktion kann durch Nachricht dargestellt werden • Antworten auf Nachrichten sind optional • Nachrichten werden durch waagerechte Pfeile zwischen den Objektlebenslinien beschrieben • auf diesen Pfeilen werden die Nachrichtennamen in der Form: nachricht (argumente) notiert • Objekte, die gerade aktiv an Interaktionen beteiligt sind, werden durch einen Balken auf ihrer Lebenslinie	Aktionen (gesendete Botschaften) mit einem Zustandswechsel verbunden sind • im Zustandsdiagramm werden die Zustände als abgerundete Rechtecke visualisiert, die mit Pfeilen verbunden sind • auf den Pfeilen stehen die Transitionen • Startzustand ist ein gefüllter Kreis, die Endzustände sind leere Kreise mit einem kleineren gefüllten Kreis in der Mitte

		Abschluss der internen Aktion und den Übergang zur nächsten Aktivität darstellen	dass eine Klasse aus einer anderen besteht • an einer Assoziation können noch Multiplizitäten, d.h. Zahlen oder Zahlenbereiche angegeben werden • diese bestimmen die Anzahl der Objekte, die miteinander in Beziehung stehen • jede Assoziation kann eine Richtung besitzen: hierfür wird ein Pfeil am Ende der Assoziation angebracht • Zugriffe können dann nur in Pfeilrichtung erfolgen (Navigationsfähigkeit) • die Vererbung stellt eine Verallgemeinerung von Eigenschaften dar - sie wird auch als Spezialisierung und Generalisierung oder "Ist-ein" Beziehung bezeichnet	gekennzeichnet	
Welche Aspekte sind nicht ersichtlich?	• keine Beschreibung eines Verhaltens oder von Abläufen des Systems	• keine Beschreibung der Anforderungen an das System • keine Beschreibung des	• keine Beschreibung der Anforderungen an das System • keine Beschreibung der	• keine Beschreibung der Anforderungen an das System • keine Beschreibung des	• keine Beschreibung der Anforderungen an das System • keine Beschreibung des

	• keine Aussage über Systemdesign	logischen Aufbaus des Systems	Interaktionen und Abläufe im System	logischen Aufbaus des Systems	logischen Aufbaus des Systems
Beispiele	• **Kfz-Versicherung** • Verschiedene Use-Cases für eine Kfz-Versicherung: o Versicherung abschließen o Versicherung kündigen o Versicherter zahlt Prämie o Sachbearbeiter behandelt Schadensfall o Sachbearbeiter ändert Konditionen	• **Kfz-Versicherung** • Szenario "Schadensfall": Ein Schadensfall tritt ein. Der Sachbearbeiter wird informiert und bearbeitet den Fall. Der Vorfall wird in den Versicherungsunterlagen vermerkt. Bei Verschulden durch den Versicherungsnehmer wird die monatliche Prämie erhöht.	• Siehe grafische Ergänzung	• **Kfz-Versicherung** • Szenarien „Versicherung abschließen" und "Versicherung auflösen"	• **Kfz-Versicherung** • Szenario "Schadensfall": Ein Schadensfall tritt ein. Der Sachbearbeiter wird informiert und bearbeitet den Fall. Der Vorfall wird in den Versicherungsunterlagen vermerkt. Bei Verschulden durch den Versicherungsnehmer wird die monatliche Prämie erhöht.

Tabelle 2: Vergleich der UML-Diagramme

Grafische Ergänzungen zu den Beispielen:

Anwendungsfalldiagramm:

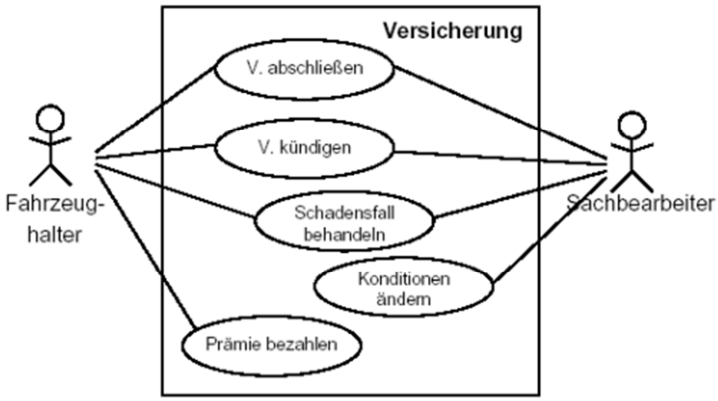


Abbildung 12: Beispiel Anwendungsfalldiagramm

Aktivitätsdiagramm:

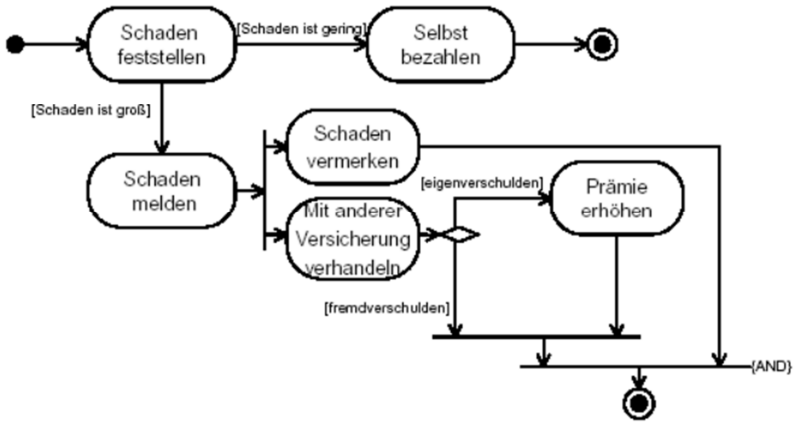


Abbildung 13: Beispiel Aktivitätsdiagramm

Klassendiagramm:

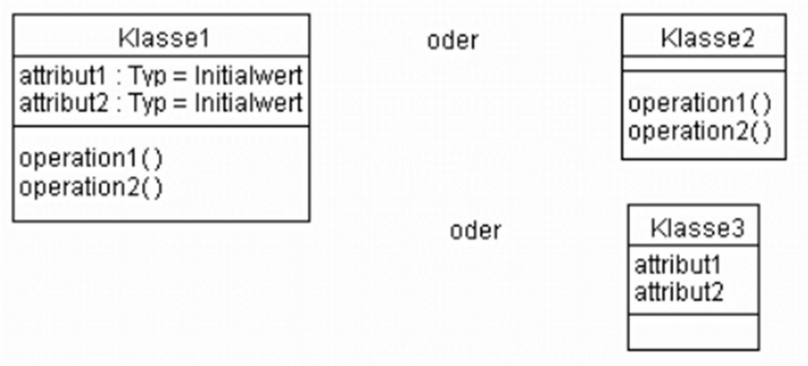


Abbildung 14: Beispiel Klassendiagramm

Sequenzdiagramm:

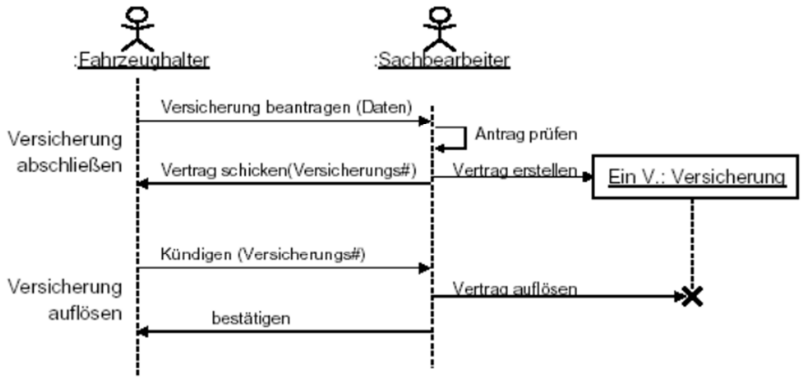


Abbildung 15: Beispiel Sequenzdiagramm

Zustandsdiagramm:

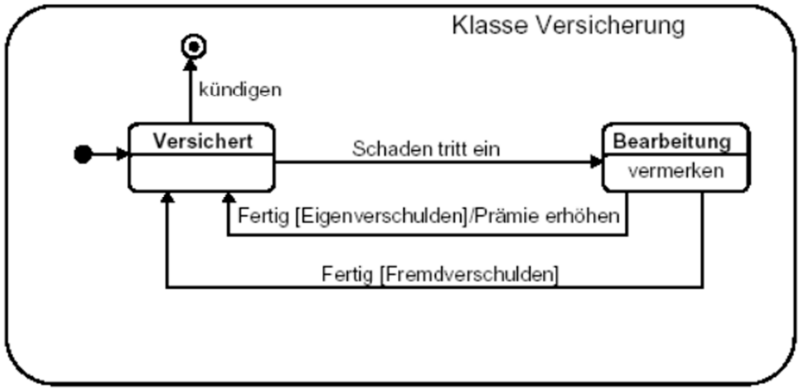


Abbildung 16: Beispiel Zustandsdiagramm

Darauf aufbauend, kann die Frage hinsichtlich der notwendigen Konkretisierung eines bestimmten Diagrammtyps ebenfalls treffend beantwortet werden:

„Welches Diagramm muss innerhalb des Designprozesses der Software konkretisiert werden?“

Das Klassendiagramm muss im Zuge des Designprozesses konkretisiert werden, da hier Aktivitäten mit den zugehörigen Abläufen und Prozessen dargestellt werden. Die Konkretisierung ist notwendig, da das Klassendiagramm, auf der Basis des Quellcodes, das Softwareprojekt in seiner Gesamtheit beschreibt. Aus verschiedenen Aspekten, zum Beispiel aus Sicht der Analytiker, lassen sich Notwendigkeiten widerspiegeln bzw. korrigieren. Ebenso können Programmierer ein vollumfängliches Bild der Software und derer integrierten Klassen erhalten. Des Weiteren stellt das Klassendiagramm neben den Vererbungsklassen auch die Beziehungen der Klassen untereinander dar und gibt somit einen Überblick zur Funktionsweise und zum Aufbau der Software bzw. des Softwaresystems.

5. Installation

Dieses Kapitel beschreibt die vollumfängliche Installation des Software Systems.

Die nachfolgenden Anweisungen und Beschreibungen sind so konzipiert, dass die Installation ohne spezielles Expertenwissen erfolgen kann. Dennoch wird eine Installation durch entsprechendes Personal mit Basiswissen in den Bereichen Linux Systeme, Linux Shell und MySQL Datenbank Administration empfohlen.

Aufgrund der Komplexität des Software Systems kommen diverse Softwareprodukte zum Einsatz, welche im Vorfeld unter Betrachtung von Lizenzbedingungen und möglicher, anfallender Kosten geprüft und ausgewählt wurden. Konkret handelst es sich um die folgenden Komponenten:

- Apache Tomcat in Version 7 (Apache License Version 2.0)
- Apache HTTP Server in Version 2.2 (Apache License Version 2.0)
- MySQL Database Server (General Public License - GPL)
- Oracle Java Development Kit (JKD) in Version 1.7 (Oracle Binary Code License - BCL)

Als Basis für das Software System kommt ein Linux Betriebssystem (64 Bit) zum Einsatz. Dieses kann grundsätzlich frei gewählt werden. Diese Installationsbeschreibung bezieht sich allerdings ausschließlich auf Debian Wheezy. Folglich können bei anderen, zum Einsatz kommenden Systemen, spezifische Parameter abweichen.

5.1. Systemvoraussetzungen

Um MiZe in einer kleinen bis mittleren Umgebung (50 aktive, 1000 registrierte Nutzer) zu betrieben, wird ein Serversystem mit folgenden Mindestanforderungen empfohlen:

- Quad Core CPU (4x 2,5 GHz)
- 8GB RAM
- 25GB HDD Speicher

Für größere Landschaften ist eine individuelle Skalierung notwendig. Diese sollte auf Basis einer Analyse des speziellen Nutzerverhaltens erfolgen.

5.2. Installation der Software

Der Einsatz von Open Source Software und Betriebssystem ermöglichen den effektiven Einsatz des Paketmanagementtools APT (Advanced Packaging Tool) um die Basiskomponenten zu installieren. Dies wird anhand der grau hinterlegten und umrahmten Zeilen beschrieben.

Die Installationsbeschreibung basiert auf der Verwendung der bash-Shell. Zum Ausführen der genannten Befehle benötigt der entsprechende Nutzer, unter dem das Software System installiert wird, privilegierte Nutzerrechte (sudo).

Installation Apache HTTP Server:

Die Installation des Apache HTTP Servers erfolgt durch folgenden Befehl:

```
sudo apt-get install apache2 apache2-doc
```

Installation MySQL Database Server:

Die Installation des MySQL Database Servers erfolgt durch folgenden Befehl:

```
sudo apt-get install mysql-server
```

Während der Installation wird ein root-Datenbankbenutzer angelegt. Das hierbei vergebene Passwort sollte sicher verwahrt werden, da es im weiteren Verlauf der Installation erneut benötigt wird. Aus Sicherheitsgründen wird die Verwendung von sicheren Passwörtern gemäß der konzerninternen Passwortvorgaben empfohlen.

Installation Apache Tomcat und Oracle JDK:

Apache Tomcat und Oracle JDK werden zusammen mit dem Softwarepaket geliefert. Dies hat den Vorteil, dass keine Abhängigkeiten zu der Versionsverwaltung des Betriebssystemdistributors bestehen. Neue Versionen der Komponenten können folglich individuell getestet und bei Bedarf in den Betrieb überführt werden.

Zusätzlich enthält das Softwarepaket bereits optimierte Konfiguration, um einen reibungslosen, effizienten und zu hohen Anteilen automatisierten Installations- und Konfigurationsablauf zu ermöglichen.

Das gelieferte Softwarepaket (MiZe.zip) muss zunächst mit einem geeigneten Tool (z.B. FileZilla oder WinsCP) auf das Zielsystem übertragen werden. Dort wiederum wird es mit Hilfe der folgenden Befehle in das Verzeichnis „/opt“ kopiert und anschließend entpackt:

```
cp -a MiZe.zip /opt
cd /opt
unzip MiZe.zip
```

Nach dem erfolgreichen Auspacken des Archives resultiert folgende Verzeichnisstruktur:

```
/opt/
   |-- apache-tomcat-7.0.61
   |-- jdk -> jdk1.7.0_79/
   |-- jdk1.7.0_79
   |-- tomcat -> apache-tomcat-7.0.61/
   |-- apache-http
```

5.3. Konfiguration der Software

Konfiguration Apache HTTP Server:

Für den Webserver wurde ebenfalls eine Musterkonfiguration erstellt, die nur noch in das korrekte Zielverzeichnis kopiert werden muss. Mit Hilfe der nachfolgenden Befehle können sowohl die bereits vorhandenen Beispielkonfigurationen des Webservers gelöscht und die eigentliche Musterkonfiguration kopiert werden:

```
rm -f /etc/apache2/sites-enabled/*
```

```
cp -a /opt/apache-http/mize /etc/apache2/sites-enabled/
```

Konfiguration MySQL Database Server:

Es ist zwingend notwendig, eine neue MySQL Datenbank inklusive der erforderlichen Datenbank Schemas anzulegen. Folgende Befehle müssen dazu abgesetzt werden:

```
mysql -u root -p

CREATE DATABASE mize;
CREATE USER 'ezim'@'localhost' IDENTIFIED BY '*PASSWORD*';
GRANT SELECT, INSERT, UPDATE, DELETE ON mize.* TO
'ezim'@'localhost' IDENTIFIED BY '*PASSWORD*';
FLUSH PRIVILEGES;
```

Das mitgelieferte „mize.sql“ Skript muss im Anschluss ausgeführt werden, um die erforderliche Datenbankstruktur zu erzeugen.

Konfiguration Apache Tomcat:

Zunächst muss das angefertigte Startskript für den Apache Tomcat Server an die korrekte Stellung kopiert werden. Dazu muss folgender Befehl ausgeführt werden:

```
cp /opt/apache-tomcat-7.0.61/tomcat /etc/init.d/
```

Anschließend wird das Startskript mit den entsprechenden Runlevel des Systems verlinkt, um ein automatisiertes Starten und Stoppen bei Systemstart zu ermöglichen:

```
cd /etc/rc3.d/
ln -s ../init.d/tomcat S90tomcat
cd /etc/rc0.d/
ln -s ../init.d/tomcat K01tomcat
```

Konfiguration Oracle JDK:

Um das ausgelieferte JDK als Standardinstanz auf dem System zu etablieren, sind folgende Befehle notwendig:

```
echo “export JAVA_HOME=/opt/jdk1.7.0_79/” >> /etc/bash.bashrc
echo “export PATH=$PATH:/opt/jdk1.7.0_79/bin” >> /etc/bash.bashrc
```

5.4. Starten und Stoppen der Applikationen

Grundsätzlich sind alle Services so konfiguriert, dass sie automatisch beim Systemstart gestartet und beim Herunterfahren gestoppt werden.

Für eventuell anfallende Wartungsarbeiten am System wird folgende Reihenfolge zum manuellen Starten und Stoppen des Systems empfohlen:

```
/etc/init.d/mysql.server start
/etc/init.d/tomcat start
/etc/init.d/apache2 start
```

```
/etc/init.d/apache2 stop
/etc/init.d/tomcat stop
/etc/init.d/mysql.server stop
```

5.5. Backupstrategien

MiZe selbst speichert keine temporären Dateien auf Filesystemebene, was die Auswahl einer geeigneten Backupmethode sehr flexibel gestaltet. Folglich ist eine Sicherung des Filesystems theoretisch nur bei Versionswechseln erforderlich. Aufgrund der Erfahrungen während der Entwicklungsphase wird dennoch eine regelmäßige, automatisierte Sicherung des Systems empfohlen. Für die Datenbank kann auf das von MySQL bereitgestellte Tool „mysqldump“ zurückgegriffen werden.

```
mysqldump -u root -p[root_password] [database_name] > dbdump.sql
z.B.
mysqldump -u root -p[root_password] mize >
Mize_Dump_YYYYMMDDHHMM.sql
```

Hinweis: Die hier getätigten Aussagen sind als Empfehlungen anzusehen. Die letztendliche Entscheidung für eine passende Backupstrategie sollte in Zusammenarbeit mit den Verantwortlichen des Rechenzentrumbetriebs ausgearbeitet und abgesprochen werden.

6. Betrieb

In der aktuellen Version bietet das Softwareprodukt MiZe bereits eine umfangreiche Ausprägung an software-seitig implementierten Prozessen. Für den Regelbetrieb sind diese Prozesse jedoch noch nicht ausreichend, um einen vollständig, fachlich automatisierten Betrieb zu ermöglichen.

Für solche Fälle werden „Expert Advices“ zu Verfügung gestellt, damit der 2nd Level Support direkt unterstützen kann. Für Fälle, die hier nicht beschrieben sind, ist der 3rd Level Support zu kontaktieren.

Kundenanforderungen:

- Benutzer löschen
- Fahrzeug löschen
- Benutzer bekommt eine Fehlermeldung beim Löschen einer Fahrt

Für diese Anforderungen können mit Hilfe des in Kapitel „2. Architektur“ beschriebenen Datenbankmodells entsprechende SQL-Skripte entwickelt werden.

Supportfälle:

- Homepage kann nicht aufgerufen werden
    - Server- bzw. Netzwerkinfrastruktur verfügbar?

- Tomcat-Prozess verifizieren und gegebenenfalls (neu) starten
- „catalina.out“ Log-Datei des Tomcat verifizieren und gegebenenfalls Tomcat-Prozess neu starten und/oder 3rd Level Support kontaktieren

- Benutzer kann sich nicht einloggen
  - Verfügbarkeit der MySQL Datenbank prüfen und gegebenenfalls (neu) starten

7. Test

Um während der Realisierungsphase den Fortschritt des Backend zu überwachen und Regressionen der einzelnen Funktionen (z.B. durch Hinzufügen neuer Funktionen) zu verhindern, wurden regelmäßig API (application programming interface) Tests durchgeführt.

Zu diesem Zwecke wurde das Tool „SoapUI“ eingesetzt (http://www.soapui.org/).
SoapUI erlaubt es, APIs und Webservices automatisch auf verfügbare Funktionen zu scannen, diese aufzurufen und die Rückmeldung der Services auf vom Benutzer definierte Eigenschaften zu überprüfen. Aufgrund des engen Zeitrahmens des Softwareentwicklungsprojektes, der hohen Benutzerfreundlichkeit, geringen Einarbeitungszeit und weltweiten Community passte SoapUI optimal für die Testunterstützung.

Das eigentliche Testverfahren in SoapUI soll hier anhand eines beispielhaft herausgegriffenen Testfalls und Tests verdeutlicht werden. Der so genannte „CRUD“-Testfall (Create, Read, Update, Delete) soll die Funktionen „Fahrt erstellen“, „Fahrt verändern“ und „Fahrt löschen“ testen. Diese einzelnen Testfälle lassen sich gut zusammenfassen, da sie seinerseits thematisch zusammengehören und andererseits mit der „Trip UUID“, dem Identifikator für Fahrten innerhalb MiZe, arbeiten.

SoapUI fast einzelne Tests in „Testcases“ zusammen, welche wiederum in „Testsuites“ gebündelt werden. Testcases und Tests werden sequenziell abgearbeitet und können untereinander in Beziehung stehen. Im nachfolgenden Testcase wird eine neue Fahrt generiert. Deren UUID wird in Schritt 2 ab die weiteren Test übergeben, um Funktionen an der Fahrt zu testen:

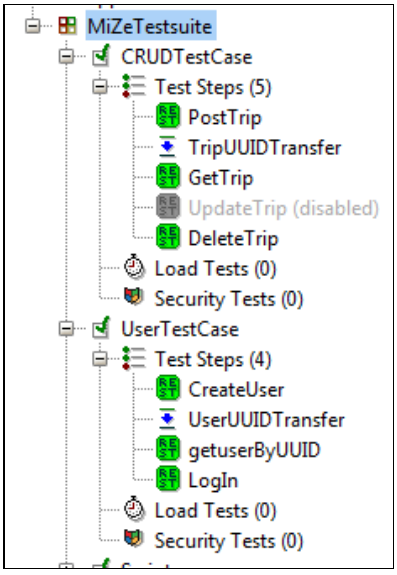


Abbildung 17: MiZe SoapUI

Eine detaillierte Betrachtung des „PostTrip“ Tests zeigt, dass dieser aus drei Teilen besteht:

1. Anfrage an das Backend (RAW Request bzw. JSON Inhalt):

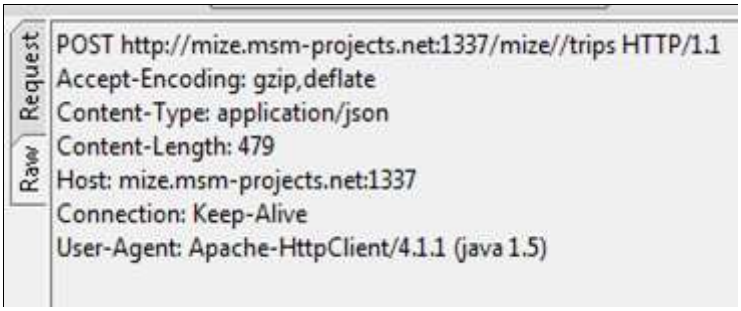


Abbildung 18: MiZe RAW Request

Abbildung 19: MiZe JSON Inhalt

2. Antwort / Response des Services (OK – Meldung inkl. UUID der angelegten Fahrt):

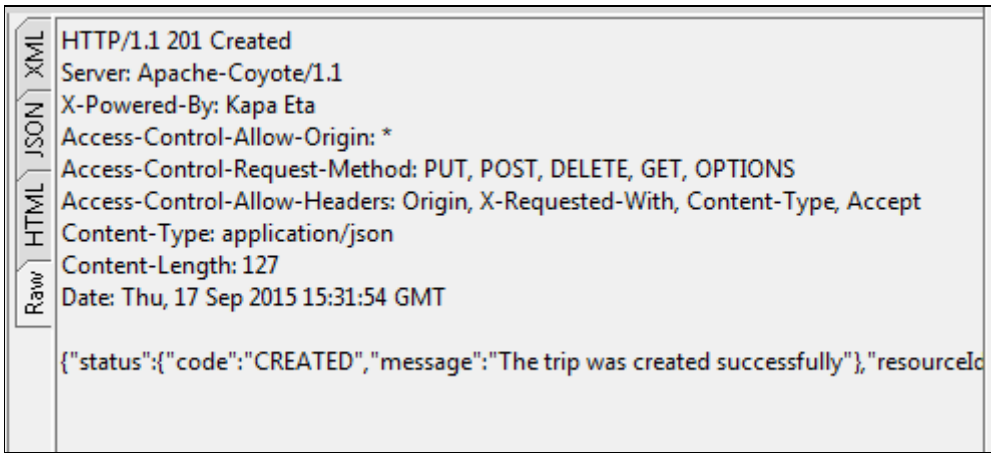


Abbildung 20: MiZe RAW Response

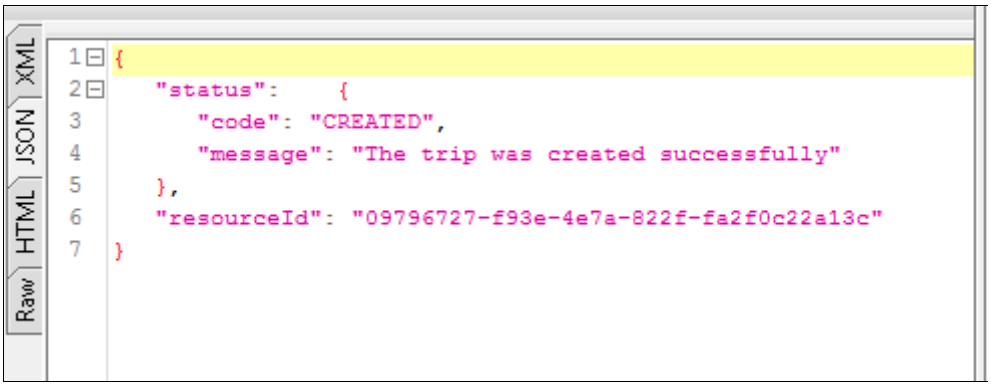


Abbildung 21: MiZe JSON Response

3. Auswertung der Antwort / Response (Prüfung, ob valider http Statuscode zurückgegeben und ein Feld namens „resourceID“ übermittelt wird):

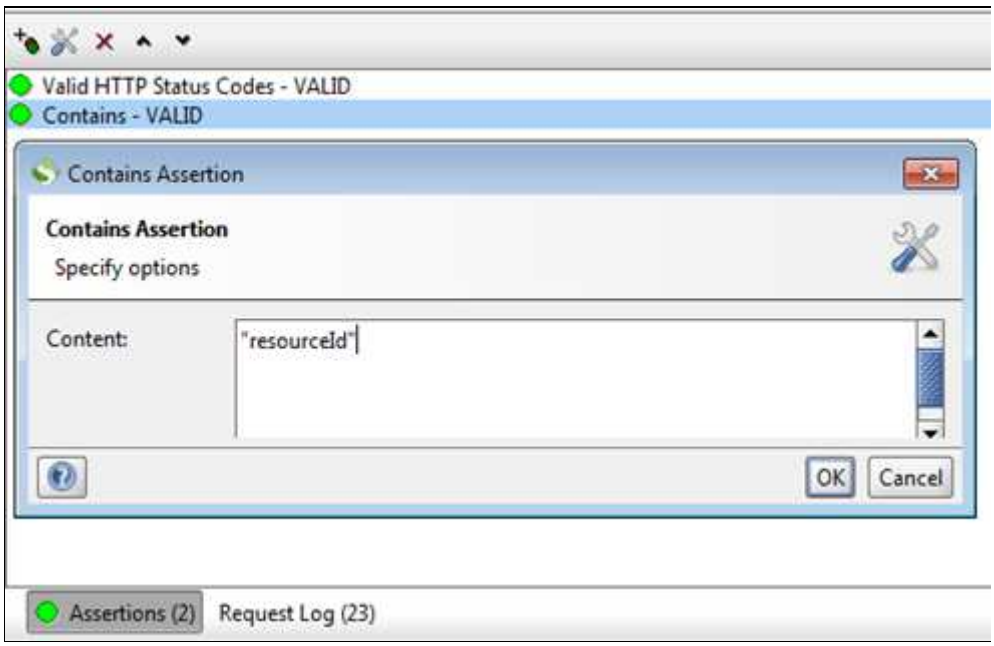


Abbildung 22: MiZe Validierung

Das komplette SoapUI Testsuite kann dem Anhang entnommen und für spätere Tests, die im Rahmen einer möglichen Wirkbetriebsüberführung notwendig sind, verwendet werden.

8. Wirkbetriebsüberführung

Da MiZe im Rahmen der Entwicklung zunächst auf einem separaten, externen Serversystem außerhalb der HfTL installiert und betrieben wurde, sollten folgende Punkte bei der Migration auf eine produktive Umgebung innerhalb der HfTL berücksichtigt werden:

- Das System kann, wie in Kapitel 5 beschrieben, problemlos auf einer neuen Umgebung installiert werden. Für die Migration sämtlicher Nutzerdaten muss die Datenbank mit Hilfe von MySQL Ex- und Import Funktionen migriert werden.
- Wie in Kapitel 5 beschrieben, sollte ein geeignetes Backupkonzept in Zusammenarbeit mit den Verantwortlichen des Rechenzentrumbetriebs ausgearbeitet werden.
- Als zusätzliche Unterstützung sollten alle systemrelevanten Dienste in einer Monitoring-Lösung aufgenommen bzw. integriert werden.
- Als zusätzliche Unterstützung sollten eine Logfile- und Dateisystemüberprüfung (z.B. mit Hilfe einer Monitoring-Lösung) etabliert werden.

All diese Schritte sollten zunächst im Rahmen einer separaten Generalprobe auf einer zusätzlichen Integrationsumgebung durchgespielt werden. Im Anschluss sind die „CRUD“ Tests, wie in Kapitel 7 beschrieben, durchzuführen.

Erst nachdem die Generalprobe und die Tests erfolgreich durchgeführt wurden, kann das Wartungsfenster für die eigentliche Wirkbetriebsüberführung geplant und genehmigt werden. Die betroffenen Nutzer sind im Vorfeld über die geplanten Aktivitäten und den Wechsel des Systems zu informieren.

A. Anhang

MiZe – Grobkonzept:

Hochschule für Telekommunikation Leipzig

MiZe Grobkonzept

Softwareentwicklungsprojekt HfTL Mitfahrzentrale

Impressum

Herausgegeben von

Softwareentwicklungsprojekt HfTL Mitfahrzentrale – MiZe
J. Dümig, T. Kilian, F. Seidel, B. Schmitz, D. Tonn, C. Wiegel
Hochschule für Telekommunikation Leipzig, Gustav-Freytag-Straße 43-45, 04277 Leipzig

Dateiname	Dokumentnummer	Dokumentname
MiZe_concept.docx	1	MiZe Grobkonzept
Version	**Zuletzt geprüft**	**Status**
1.0	29.04.2015	Final
Autor	**Inhaltlich geprüft von**	**Veröffentlicht von**
D. Tonn	T. Kilian	D. Tonn
Ansprechpartner	**Telefon / FAX**	**E-Mail**
D. Tonn	nur per E-Mail	s134325@hft-leipzig.de

Kurzbeschreibung

Dieses Dokument dient der Definition der Rahmenbedingungen für das Softwareentwicklungsprojekt „HfTL Mitfahrzentrale – MiZe", das im Rahmen des Moduls Softwareengineering durch die oben genannten Studenten der Hochschule für Telekommunikation Leipzig im Sommersemester 2015 erbracht wird.

Inhalt

1. Adressaten des Softwaresystems ..... - 1 -
2. Funktionen und Berechtigungen der Systembenutzer ..... - 1 -
3. Geschäftsprozessinformationen ..... - 2 -
4. Zusatzfunktionen ..... - 3 -
5. Definition Anwendungsfall ..... - 3 -
6. Beschreibung von Anwendungsfällen ..... - 3 -

1. Adressaten des Softwaresystems

Durch die im Softwareentwicklungsprojekt „HfTL Mitfahrzentrale – MiZe“ vorgegebene 3-Tier Architektur, bestehend aus

- Tier 1: MySQL Datenbank Backend
- Tier 2: Java Webservice Backend
- Tier 3: HTML5 User Interface Frontend

lassen sich die nachfolgenden User- bzw. Personengruppen ableiten:

- **Systemadministratoren**

  Systemadministratoren sind für die Verwaltung und Konfiguration der entsprechend zum Einsatz kommenden Hard- und Softwarekomponenten verantwortlich.

- **Entwickler**

  Entwickler sind für die initiale Implementierung der Software und potenzielle Upgrades notwendig.

- **Endanwender**

  Endanwender des Systems sind alle Studenten der Hochschule für Telekommunikation Leipzig und bilden folglich die entsprechende Usergroup für das eigentliche Userinterface.

2. Funktionen und Berechtigungen der Systembenutzer

Die in Kapitel 1 aufgelisteten Adressaten des Softwaresystems lassen sich durch die folgenden Funktionen und Berechtigungen unterscheiden:

- **Systemadministratoren**

  Systemadministratoren benötigen aufgrund ihrer Verwaltungs- und Konfigurationsfunktion permanenten, uneingeschränkten Vollzugriff auf alle entsprechenden Hard- und Softwarekomponenten des Systems.

- **Entwickler**

  Entwickler benötigen während der Implementierungs- und Testphase ebenfalls uneingeschränkten Zugriff auf die Softwarekomponenten des Systems. Sobald das System in den produktiven Betrieb übergeht, sind diese Berechtigungen entsprechend zu entfernen.

- **Endanwender**

Endanwender benötigen spezielle, auf ein Szenario oder einen Anwendungsbereich abgestimmte Funktionen und Berechtigungen. Endanwender, die über das Softwaresystem Fahrten anbieten, sind auf folgende Funktionen und Berechtigungen angewiesen:

- Registrieren an der Applikation
- Anmelden an der Applikation
- Abmelden an der Applikation
- Anlegen von Fahrten
- Löschen von eigenen Fahrten
- Modifizieren von eigenen Fahrten
- Sichten von Mitfahranfragen
- Annehmen von Mitfahranfragen
- Ablehnen von Mitfahranfragen
- Bewerten von Mitfahrern
- Stornofunktion für eigene, angelegte Fahrten

Endanwender, die über das Softwaresystem Fahrten suchen und als Mitfahrer in Anspruch nehmen wollen, benötigen folgende Funktionen und Berechtigungen:

- Registrieren an der Applikation
- Anmelden an der Applikation
- Abmelden an der Applikation
- Suchen von Fahrten
- Erstellen von Mitfahranfragen
- Zurückziehen von Mitfahranfragen
- Bewerten von Fahrern

3. Geschäftsprozessinformationen

Die nachfolgend aufgelisteten Informationen eines Endanwenders sind erforderlich, um die im System integrierten Geschäftsprozesse (z.B. Anbieten einer Fahrt) zu ermöglichen.

Login- und Kontaktinformationen:

- Nachname
- Name
- Matrikelnummer
- E-Mail Adresse
- Mobilfunknummer

Fahrzeug- und Reiseinformationen:

- Fahrzeugtyp
- Anzahl der freien Plätze
- Fahrstil
- Preis

- Abfahrtsort
- Zielort

4. Zusatzfunktionen

Die nachstehend formulierten Funktionen repräsentieren Anforderungen, die im Rahmen der Entwicklung des Softwareprojektes umgesetzt werden können, jedoch nicht zwingend erforderlich sind. Sie stellen vielmehr optionale, für den Endanwender nützliche Features dar.

- HfTL LDAP-Anbindung
- Mobile-Device-Support
- Smartphone Applikation
- Geschlechtergetrennte Fahrten
- Vielfahrer Bonussystem
- Integration von Online Zahlungsdiensten
- SMS Benachrichtigung

5. Definition Anwendungsfall

Ein Anwendungsfall bündelt alle möglichen Szenarien, die eintreten können, wenn ein Akteur versucht, mit Hilfe des betrachteten Systems ein bestimmtes fachliches Ziel zu erreichen (z.B. das Anbieten einer Fahrt). Er beschreibt, was inhaltlich beim Versuch der Zielerreichung passieren kann und abstrahiert von konkreten technischen Lösungen. Das Ergebnis des Anwendungsfalls kann ein Erfolg oder Fehlschlag/Abbruch sein.

6. Beschreibung von Anwendungsfällen

Die im Anschluss festgehaltenen Anwendungsfälle dienen der beispielhaften Beschreibung ausgewählter Szenarien anhand eines vorgegebenen Musters.

Anwendungsfall – Fahrt anbieten:

use case:
Fahrt anbieten

actors:
Fahrer

precondition:
gültiges Profil

main flow:
Fahrer bietet Fahrt mit Startort, Abfahrtzeit, Anzahl von freien Plätzen, Preis und Fahrstil an

alternative flow:
Systemadministrator (1-Level Support)stellt Fahrt für Fahrer ein

postcondition:
Fahrt wird in der Applikation publiziert

exceptional flow
Fahrt ist als inaktiv markiert

postcondition:
Ergebnis der Ausnahmesituationen

end:
Fahrt publizieren

Anwendungsfall – Registrierung :

use case:
Registrieren

actors:
Neuer, nicht angemeldeter Benutzer

precondition:
none

main flow:

User gibt seine Daten (Name, Vorname, Matrikelnummer, E-Mail) ein
User klickt auf Registrieren
System legt Account an und setzt Confirmation Pending Flag
User bekommt eine E-Mail mit Bestätigungslink
User klickt den Link und besucht die Webseite
System entfernt Confirmation Pending Flag

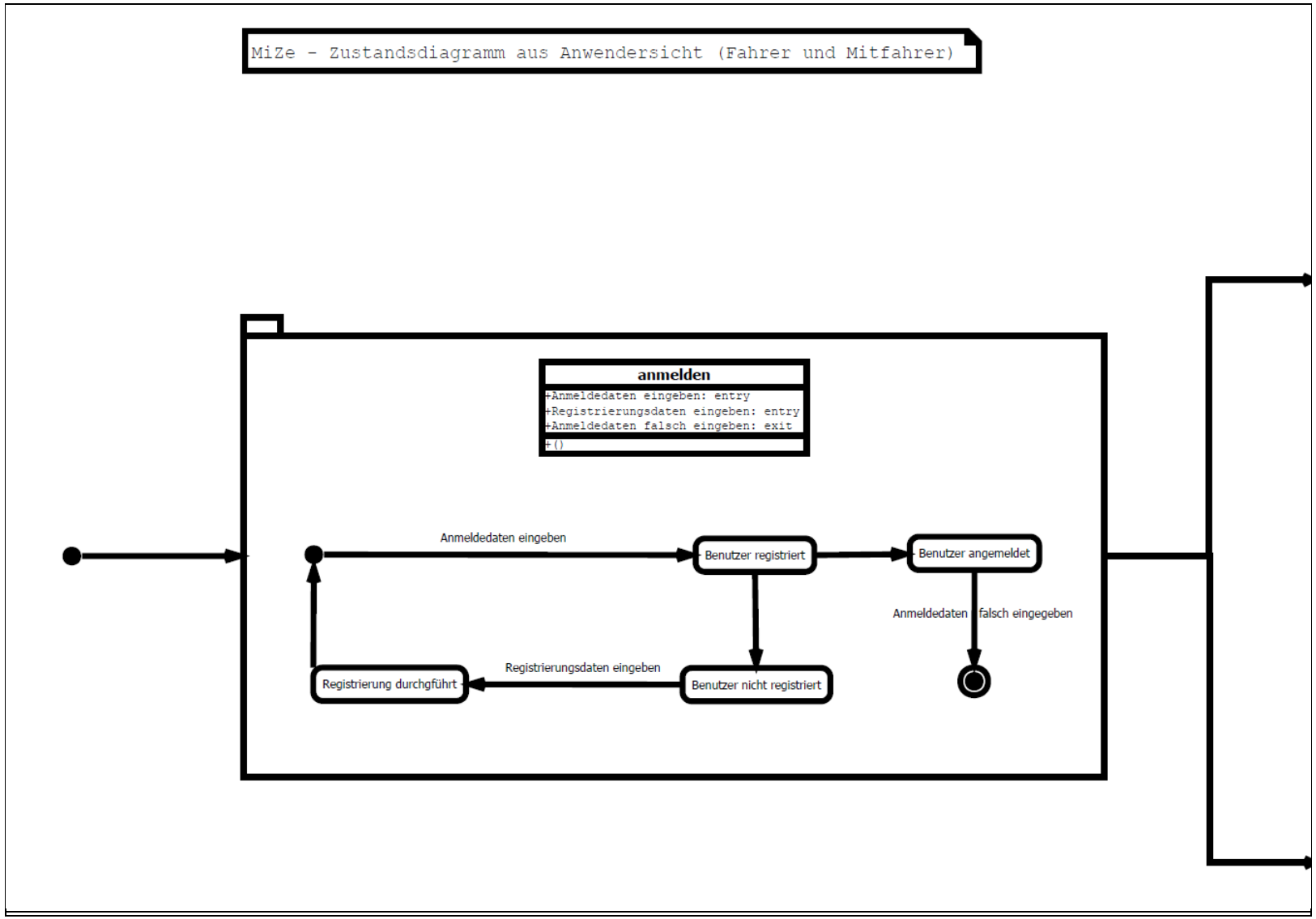
alternative flow:
Keine Bestätigung
Beschreibung des alternativen Ablaufs des Anwendungsfalls

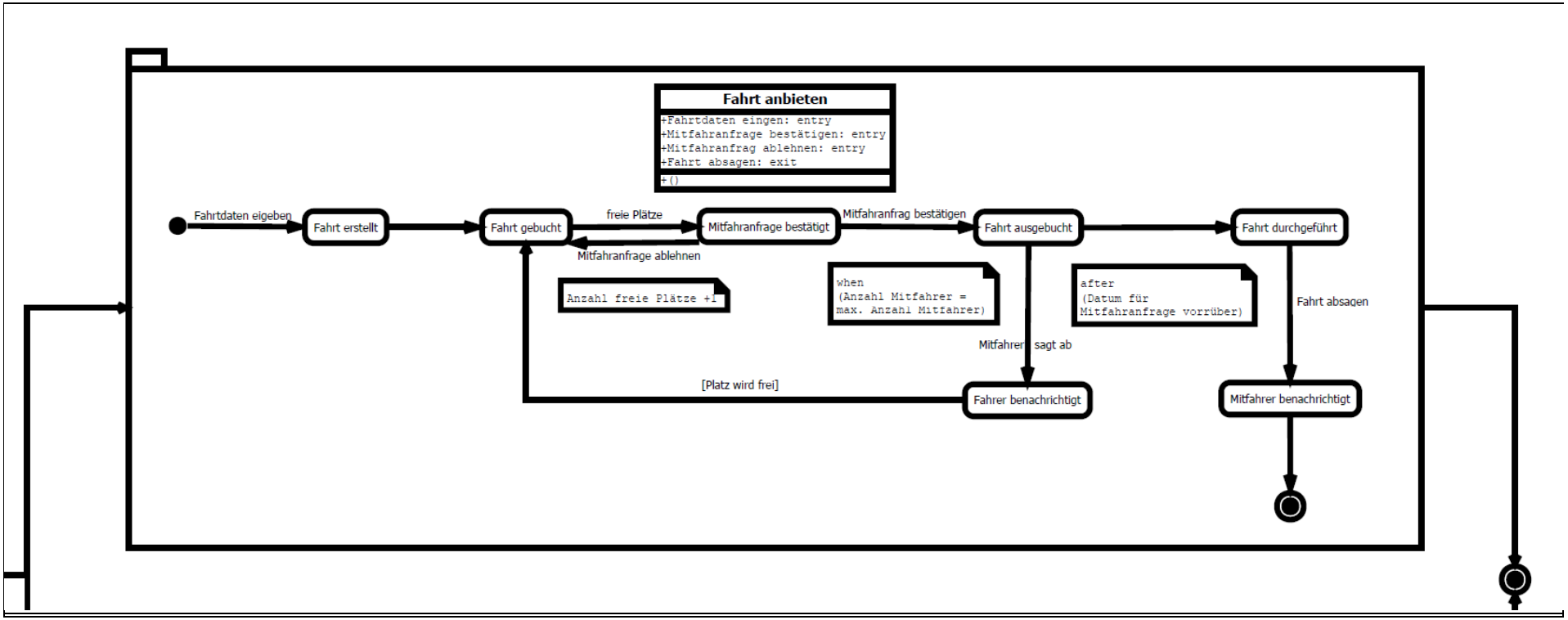
User gibt seine Daten (Name, Vorname, Matrikelnummer, E-Mail) ein
User klickt auf Registrieren
System legt Account an und setzt Confirmation Pending Flag
User bekommt eine E-Mail mit Bestätigungslink
Das System erhält binnen zwei Wochen keine Bestätigung

Der Account wird wieder gelöscht

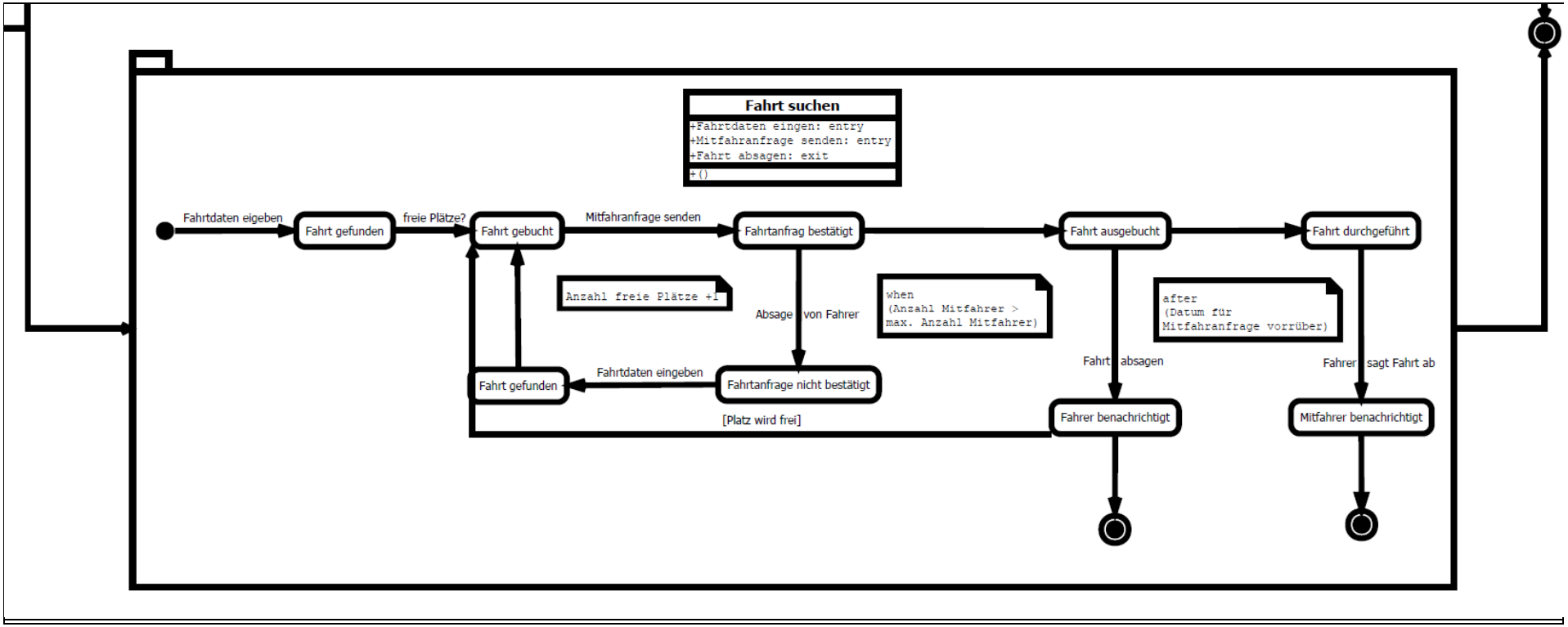
postcondition: Useraccount ist angelegt
exceptional flow: Keine gültige Matrikelnummer Beschreibung der Ausnahme
User wird informiert, dass das Programm momentan nur für HfTL Studenten gedacht ist. **postcondition:** Es wird kein Account erstellt. **end:** Teilnehmer anmelden

MiZe – Zustandsdiagramm:

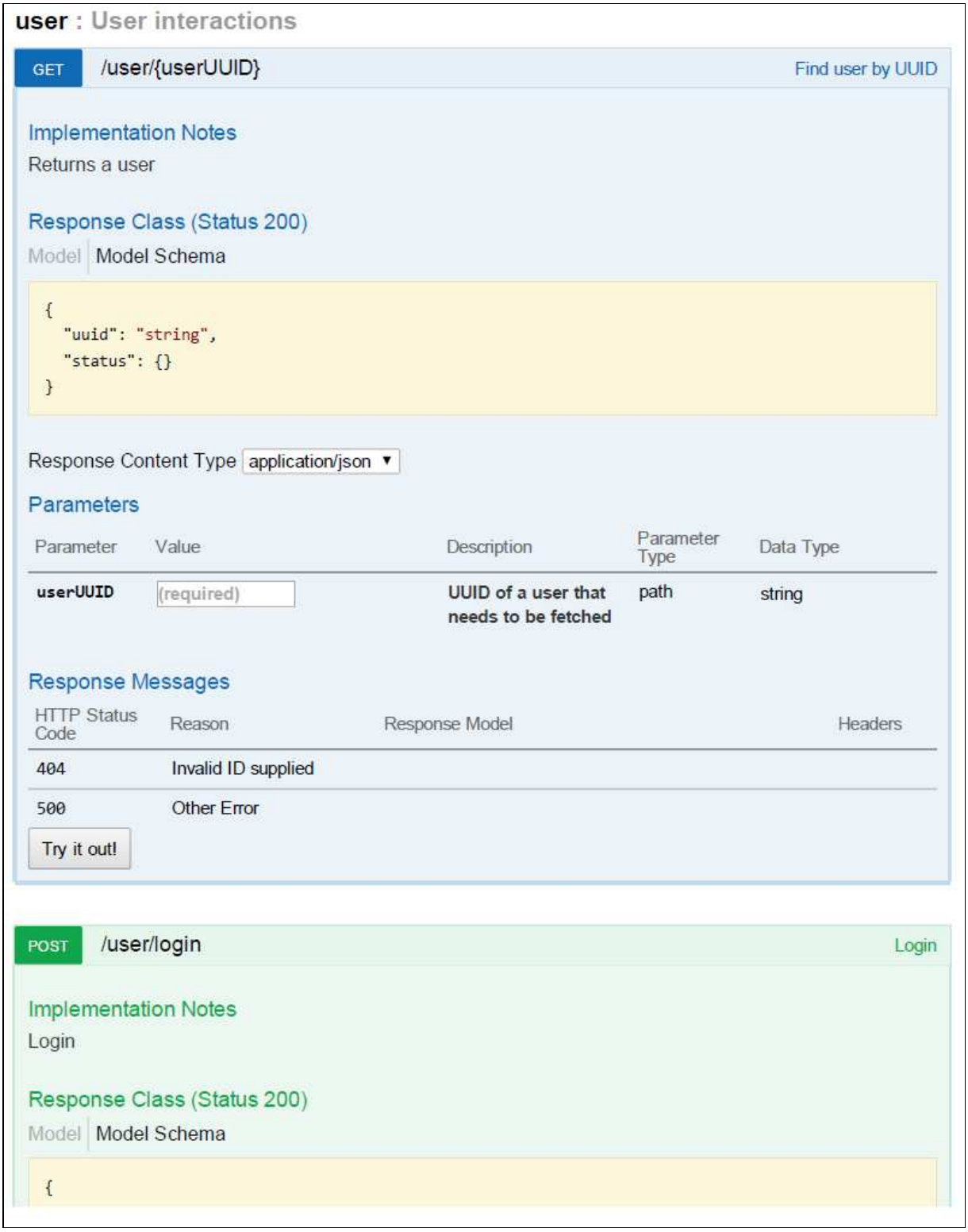


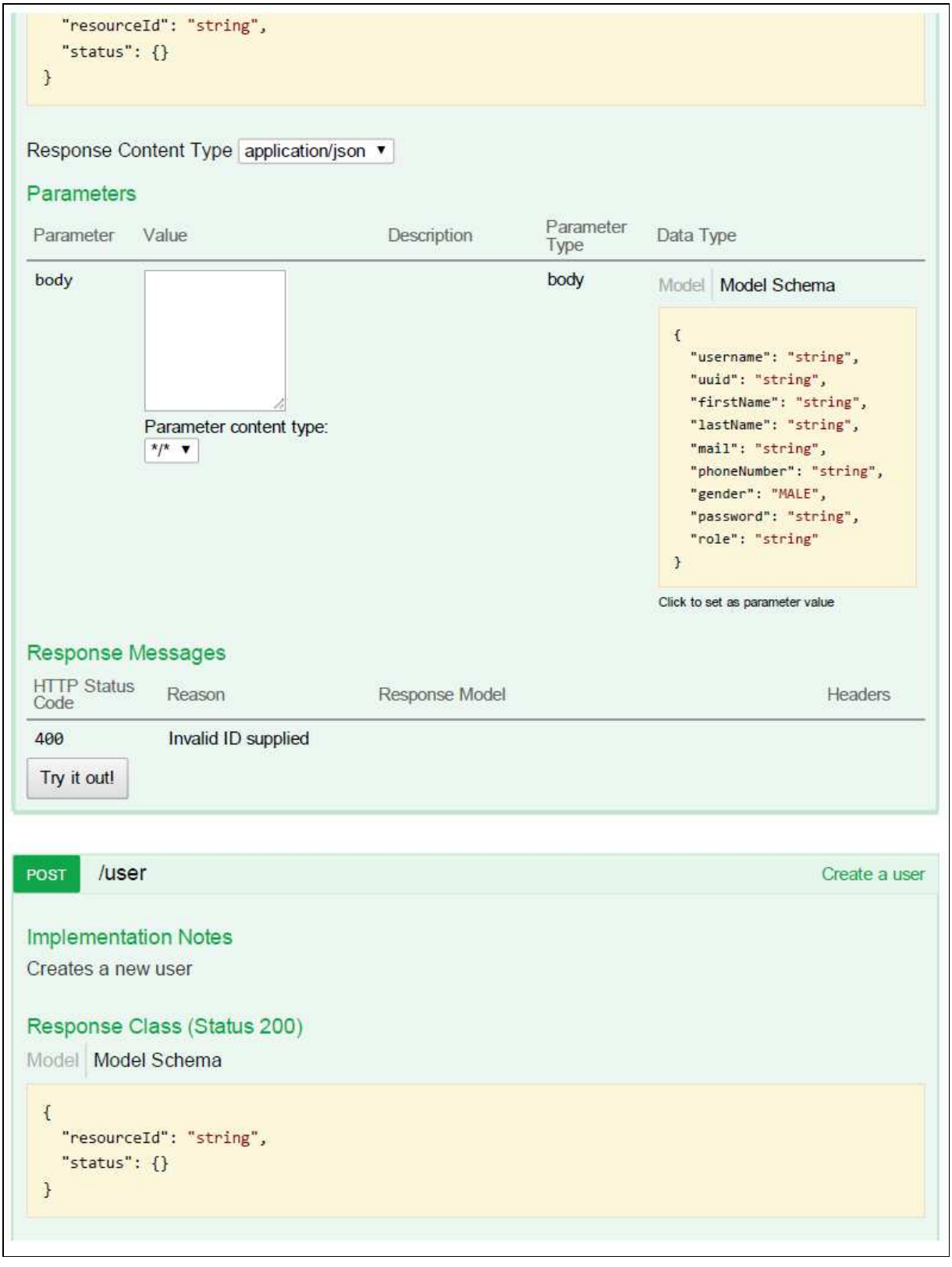

Fahrt anbieten
+Fahrtdaten eingen: entry
+Mitfahranfrage bestätigen: entry
+Mitfahranfrag ablehnen: entry
+Fahrt absagen: exit
+()
Fahrtdaten eigeben
Fahrt erstellt
Fahrt gebucht
freie Plätze
Mitfahranfrage bestätigt
Mitfahranfrag bestätigen
Fahrt ausgebucht
Fahrt durchgeführt
Mitfahranfrage ablehnen
Anzahl freie Plätze +1
when
(Anzahl Mitfahrer =
max. Anzahl Mitfahrer)
after
(Datum für
Mitfahranfrage vorrüber)
Fahrt absagen
Mitfahrer sagt ab
[Platz wird frei]
Fahrer benachrichtigt
Mitfahrer benachrichtigt

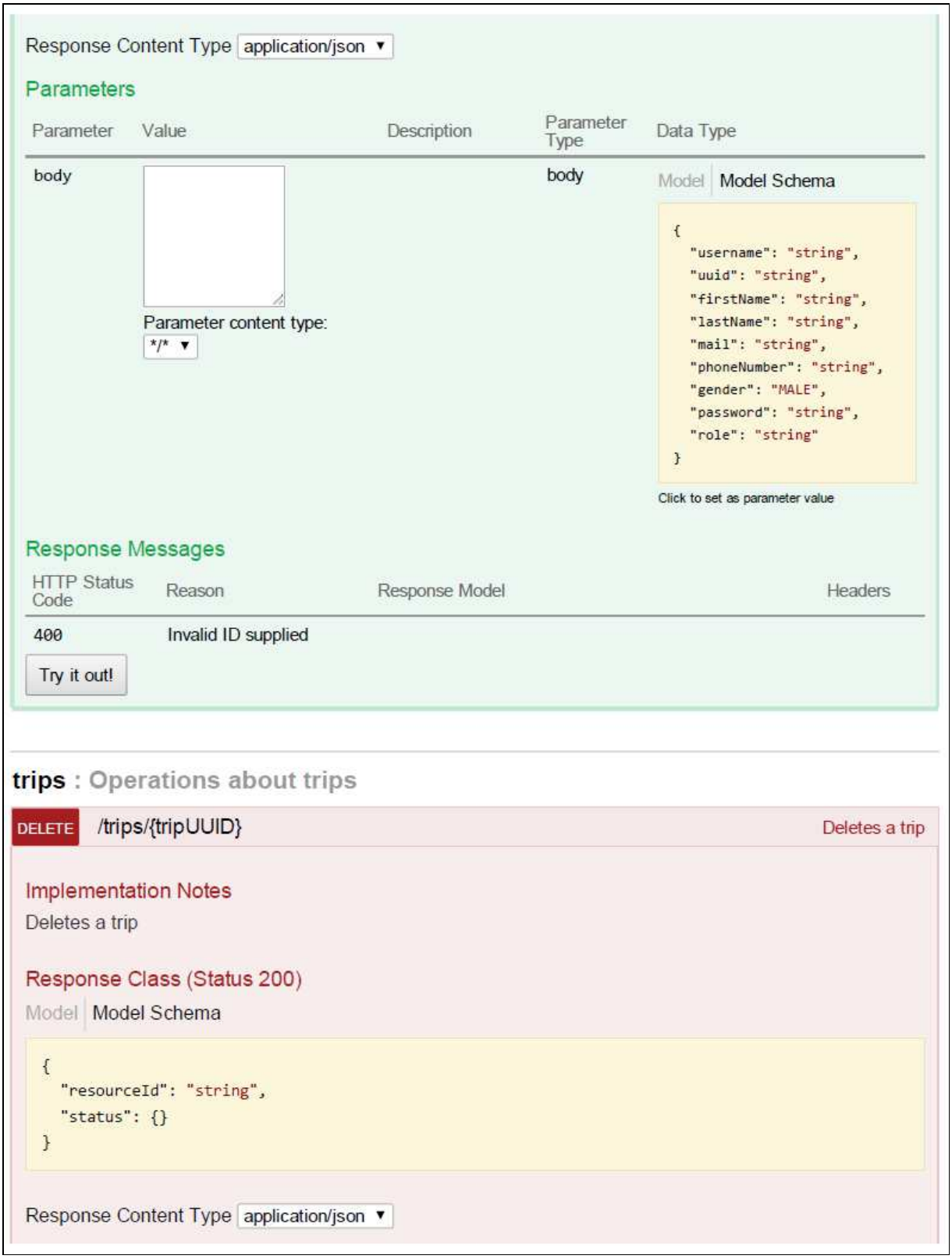
X


Fahrt suchen
+Fahrtdaten eingen: entry
+Mitfahranfrage senden: entry
+Fahrt absagen: exit
+ ()
Fahrtdaten eigeben
Fahrt gefunden
freie Plätze?
Fahrt gebucht
Mitfahranfrage senden
Fahrtanfrag bestätigt
Fahrt ausgebucht
Fahrt durchgeführt
Anzahl freie Plätze +1
Absage von Fahrer
when
(Anzahl Mitfahrer >
max. Anzahl Mitfahrer)
after
(Datum für
Mitfahranfrage vorrüber)
Fahrt absagen
Fahrer sagt Fahrt ab
Fahrt gefunden
Fahrtdaten eingeben
Fahrtanfrage nicht bestätigt
[Platz wird frei]
Fahrer benachrichtigt
Mitfahrer benachrichtigt

MiZe – Swagger-UI:




"resourceId": "string",
"status": {}
}
Response Content Type application/json
Parameters
Parameter Value Description Parameter Type Data Type
body body Model Model Schema
{
"username": "string",
"uuid": "string",
"firstName": "string",
"lastName": "string",
"mail": "string",
"phoneNumber": "string",
"gender": "MALE",
"password": "string",
"role": "string"
}
Parameter content type:
/
Click to set as parameter value
Response Messages
HTTP Status Code Reason Response Model Headers
400 Invalid ID supplied
Try it out!
POST /user Create a user
Implementation Notes
Creates a new user
Response Class (Status 200)
Model Model Schema
{
"resourceId": "string",
"status": {}
}



Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	Parameter content type: */*		body	Model Schema

```
{
  "username": "string",
  "uuid": "string",
  "firstName": "string",
  "lastName": "string",
  "mail": "string",
  "phoneNumber": "string",
  "gender": "MALE",
  "password": "string",
  "role": "string"
}
```

Click to set as parameter value

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		

Try it out!

trips : Operations about trips

DELETE /trips/{tripUUID} Deletes a trip

Implementation Notes

Deletes a trip

Response Class (Status 200)

Model Model Schema

```
{
  "resourceId": "string",
  "status": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
tripUUID	(required)	**UUID of trip that needs to be deleted**	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Trip not found		

Try it out!

GET /trips/{tripUUID} Find trip by UUID

Implementation Notes

Returns a trip

Response Class (Status 200)

Model | Model Schema

```
{
  "trips": {},
  "status": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
tripUUID	(required)	**UUID of trip that needs to be fetched**	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
404	Invalid ID supplied		
500	Other Error		

Try it out!

PUT /trips/{tripUUID} Update a trip

Implementation Notes

Updates a trip

Response Class (Status 200)

Model | Model Schema

```
{
  "resourceId": "string",
  "status": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	Parameter content type: application/json		body	Model | Model Schema { "startTime": "string", "description": "string", "uuid": "string", "from": {}, "to": {}, "freeSeats": 0, "price": 0, "active": true, "participants": {}, "createTime": "string", "updateTime": "string" } Click to set as parameter value
tripUUID	(required)	**UUID of trip that needs to be updated**	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		

404	Trip not found

Try it out!

GET /trips — Get all trips

Implementation Notes

Returns a list of all trips in a given radius from a geo coordination

Response Class (Status 200)

Model | Model Schema

```
{
  "trips": {},
  "status": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
latitude		Latitude of the Geo Coordinate	query	double
longitude		Longitude of the Geo Coordinate	query	double
radius		The radius with the geo coordinates as center	query	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Trip not found		

Try it out!

POST /trips Create a trip

Implementation Notes

Creates a new trip based on the JSON

Response Class (Status 200)

Model | Model Schema

```
{
  "resourceId": "string",
  "status": {}
}
```

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	Parameter content type: application/json ▾		body	Model | Model Schema

```
{
  "startTime": "string",
  "description": "string",
  "uuid": "string",
  "from": {},
  "to": {},
  "freeSeats": 0,
  "price": 0,
  "active": true,
  "participants": {},
  "createTime": "string",
  "updateTime": "string"
}
```

Click to set as parameter value

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Trip not found		

Try it out!

DELETE /trips/{userUUId}/{tripUUID} Unbook a trip

Implementation Notes

Unbooks a trip for the logged user

Response Class (Status 200)

Model | Model Schema

```
{
  "resourceId": "string",
  "status": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
userUUId	(required)	**UUID of the user**	path	string
tripUUID	(required)	**UUID of the trip**	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Trip not found		

Try it out!

POST /trips/{userUUId}/{tripUUID} Book a trip

Implementation Notes

Books a trip for the logged user

Response Class (Status 200)

Model | Model Schema

```
{
  "resourceId": "string",
  "status": {}
```

```
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
userUUId	(required)	**UUID of the user**	path	string
tripUUID	(required)	**UUID of the trip**	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Trip not found		

Try it out!

vehicles : Operations about vehicle

DELETE /vehicles/{vehicleUUID} Deletes a vehicle

Implementation Notes

Deletes a vehicle by UUID

Response Class (Status 200)

Model | Model Schema

```
{
  "resourceId": "string",
  "status": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
vehicleUUID	(required)	**UUID of vehicle that needs to be deleted**	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Trip not found		

Try it out!

PUT /vehicles/{vehicleUUID} Update a vehicle

Implementation Notes

Updates a vehicle

Response Class (Status 200)

Model | Model Schema

```
{
  "resourceId": "string",
  "status": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	Parameter content type: */*		body	Model | Model Schema { "model": "string", "make": "string", "seats": 0, "userId": "string", "vehicleId": "string" } Click to set as parameter value
vehicleUUID	(required)	**UUID of vehicle that needs to be updated**	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Trip not found		

Try it out!

POST /vehicles Create a vehicle

Implementation Notes

Creates a new vehicle based on the JSON

Response Class (Status 200)

Model | Model Schema

```
{
  "resourceId": "string",
  "status": {}
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	Parameter content type: application/json		body	Model | Model Schema

```
{
  "model": "string",
  "make": "string",
  "seats": 0,
  "userId": "string",
  "vehicleId": "string"
}
```

Click to set as parameter value

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	Invalid ID supplied		
404	Trip not found		

Try it out!

GET /vehicles/id/{vehicleUUID} Find a vehicle by UUID

Implementation Notes

Returns a vehicle by UUID if the user is allowed to retrieve the information

Response Class (Status 200)

Model | Model Schema

```
{
  "vehicles": {},
  "status": {}
}
```

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
vehicleUUID	(required)	**ID of the vehicle that needs to be fetched**	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
400	INVALID.UUID		
404	TRIP.NOT.FOUND		
500	Other Error		

Try it out!

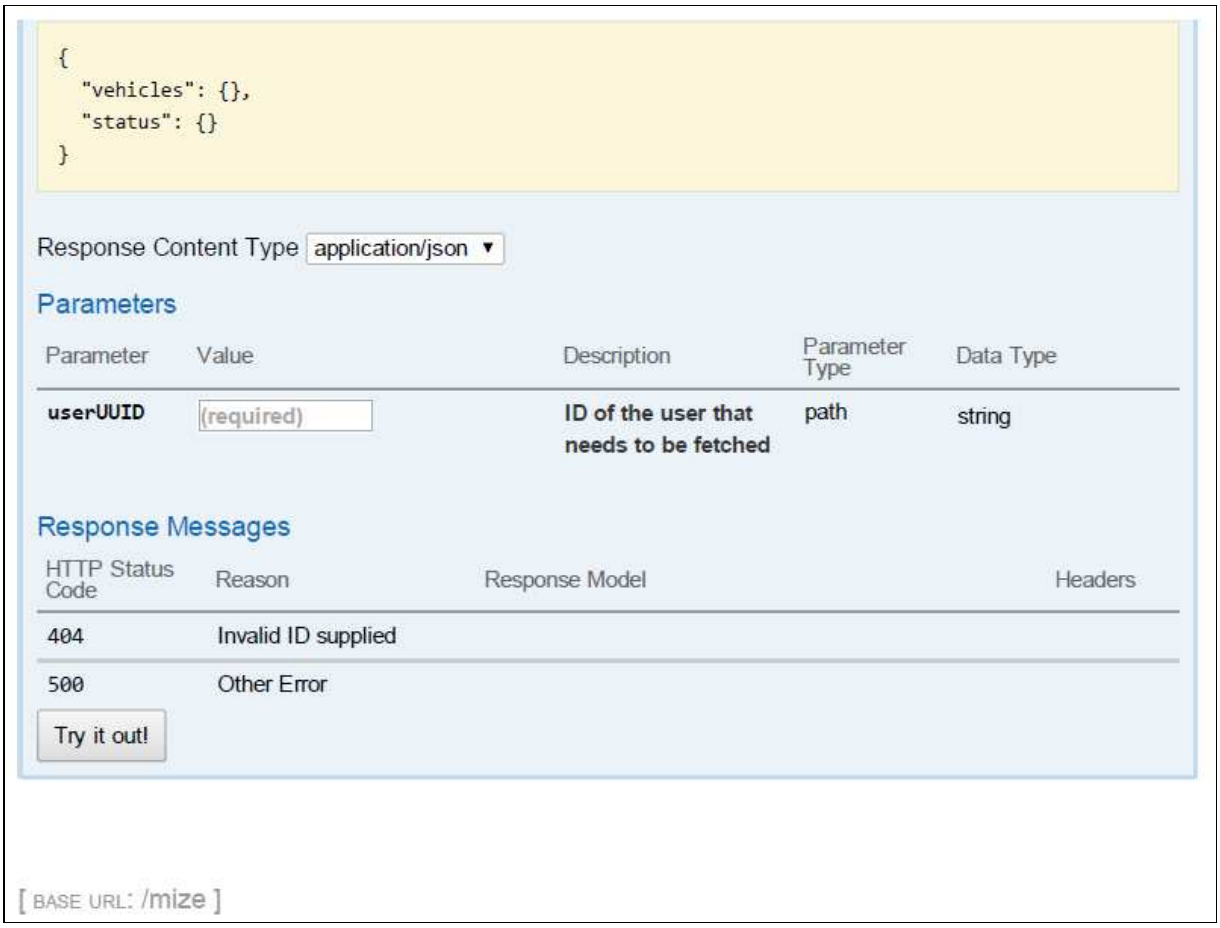
GET /vehicles/user/{userUUID} Find vehicles by UUID of an user

Implementation Notes

Returns all vehicles by user UUID if the user is allowed to retrieve the information

Response Class (Status 200)

Model | Model Schema



MiZe – SoapUI Testsuite:

```
<?xml version="1.0" encoding="UTF-8"?>

<con:testSuite id="a8941494-9d9f-436e-a882-05e16caf83e8"
name="MiZeTestsuite"
xmlns:con="http://eviware.com/soapui/config"><con:settings/><con:run
Type>SEQUENTIAL</con:runType><con:testCase id="9467fc5c-0b9e-454e-
8615-a14f6bff7e44" failOnError="true" failTestCaseOnErrors="true"
keepSession="false" maxResults="0" name="CRUDTestCase"
searchProperties="true"><con:settings/><con:testStep
type="restrequest" name="PostTrip" id="37293b30-8c52-49cc-8f55-
87c6951ac0be"><con:settings/><con:config service="application"
resourcePath="/mize//trips" methodName="POST - createTrip"
xsi:type="con:RestRequestStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><con:restRequest name="PostTrip" id="18433af9-c4fa-4605-
bb02-c9afa04b88b7" mediaType="application/json"
postQueryString="false"><con:settings><con:setting
id="com.eviware.soapui.impl.wsdl.WsdlRequest@request-
headers">&lt;entry key="x-uuid" value="2081f0df-c5cd-4a4d-82ea-
c2ffb0c5c76c"
xmlns="http://eviware.com/soapui/config"/></con:setting></con:settin
```

```
gs><con:endpoint>http://mize.msm-
projects.net:1337</con:endpoint><con:request>{

  "from": {"street": "Musterstraße", "streetNumber": "12",
"zipCode": "00123", "city": "Musterstadt", "country": "GERMANY",
"geoCoordinate": {"latitude": 49.872693, "longitude": 8.629604}},

  "to": {"street":"Musterstraße", "streetNumber": "22", "zipCode":
"00123", "city": "Musterstadt", "country": "GERMANY",
"geoCoordinate": {"latitude": 51.312679, "longitude": 12.374931}},

  "freeSeats": 0,

  "price": 0,

  "active": true,

  "startTime": "2015-07-20T14:34:00+02:00",

  "description": "Test"

}</con:request><con:originalUri>http://mize.msm-
projects.net/mize//trips</con:originalUri><con:assertion type="Valid
HTTP Status Codes" id="496ff2c1-8118-4cfb-85da-6df431c6e48f"
name="Valid HTTP Status
Codes"><con:configuration><codes>201</codes></con:configuration></co
n:assertion><con:assertion type="Simple Contains" id="59bd1be2-a412-
4dab-8408-f6b30d0cf309"
name="Contains"><con:configuration><token>"resourceId"</token><ignor
eCase>false</ignoreCase><useRegEx>false</useRegEx></con:configuratio
n></con:assertion><con:credentials><con:authType>No
Authorization</con:authType></con:credentials><con:jmsConfig
JMSDeliveryMode="PERSISTENT"/><con:jmsPropertyConfig/><con:parameter
s/></con:restRequest></con:config></con:testStep><con:testStep
type="transfer" name="TripUUIDTransfer" id="016e1198-2a2a-443a-86a0-
91ccb45aff18"><con:settings/><con:config
xsi:type="con:PropertyTransfersStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><con:transfers
setNullOnMissingSource="true" transferTextContent="true"
failOnError="false" ignoreEmpty="false" transferToAll="true"
entitize="false"
transferChildNodes="false"><con:name>resourceID</con:name><con:sourc
eType>Response</con:sourceType><con:sourceStep>PostTrip</con:sourceS
tep><con:sourcePath>resourceId</con:sourcePath><con:targetType>tripU
UID</con:targetType><con:targetStep>DeleteTrip</con:targetStep><con:
type>JSONPATH</con:type><con:targetTransferType>JSONPATH</con:target
TransferType><con:upgraded>true</con:upgraded></con:transfers><con:t
ransfers setNullOnMissingSource="true" transferTextContent="true"
failOnError="false" transferToAll="false" ignoreEmpty="false"
entitize="false"
transferChildNodes="false"><con:name>ResourceIdToGetTrip</con:name><
con:sourceType>Response</con:sourceType><con:sourceStep>PostTrip</co
```

```
n:sourceStep><con:sourcePath>resourceId</con:sourcePath><con:targetT
ype>tripUUID</con:targetType><con:targetStep>GetTrip</con:targetStep
><con:type>JSONPATH</con:type><con:targetTransferType>JSONPATH</con:
targetTransferType><con:upgraded>true</con:upgraded></con:transfers>
<con:transfers setNullOnMissingSource="true"
transferTextContent="true" failOnError="false" ignoreEmpty="false"
transferToAll="false" entitize="false"
transferChildNodes="false"><con:name>ResourceIdToUpdateTrip</con:nam
e><con:sourceType>Response</con:sourceType><con:sourceStep>PostTrip<
/con:sourceStep><con:sourcePath>resourceId</con:sourcePath><con:targ
etType>tripUUID</con:targetType><con:targetStep>UpdateTrip</con:targ
etStep><con:type>JSONPATH</con:type><con:targetTransferType>JSONPATH
</con:targetTransferType><con:upgraded>true</con:upgraded></con:tran
sfers></con:config></con:testStep><con:testStep type="restrequest"
name="GetTrip" id="6b2b3818-b29e-4c61-b2fb-
47575889118e"><con:settings/><con:config service="application"
resourcePath="/mize//trips/{tripUUID}" methodName="GET -
getTripByUUID" xsi:type="con:RestRequestStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><con:restRequest name="GetTrip" id="2c321551-7e48-4007-
8f0f-c470918ed632"
mediaType="application/json"><con:settings><con:setting
id="com.eviware.soapui.impl.wsdl.WsdlRequest@request-
headers">&lt;xml-
fragment/></con:setting></con:settings><con:endpoint>http://mize.msm
-
projects.net:1337</con:endpoint><con:request/><con:originalUri>http:
//mize.msm-
projects.net/mize//trips/123</con:originalUri><con:assertion
type="Valid HTTP Status Codes" id="dd3c407f-ce27-46cf-9501-
0a8578832ae4" name="Valid HTTP Status
Codes"><con:configuration><codes>200</codes></con:configuration></co
n:assertion><con:credentials><con:selectedAuthProfile>Basic</con:sel
ectedAuthProfile><con:addedBasicAuthenticationTypes>Basic</con:added
BasicAuthenticationTypes><con:authType>Global HTTP
Settings</con:authType></con:credentials><con:jmsConfig
JMSDeliveryMode="PERSISTENT"/><con:jmsPropertyConfig/><con:parameter
s><entry key="tripUUID" value="85471c30-bda1-4317-97d4-e2882dd97578"
xmlns="http://eviware.com/soapui/config"/></con:parameters></con:res
tRequest></con:config></con:testStep><con:testStep
type="restrequest" name="UpdateTrip" id="ae867fab-a609-4117-a320-
661b2a28297b"><con:settings/><con:config service="application"
resourcePath="/mize//trips/{tripUUID}" methodName="PUT - updateTrip"
xsi:type="con:RestRequestStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><con:restRequest name="UpdateTrip" id="5b09f3fe-76b8-44c6-
8732-4d9831876a36" mediaType="application/json"
postQueryString="false"><con:settings><con:setting
id="com.eviware.soapui.impl.wsdl.WsdlRequest@request-
```

*headers"><xml-
fragment/></con:setting></con:settings><con:endpoint>http://mize.msm
-
projects.net:1337</con:endpoint><con:request/><con:originalUri>http:
//mize.msm-projects.net/mize//trips/0ae768e2-51bd-482c-8bc5-
0d02f17f2c71</con:originalUri><con:assertion type="Valid HTTP Status
Codes" id="f120892f-d0bd-4d4f-bbae-09ed9842d083" name="Valid HTTP
Status
Codes"><con:configuration><codes>200</codes></con:configuration></co
n:assertion><con:credentials><con:authType>No
Authorization</con:authType></con:credentials><con:jmsConfig
JMSDeliveryMode="PERSISTENT"/><con:jmsPropertyConfig/><con:parameter
s><entry key="tripUUID" value="85471c30-bda1-4317-97d4-e2882dd97578"
xmlns="http://eviware.com/soapui/config"/></con:parameters></con:res
tRequest></con:config></con:testStep><con:testStep
type="restrequest" name="DeleteTrip" id="416ae6b1-3854-44ed-b728-
c5a9688beb87"><con:settings/><con:config service="application"
resourcePath="/mize//trips/{tripUUID}" methodName="DELETE -
deleteTrip" xsi:type="con:RestRequestStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><con:restRequest name="DeleteTrip" id="31dfb3df-093c-45ec-
848f-0efa7c1e9e19" mediaType="application/json"
postQueryString="false"><con:settings><con:setting
id="com.eviware.soapui.impl.wsdl.WsdlRequest@request-
headers"><xml-
fragment/></con:setting></con:settings><con:endpoint>http://mize.msm
-
projects.net:1337</con:endpoint><con:request/><con:originalUri>http:
//mize.msm-projects.net/mize//trips/</con:originalUri><con:assertion
type="Valid HTTP Status Codes" id="ec2dd9e1-b5cf-46f3-81e3-
138b2b107389" name="Valid HTTP Status
Codes"><con:configuration><codes>200</codes></con:configuration></co
n:assertion><con:assertion type="Simple Contains" id="022f69c4-d02f-
40c4-9410-d36b9b61cbd1"
name="Contains"><con:configuration><token>"resourceId"</token><ignor
eCase>false</ignoreCase><useRegEx>false</useRegEx></con:configuratio
n></con:assertion><con:credentials><con:authType>No
Authorization</con:authType></con:credentials><con:jmsConfig
JMSDeliveryMode="PERSISTENT"/><con:jmsPropertyConfig/><con:parameter
s><entry key="tripUUID" value="85471c30-bda1-4317-97d4-e2882dd97578"
xmlns="http://eviware.com/soapui/config"/></con:parameters></con:res
tRequest></con:config></con:testStep><con:properties/></con:testCase
><con:testCase id="be442feb-f848-469c-afc3-afa7598ccd9e"
failOnError="true" failTestCaseOnErrors="true" keepSession="false"
maxResults="0" name="UserTestCase"
searchProperties="true"><con:settings/><con:testStep
type="restrequest" name="CreateUser" id="e1d478df-4763-4bc1-aa88-
a7e0f6239b0e"><con:settings/><con:config service="application"
resourcePath="/mize//user" methodName="POST - createUser"*

```
xsi:type="con:RestRequestStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><con:restRequest name="CreateUser" id="e65ad43e-b468-4cb1-
b27f-b42b49388d3e" mediaType="application/json"
postQueryString="false"><con:settings><con:setting
id="com.eviware.soapui.impl.wsdl.WsdlRequest@request-
headers">&lt;xml-
fragment/></con:setting></con:settings><con:endpoint>http://mize.msm
-projects.net:1337</con:endpoint><con:request>{

  "gender": "MALE",

  "lastName": "User",

  "firstName": "Test",

  "mail": "Testy@test.com",

  "phoneNumber": "1234",

  "password": "Test1234",

  "uuid": "string",

  "username": "MiZeTestUser"

}</con:request><con:originalUri>http://mize.msm-
projects.net/mize//user</con:originalUri><con:assertion type="Valid
HTTP Status Codes" id="ca8a5bf5-f2d9-4db8-9cb7-386594083c8d"
name="Valid HTTP Status
Codes"><con:configuration><codes>200</codes></con:configuration></co
n:assertion><con:assertion type="Simple Contains" id="521cb615-2874-
4db3-8d8d-8bd8f6553f3e"
name="Contains"><con:configuration><token>"uuid"</token><ignoreCase>
false</ignoreCase><useRegEx>false</useRegEx></con:configuration></co
n:assertion><con:credentials><con:authType>No
Authorization</con:authType></con:credentials><con:jmsConfig
JMSDeliveryMode="PERSISTENT"/><con:jmsPropertyConfig/><con:parameter
s/></con:restRequest></con:config></con:testStep><con:testStep
type="transfer" name="UserUUIDTransfer" id="b3d2b5e8-09a4-447f-803f-
19d0f27b04e6"><con:settings/><con:config
xsi:type="con:PropertyTransfersStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><con:transfers
setNullOnMissingSource="true" transferTextContent="true"
failOnError="true" ignoreEmpty="false" transferToAll="false"
entitize="false"
transferChildNodes="false"><con:name>UUIDToGetUser</con:name><con:so
urceType>Response</con:sourceType><con:sourceStep>CreateUser</con:so
urceStep><con:sourcePath>uuid</con:sourcePath><con:targetType>userUU
ID</con:targetType><con:targetStep>getuserByUUID</con:targetStep><co
n:type>JSONPATH</con:type><con:targetTransferType>JSONPATH</con:targ
etTransferType><con:upgraded>true</con:upgraded></con:transfers></co
```

```
n:config></con:testStep><con:testStep type="restrequest"
name="getuserByUUID" id="f4e114f4-6a36-4a1e-93d6-
0b179b748c49"><con:settings/><con:config service="application"
resourcePath="/mize//user/{userUUID}" methodName="GET -
getUserByUUID" xsi:type="con:RestRequestStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><con:restRequest name="getuserByUUID" id="2a1d2929-a675-
4e6b-b776-9387391ecba8"
mediaType="application/json"><con:settings><con:setting
id="com.eviware.soapui.impl.wsdl.WsdlRequest@request-
headers">&lt;xml-
fragment/></con:setting></con:settings><con:endpoint>http://mize.msm
-
projects.net:1337</con:endpoint><con:request/><con:originalUri>http:
//mize.msm-projects.net/mize//user/515602f1-8cb8-4f69-b2de-
67b326882044</con:originalUri><con:assertion type="Valid HTTP Status
Codes" id="4c175d20-ad22-4594-ac01-f486c19e9c01" name="Valid HTTP
Status
Codes"><con:configuration><codes>200</codes></con:configuration></co
n:assertion><con:assertion type="Simple Contains" id="41516a81-e956-
408b-b6f7-71f20705e31f"
name="Contains"><con:configuration><token>"uuid"</token><ignoreCase>
false</ignoreCase><useRegEx>false</useRegEx></con:configuration></co
n:assertion><con:credentials><con:authType>No
Authorization</con:authType></con:credentials><con:jmsConfig
JMSDeliveryMode="PERSISTENT"/><con:jmsPropertyConfig/><con:parameter
s><entry key="userUUID" value="6f250d95-44ba-4e08-a595-d96984d9f527"
xmlns="http://eviware.com/soapui/config"/></con:parameters></con:res
tRequest></con:config></con:testStep><con:testStep
type="restrequest" name="LogIn" id="b17b81fd-c64d-4858-a577-
085de3ebaf81"><con:settings/><con:config service="application"
resourcePath="/mize//user/login" methodName="POST - loginUser"
xsi:type="con:RestRequestStep"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><con:restRequest name="LogIn" id="ddbb399f-9f92-4f06-b6af-
8099a1fdce9f" mediaType="application/json"
postQueryString="false"><con:settings><con:setting
id="com.eviware.soapui.impl.wsdl.WsdlRequest@request-
headers">&lt;xml-
fragment/></con:setting></con:settings><con:endpoint>http://mize.msm
-projects.net:1337</con:endpoint><con:request>{

  "password": "Test1234",

  "username": "MiZeTestUser"

}</con:request><con:originalUri>http://mize.msm-
projects.net/mize//user/login</con:originalUri><con:assertion
type="Valid HTTP Status Codes" id="c36304b7-3f63-4e24-9176-
ffce252b03e3" name="Valid HTTP Status
```

```
Codes"><con:configuration><codes>200</codes></con:configuration></co
n:assertion><con:credentials><con:authType>No
Authorization</con:authType></con:credentials><con:jmsConfig
JMSDeliveryMode="PERSISTENT"/><con:jmsPropertyConfig/><con:parameter
s/></con:restRequest></con:config></con:testStep><con:properties/></
con:testCase><con:properties/><con:reportParameters/></con:testSuite
>
```