Westsächsische Hochschule Zwickau

Fakultät Physikalische Technik/Informatik

Studiengang: Informatik

Bachelorarbeit

Testfallgenerator für die Automatisierungstest von 4G/5G-Funkzugangsnetz

Shiyi Ying

Matrikel: 36449

Mentor:

Prof. Dr.-Ing. Thomas Franke

Herr Zhiwei Chen

22. 07. 2022

Thema der Bachelorarbeit

Testfallgenerator für die Automatisierungstest von 4G/5G-Funkzugangsnetz

Autorenreferat

Innerhalb des Basisstationsherstellers entwickelt das Softwareteam automatisierte Testfälle und zugehörige Softwarebibliotheken für Testingenieure Mit der Popularität von 5G hat die Komplexität der Funktionen und Szenarien von Basisstationen zugenommen, was zu einer hohen Nachfrage nach Testautomatisierung für Basisstationen geführt hat. In dieser Arbeit wurde die Möglichkeit untersucht, ein System zu entwickeln, das die Erstellung und Ausführung von Testfällen erheblich vereinfacht, um die Arbeitsbelastung von Testingenieuren und Entwicklern zu verringern. Bei der Konzeption dieser Arbeit wurden Anforderungsanalyse und Datenmodellierung eingesetzt. Die System-implementierung ist eine Webanwendung, die auf Python und dem Django-Framework basiert und mit der sich Robot Framework-Testfälle generieren und ausführen lassen. Die implementierte Anwendung demonstriert die Machbarkeit des in dieser Arbeit beschriebenen Systems.

Bachelorarbeit

Studiengang Informatik, Fakultät Physikalische Technik/Informatik

Westsächsische Hochschule Zwickau

Betreuer/Erstgutachter: Prof. Dr.-Ing. Thomas Franke

Zweitgutachter: Prof. Dr.-Ing. Rainer Wasinger

Betreuer in Unternehmen: Herr Zhiwei Chen (Nokia Network Solution Technology)

Abgegeben am 22.07.2022

Inhaltsverzeichnis

Abkürzungsverzeichnis 3

1 Einführung 6

1.1 Automatisierungstest von 4G/5G-Funkzugangsnetz 7

1.2 Zielsetzung 8

1.3 Gliederung der Arbeit 9

2 Grundlagen 10

2.1 Umgebung zur RAN-Testautomatisierung 10

2.1.1 gNB/eNB 11

2.1.2 UE 12

2.1.3 Programmierbare Dämpfungsglied (PA) 14

2.2 Robot-Framework 15

2.2.1 Struktur und Syntax 15

2.2.2 Ausführung und Logs 17

2.3 Django-Framework 17

3 Analyse 18

3.1 Szenarien 18

3.1.1 Testfällen verwalten und bearbeiten 18

3.1.2 Testumgebung verwalten 19

3.1.3 Test Set, Testdurchführung und Ergebnisanzeige 20

3.1.4 Testschritte und Testumgebungselemente aktualisieren 21

3.1.5 Überblick 21

3.2 Anforderungsanalyse 22

3.2.1 Funktionale Anforderungen 23

3.2.2 Nicht-funktionale Anforderungen 31

3.3 Zusammenfassung 32

4 Entwurf und Architektur 32

4.1 System Überblick 32

4.2 Datenarchitektur 33

4.3 Model-View-Controller und Django MTV 35
4.3.1 Model 36
4.3.2 View / Django Template 40
4.3.3 Controller / Django Views 41
5 Realisierung 42
5.1 Implementierung 42
5.1.1 Modellklassen 44
5.1.2 Realisierung von Szenarien 44
5.2 Evaluation 52
5.2.1 Abdeckung der Anforderungen 52
6 Zusammenfassung 55
6.1 Auswertung der Ergebnisse 55
6.2 Ausblick 56
7 Literaturverzeichnis 57

Abkürzungsverzeichnis

3GPP ..... 3rd Generation Partnership Project

4G ..... vierte Generation des Mobilfunks

5G ..... fünfte Generation des Mobilfunks

AAU ..... Active Antenna Unit, Antennenmodule für 5G Basisstationen

BBU ..... Baseband Unit für Basisstationen

eNB ..... eNodeB, 4G Basisstationen

ERM ..... Entity-Relationship Model

gNB ..... eNodeB, 5G Basisstationen

IMEI ..... International Mobile Equipment Identity

IMSI ..... International Mobile Subscriber Identity

NR ..... 5G New Radio

PA ..... Programmierbare Dämpfungsglied

RAN ..... Radio Access Network, Funkzugangsnetz

REST ..... Representational state transfer

RRU ..... Radio Remote Unit, Antennenmodule für 4G Basisstationen

SUT ..... System Under Test

TA ..... Testautomatisierung

UE ..... User Equipment, Teilnehmergerät

1 Einführung

Mobilfunknetze sind heute eine der Infrastrukturen weltweit, und das Funkzugangsnetz (RAN) ist einer der wichtigsten Teile davon. Das Funkzugangsnetz besteht aus einer Reihe von Hardware- und Softwaregeräten, hauptsächlich Basisstationen, die die Endgeräte (z. B. Mobiltelefone) mit dem Netz verbinden. [Jon21] Daher benötigen Funkzugangsnetze stabile, sichere, kostengünstige und leistungsstarke Basisstationen, um den Betrieb der Infrastruktur zu sichern.

Die Hauptthersteller von Basisstationen sind Huawei, Ericsson, Nokia, ZTE, Samsung, usw. Regionale Mobilfunknetzbetreiber (z. B. Telekom, Vodafone, CMCC, AT&T usw.) sind Partnerschaften mit diesen Herstellern eingegangen und nutzen deren Geräte zum Aufbau von Mobilfunknetzen. Diese Netze reichen von öffentlichen Mobilfunknetzen, die die Menschen im täglichen Leben nutzen, bis hin zu speziellen Funknetzen, die in Branchen wie Eisenbahn, Häfen und Logistik eine Rolle spielen.

Die Hersteller von Basisstationsausrüstungen liefern ihren Kunden nicht nur die Hardware für Basisstationen, sondern auch ihre Software und eine Reihe von zusätzlicher Anwendungen. Bevor die neuen Hardware- und Softwareversionen der RAN-Geräte für die Kunden freigegeben werden, müssen die Hersteller sie testen. Diese Tests sollen sicherstellen, dass die Funktionalität, Leistung, Stabilität und Kompatibilität der RAN-Geräte den Erwartungen entsprechen. 5G-Netze umfassen Funktionen wie Mobile Edge Computing, Network Slicing und selbstregulierendes Netzmanagement. Die Softwaresysteme für RAN-Geräte werden daher immer komplexer [5GP16]. Bei der Tests des RAN hat daher die Softwaretests nicht weniger oder sogar mehr Gewicht als die Hardwaretests.

Die vorgenannten Tests an RAN-Geräten werden als Abnahmetests bezeichnet. Während der Abnahmetests verwenden die Testingenieure Basisstationen, UE (Teilnehmergerät) und zugehörige Ausrüstung, um reale Szenarien zu simulieren. Während des Tests sammeln die Testingenieure Daten und Logs zur Inspektion und Analyse. Je nach Zweck können die Tests im Labor oder im Feld durchgeführt werden. Durch die Durchführung von Tests identifizieren die Testingenieure mögliche Hardware- und Softwareprobleme der Basisstation und übermitteln diese an die Forschungs- und Entwicklungsabteilung.

1.1 Automatisierungstest von 4G/5G-Funkzugangsnetz

Zu den üblichen Aufgaben bei RAN-Tests gehören unter anderem die folgenden:

- Konfigurieren und Verwalten von Basisstationen.
- Programmierbare Dämpfungsglied (PA) einstellen
- Betreiben des UE(s) zur Simulation des Benutzerverhaltens
- Log-Erfassung für Basisstationen und UEs

Die manuelle Durchführung dieser Aufgaben bedeutet, dass die Testingenieure mehr Zeit und Aufwand benötigen und dass die Genauigkeit der Tests nicht vollständig garantiert werden kann. Die 5G-NR-Releases des 3GPP haben eine große Anzahl von Funktionen und unterschiedliche Netztopologie mitgebracht. Infolgedessen ist die Tests von RAN-Geräten vielfältiger und komplexer werden. Die Effizienz, die Genauigkeit, die Stabilität und die Kosten der manuellen Tests sind für diese Anforderungen nicht mehr ausreichend. Deshalb ist das automatisierte Testen von 5G-RAN-Geräten zum aktuellen Industrietrend geworden. Zusätzlich unterscheiden sich Abnahmekriterien, Verfahren und Informations-formate von Kunde zu Kunde, was die Komplexität der Testfälle weiter erhöht. [Che20]

Für die Bereitstellung automatisierter Testfälle durch die Entwickler des Testautomatisierungsteams (TA) gibt es zwei Hauptansätze. Wie in Abbildung 1.1 dargestellt.

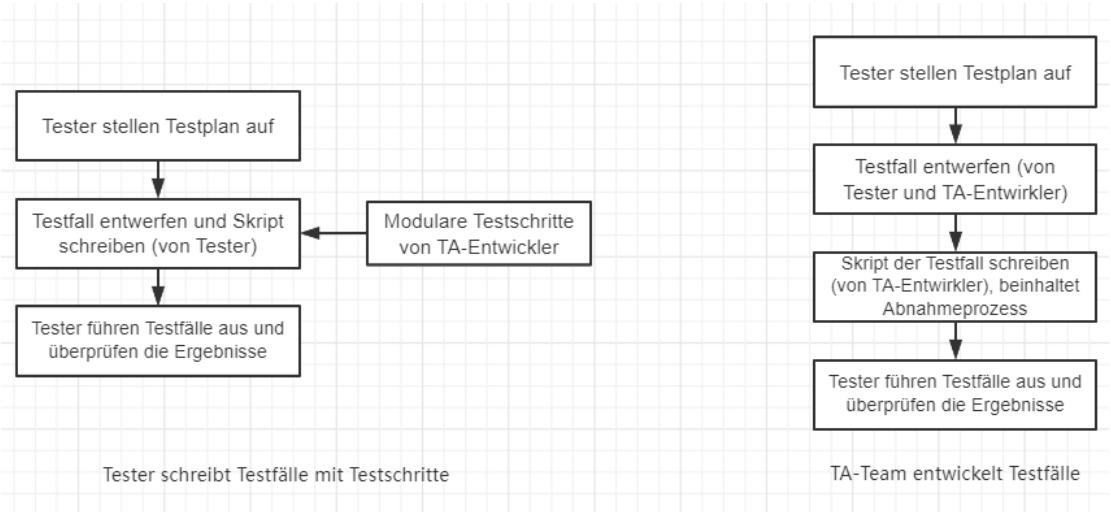

Abbildung 1.1: Hauptansätze der Durchführung von RAN-Automatisierungstests

1. Der erste Ansatz ist, dass das TA-Team modulare Testschritte entwickelt. Auf der Grundlage dieser Testschritte stellen die Testingenieure verschiedene Testfälle in der Sprache des Testframeworks (Python oder Robot Framework) zusammen.

Die Vorteile dieses Ansatzes sind:

- Die Entwickler des TA-Teams müssen nicht das komplette Geschäft kennen.
- Der Entwickler muss sich nur um die einzelnen Module befassen, die den Testschritten entsprechen (wie z.B Schnittstellen, Protokolle...)
- Hohe Wiederverwendbarkeit
- Keine Anforderungsanalyse für TA-Team, niedriger Liefer- und Wartungsaufwand

Die Nachteile sind:

- Die Testingenieure im Geschäftsteam müssen die Syntax und Logik des Testrahmens kennenlernen.
    - sogar auch die entsprechenden Bibliotheken und Schnittstellen

2. Das TA-Team entwickelt geeignete Testfälle gemäß den Anforderungen der Testingenieure.

Die Vorteile dieses Ansatzes sind:

- Im Idealfall können die Testingenieure die Testfälle sofort nach der Lieferung verwenden.
- Testingenieure müssen sich nur auf ihr eigenes Fachgebiet konzentrieren (Fachwissen über Funkzugangsnetze)

Die Nachteile sind:

- Das TA-Team muss an der Entwicklung von Testfällen beteiligt werden, einschließlich des Verständnisses der Anforderungen und der damit verbundenen Forschung.
- Jeder Entwickler im TA-Team muss ein gutes Verständnis des Funkzugangsnetzes auf Makroebene haben
- Der Testfallcode muss vor der Lieferung den Abnahmeprozess bestehen.
- Das TA-Team muss die nachfolgenden Aktualisierungen und Wartungsarbeiten durchführen

Es ist ersichtlich, dass beide Ansätze dem Testingenieur oder dem Entwickler des TA-Teams eine große Last an zusätzlicher Arbeit aufbürden. Mögliche Lösungen für dieses Problem werden im nächsten Abschnitt vorgestellt.

1.2 Zielsetzung

Ziel dieser Arbeit ist die Entwicklung eines Testfallgenerators für das automatisierte Test von RAN. der es dem Benutzer der es dem Benutzer ermöglicht, die benötigten Testfälle

über eine grafische Oberfläche zu erstellen. Mit diesem Tool müssen die Entwickler des TA-Teams nur den Logikcode hinter den einzelnen Testschritten und Tools entwickeln und aktualisieren, anstatt alle Aspekte des RAN-Fachwissens vollständig zu verstehen.

Zweitens sollte die Anwendung in der Lage sein, Testfälle zu verwalten und auszuführen. Die Anwendung sollte auch eine gute Erweiterbarkeit und Kompatibilität mit dem zu testenden System oder der Umgebung gewährleisten.

Um die Nutzung und Verwaltung zu erleichtern, sollte diese Anwendung auf einem zentralen Server installiert werden und eine Weboberfläche für die oben genannten Funktionen bieten.

1.3 Gliederung der Arbeit

Diese Arbeit ist in die nachfolgenden Kapitel unterteilt.

Der erste Teil ist die Einleitung, die kurz in das Themengebiet (RAN-Testautomatisierung) einführt und das Ziel der Arbeit darlegt.

Der zweite Teil befasst sich mit den Grundlagen der RAN-Automatisierungstests, einschließlich Hintergrundwissen über das zu testende System und die für RAN-Automatisierungstests verwendeten Tools und Frameworks. Zusätzlich wird in diesem Kapitel eine kurze Einführung in das Django-Framework gegeben, auf dem die Implementierung dieser Arbeit basiert.

Als nächstes wurden in dritten Kapitel die zuvor definierten Ziele analysiert, indem zunächst die Nutzungsszenarien für die Anwendung des automatischen Testfallgenerators definiert wurden. Dann wurde eine Anforderungsanalyse durchgeführt, um die funktionalen und nicht-funktionalen Anforderungen zu beschreiben.

In vierten Kapitel wurde ein bestimmter Entwurf vorgestellt und eine erste Bewertung vorgenommen. Auf der Grundlage der festgelegten Anforderungen werden Entwurfsentscheidungen getroffen und anhand von Diagrammen erläutert.

Im fünften Kapitel der Arbeit wird die Umsetzung des im vorherigen Kapitel beschriebenen Entwurfs vorgestellt. Anschließend wird die Umsetzung bewertet. Es wird aufgezeigt, welche Anforderungen in der Implementierung umgesetzt werden können.

Das letzte Kapitel, beinhaltet eine Reflexion über die zu Beginn der Arbeit gesetzten Ziele. Darüber hinaus wird ein Ausblick auf mögliche zukünftige Erweiterungen gegeben.

2 Grundlagen

2.1 Umgebung zur RAN-Testautomatisierung

Der RAN-Automatisierungstest kann in zwei Hauptteile unterteilt werden: das zu testende System (System Under Test, SUT) und die Automatisierungsumgebung. Das SUT enthält die direkt am Test beteiligte Hardware und Software, einschließlich der Basisstation, des UEs, des programmierbaren Dämpfungsglieds. Die Automatisierungsumgebung umfasst den Testserver, der für die Ausführung des Testfallcodes zuständig ist, das UE-Testtool (UE-Tester, Pseudonym) und seinen Steuer-PC sowie den Server, der für die Ausführung der Leistungstests zuständig ist.

Das gesamte System wird über das Labornetzwerk verwaltet. Die logische Verbindung der Elemente des SUT mit der Testautomatisierungsumgebung ist in Abbildung 2 dargestellt.

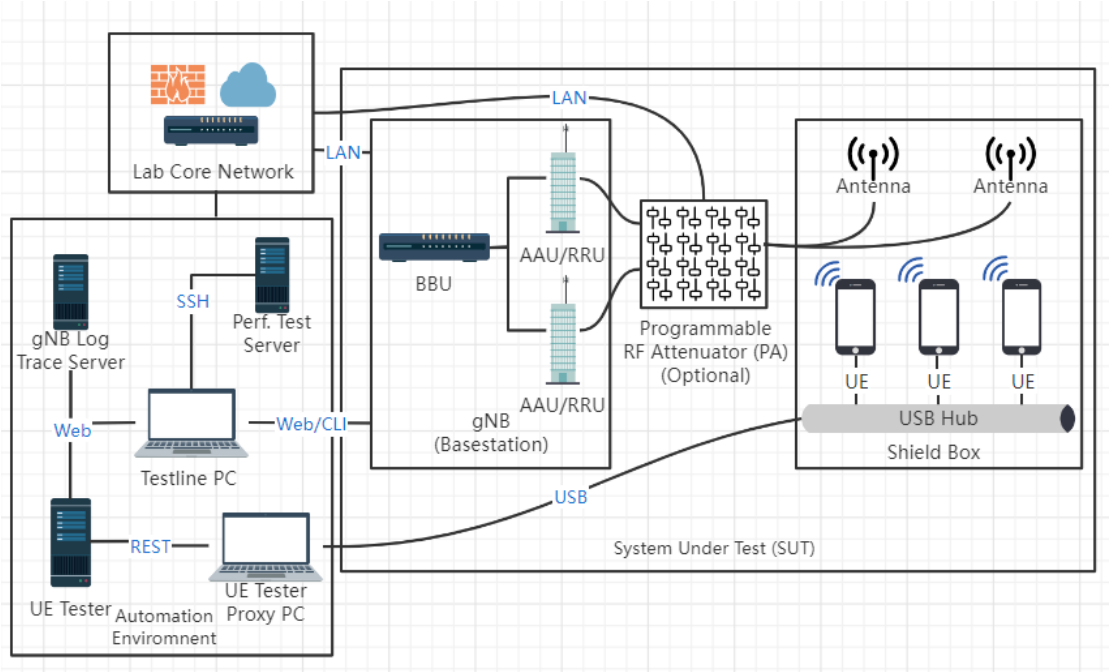

Abbildung 2.1: Testumgebung im Überblick

Im Gegensatz zum manuellen Testen ersetzt der Test-PC das Testpersonal und wandelt manuelle Vorgänge in die Ausführung von Testfallcode um. Er kommuniziert und führt Operationen mit anderen Servern in Automatisierungsumgebungen und dem SUT aus.

Zu diesen Operationen gehören unter anderem das Aufrufen von REST-APIs, das Simulieren von Webseitenklicks, das Initiieren von SSH-Verbindungen usw. Die Testfälle verwenden das Robot Framework, das auf Python basiert (später im Detail beschrieben). Der Testingenieur überträgt die vorbereitete Testsuite auf den Test-PC und führt sie durch. Er erhält nach der Ausführung automatisch erstellte Logs. Darüber hinaus werden einige Testfälle direkt in Python geschrieben und durchgeführt.

2.1.1 gNB/eNB

Bei den 5G-Standards des 3GPP wird die Basisstation als GNodeB (gNB) bezeichnet, die sich aus dem NodeB von 3G und dem Evolved Node B (eNB) von 4G entwickelt hat, wobei das G in 5G für "Next Generation" steht. [5GN19]

Unter den Testfällen für das RAN ist der gNB das Haupt-Testobjekt. Logischerweise hat jeder gNB seine Basisstations-ID, IP und andere Konfigurationsinformationen. Ein gNB hat eine oder mehrere logische Funkzellen (Cells), die durch seine Hardwarekonfiguration bestimmt werden, d.h. seine Antenneneinheit (AAU[1]/RRU[2]) am Frontend und das Kernnetz am Backend. Auf der physischen Ebene stimmen die Zelle und die RRU/AAU der Basisstation aufgrund von Faktoren wie Abdeckungs-winkel (Sektor), Frequenzband und sogar Kolokation (d. h., dass verschiedene Betreiber RAN-Anlagen gemeinsam nutzen) nicht genau überein. Wenn ein UE mit einer gNB/eNB verbunden ist, ist es tatsächlich mit einer logischen Zelle im RAN verbunden, wobei seine Konnektivität, Authentifizierung, Datenkonnektivität und andere Angelegenheiten von der BBU[3] in der gNB gehandhabt und weitergeleitet werden, während die Datenübertragung von der physischen Antenneneinrichtung durchgeführt wird. [Pet20]

Testingenieure können das System über die Weboberfläche des gNB/eNB oder das Befehlsterminal verwalten und konfigurieren. Während der Inbetriebnahme und der Tests können die Testingenieure auch Logs des gNB/eNB-Betriebs zur Analyse sammeln. Da die gNB/eNB-Software die Verarbeitung der Verbindungsschicht, der Netzschicht und der Steuerungsschicht umfasst, unterscheiden sich die Typen der Logs und die Methode, wie sie gesammelt werden. Im Falle von Anomalien erfassen die Testingenieure in der Regel auch Datenpakete, die Snapshots genannt werden, um den Status des gNB/eNB für eine kurze Zeitspanne zu erhalten, damit die Testingenieure das Problem weiter identifizieren und analysieren können.

Bei der Durchführung von Tests sind die wichtigsten Methoden zur Automatisierung von

[1] AAU: Active Antenna Unit: Antennenmodule von 5G Basisstationen

[2] RRU: Radio Remote Unit: Antennenmodule von 4G Basisstationen

[3] BBY: Baseband Unit Basisband-Einheit von Basisstationen

gNB/eNB-Vorgängen die Simulation von Webseitenoperationen und der Aufruf von Befehlszeilenschnittstellen. Auf dieser Grundlage entwickelte das TA-Team eine Reihe von Python-basierten Bibliotheken, um diesen Prozess zu automatisieren. Unter anderem basiert die Simulation von Weboperationen auf Selenium. Da der Bedarf an Logerfassung gestiegen ist, wurden webbasierte Anwendung entwickelt und verwendet, die mehrere Logs für gNB/eNB erfassen können. Diese Anwendung vereinfacht die Logerfassung und bietet eine Verwaltungs- und Speicherplattform. Über die bereitgestellte REST-API kann sie auch in der automatisierte Testkette integriert werden.

2.1.2 UE

In den 3GPP-Spezifikationen für 4G LTE und 5G NR ist ein Teilnehmergerät (UE) die Gerät, das direkt vom Endnutzer für die Kommunikation verwendet wird. Die Verbindung des UE mit dem eNB/gNB erfolgt gemäß den einschlägigen 3GPP-Spezifikationen. Das UE kann ein Mobiltelefon, ein mit einem mobilen Breitbandadapter ausgestattetes Laptop oder ein beliebiges anderes Gerät sein. Über die Funkschnittstelle ist das UE mit dem Node B /eNB / gNB verbunden und in einer logischen Zelle registriert [3GP22] [3GP221]. Bei der Testen von RANs wird der UE-Typ in der Regel nach den Anforderungen des Netzbetreibers (d. h. des Kunden) ausgewählt. Diese UEs sind typischerweise Smartphones mit Android/iOS-Betriebssystemen. In einigen speziellen Testfällen werden auch einige spezielle Geräte als UE verwendet.

Die UEs in Android / iOS können auch für kommerzielle und dedizierte Zwecke unterteilt werden. Kommerzielle Android / iOS UEs haben die gleiche Funktionalität wie handelsübliche Geräte. Die automatisierte Steuerung kommerzieller UEs kann durch die Entwicklung einer App über das SDK des Betriebssystemherstellers und den Empfang der entsprechenden Befehle erreicht werden. Android-Geräte können auch über ADB-Befehle ferngesteuert werden, z. B. für das Verbinden/Trennen der Netzwerkverbindung (Flugzeugmodus), den Neustart des Geräts, den Netzwerkmodus, die Bildschirm-steuerung (z. B. scrcpy) usw. [Goo22]

Der dedizierte UE hat die Möglichkeit, das Basisband zu debuggen und zu diagnostizieren. Sie wird den Testingenieuren vom Chip- oder Basisbandhersteller (z. B. Qualcomm, MTK, Huawei) oder einer autorisierten Drittpartei zur Verfügung gestellt. Die Hersteller definieren und liefern Basisband-Debugging-Schnittstellen/Befehle und spezifische Diagnose-/Testsoftware. Mit dem Test auf dem dedizierten UE kann der Testingenieur die Debugging-Informationen und das Kommunikationslog zwischen dem UE und der Basisstation erhalten. In der Praxis sind die Testfälle für kommerzielle UEs aufgrund von Faktoren wie Hardwarekosten, Softwarelizenzen, Entwicklungsschwierigkeiten bei der Automatisierung und Kundenanforderungen umfangreicher als für spezielle UEs.

Für UE-Automatisierungstests von Android- / iOS-Geräten bieten bestehende Tools eine Kombination aus Webserver, Android- / iOS-App und Steuer-PC, genannt UE Tester (Pseudonym). UE Tester implementiert Testaufgaben wie die Messung der Netzgeschwindigkeit (iPerf), Anrufe, SMS, Webbrowsing, Videostreaming und die oben erwähnten Steuerungsaufgaben. Gemäß den Anforderungen des jeweiligen Testfalls kann die Applikation auf der UE direkt über die Funkschnittstelle oder indirekt über den Steuer-PC mit dem Webserver des Testwerkzeugs verbunden werden. Häufiger wird die indirekte Verbindung über den Steuer-PC genutzt. In diesem Fall wird die gesamte Verbindung der Steuerungsebene über das kabelgebundene Netzwerk des Labors realisiert, das stabiler als die Funkschnittstelle ist. Durch den Anschluss an den Steuer-PC ist es zudem möglich, eine größere Bandbreite an Testmethoden und Steuerungsaufgaben zu realisieren. Außerdem bietet die Webseite des UE Testers selbst RESTful APIs, die von Automatisierungstestfällen aufgerufen werden können, um die gleichen Operationen wie auf der Webseite zu erzielen.

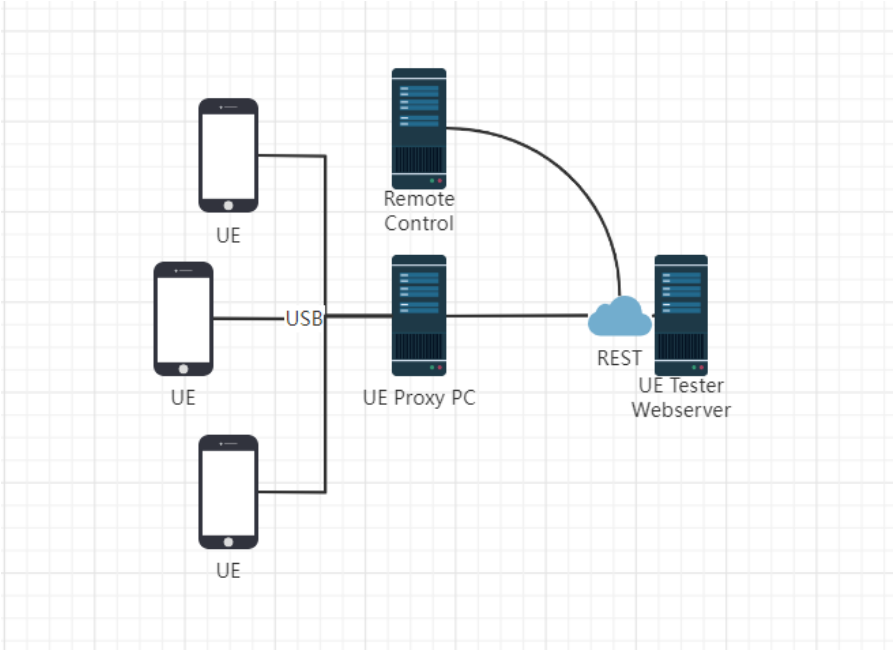

Abbildung 2.2: UE Tester Topologie

Das UE hat eine eindeutige Nummer IMEI (International Mobile Equipment Identity) und eine vom Hersteller bereitgestellte Seriennummer, und seine eingebaute Telefonnummer (SIM-Karte oder ESIM) hat eine eindeutige Kennung IMSI (International Mobile Subscriber Identity). Anhand dieser Felder kann man ein bestimmtes UE unter den Testwerkzeugen und Testfällen identifizieren und auswählen.

2.1.3 Programmierbare Dämpfungsglied (PA)

Programmierbare Dämpfungsglieder (Programmable Attenuator, PA) können die Hochfrequenz-Signalenergie anpassen, um die reale Signalschwäche zu simulieren, wodurch die komplexen realen Bedingungen der Feldsignalabdeckung in der Laborumgebung simuliert werden können. PA werden häufig bei der Leistungstests verschiedener Funkkommunikations-systeme eingesetzt. Das Kernelement der PA ist das Abschwächer, das die Amplitude oder Leistung des Signals reduziert, ohne die Signalform zu verzerren. [Gan16]

Im Labor ist der Eingangsanschluss des PA mit der Antenne des eNB/gNB und der Ausgangsanschluss mit der Antenne des abgeschirmten Gehäuses des Endgeräts verbunden. Mit der vom PA-Hersteller zur Verfügung gestellten Software kann der Testingenieur den Dämpfungswert jedes Ports (Kanals) einstellen, um Änderungen des UE-Signals zu simulieren. Im Testszenario ist es auch möglich, 2 oder mehr gNB/eNB-Zellensignale über den PA mit der Shield Box zu verbinden und so die Bewegung von Nutzern zwischen Basisstationen zu simulieren.

PA-Geräte gibt es von verschiedenen Herstellern, und die gleichen Hersteller bieten auch eine Vielzahl von PA-Modellen an. Verschiedene PA-Geräte können unterschiedliche Betriebssoftware, Dämpfungseinstell-bereiche (z.B. 0dB-90dB, 0dB-120dB usw.) und Kanalzahlen haben.

Die Betriebssoftware bietet Funktionen wie die Einstellung der Dämpfungswerte einzelner oder mehrerer Kanäle und sogar die Aufzeichnung und Wiedergabe für Gruppen von Kanälen. Der PA-Hersteller stellt auch APIs zur Verfügung, die den Funktionen der Betriebssoftware entsprechen und es ermöglichen, die Steuerung des PA in den Testfällen zu automatisieren. Abbildung 4 unten zeigt eine typische Benutzeroberfläche der PA-Betriebssoftware des Herstellers HBTE. [HBT22]

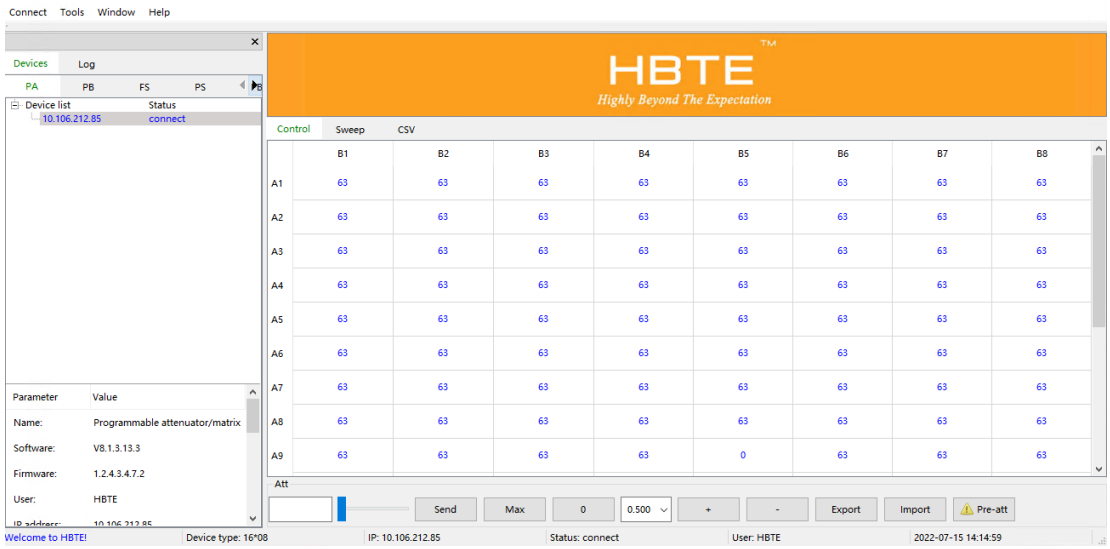

Abbildung 2.3: Benutzeroberfläche der PA-Betriebssoftware von HBTE

2.2 Robot-Framework

Robot Framework ist ein generisches Framework für automatisierte Tests, das hauptsächlich für Abnahmetests verwendet wird. Es kann für die Testautomatisierung und die Automatisierung von Roboterprozessen (RPA) verwendet werden. [Rob22]

Die Grundidee des Robot Frameworks wurde 2005 von Pekka Klärck entwickelt. Es handelt sich um ein schlüsselwortgesteuertes Test-Framework, das eine tabellarische Testdaten-Syntax verwendet. [Lau06] Die erste Version des Robot Framework wurde im selben Jahr bei Nokia Networks entwickelt und wird im gesamten Unternehmen intensiv genutzt. Es wird zum Testen verschiedener Geräte, Softwaresysteme und Protokolle über GUIs, APIs und verschiedene andere Schnittstellen verwendet. Derzeit ist Robot Framework ein Python-basiertes Open-Source-Framework mit einer aktiven Gemeinschaft von Mitwirkenden. Die aktuellen Anwendungsfälle für den automatisierten Test von Funkzugangsnetzen werden auf der Basis von Robot Framework 3 entwickeln.

2.2.1 Struktur und Syntax

Die Testsuite für das Robot-Framework ist die Datei mit der Endung .robot. Die Datei ist in 4 Teile unterteilt, nämlich Settings , Variables , Test Cases und Keywords.

Im Teil Settings können Entwickler die aktuelle Datei konfigurationen, wie z. B. Python-Bibliotheken oder andere .robot-Dateien referenzieren, Setup/Teardown festlegen usw.

Im Teil Variables können die für den Test benötigten Variablen definiert werden. Variablen im Robot Framework haben die Form ${variable_name}. Dabei steht das Präfix „$“ für eine Variable, „@“ für eine Liste und „&“ für ein Wörterbuch. Die folgende Abbildung 2.4 zeigt einige Beispiele für Variablen.

```
*** Variables ***
${var_name}     value
@{list_name}    a b c ${var1}
&{dict_name}    key1=sf key2=${list1}
```

Abbildung 2.4: Robot Framework Variables

Der Teil Test Cases definiert die spezifische Testfälle. Darunter können die Testfälle interne oder externe Schlüsselwörter (Keyword) aufrufen, wobei externe Schlüsselwörter zuvor in das Setup importiert werden müssen. Die für das Schlüsselwort erforderlichen Parameter müssen auf der rechten Seite des Schlüsselworts angegeben werden. Und Der

Rückgabewert des Schlüsselworts kann als Variable auf der linken Seite empfangen werden. Die folgende Abbildung 2.5 zeigt Beispiele für die Verwendung im Testfall.

```
*** Test Cases ***
Test Case Name
    Keyword1
    Keyword2     ${arg1}    ${arg2}    # Keyword mit 2 Argumenten
    Keyword3     @{arg3}               # KW mit Liste von Argumenten
    ${result}    Keyword4              # KW mit Rückgabewert
```

Abbildung 2.5: Robot Framework Testfall

Im Teil Keywords werden die eigene Schlüsselwörter für die Robot-Datei definiert. Ähnlich wie im Test Case kann das Keyword andere Keywords innerhalb oder außerhalb der Datei aufrufen. Der Unterschied besteht darin, dass Keywords Parameter und Rückgabewerte definieren und in verschiedenen Test Cases/ Keywords wiederverwendet werden können, ähnlich wie Funktionen in Programmiersprachen.

In den Test-Cases und Keywords können auch for-Anweisungen zur Ausführung von Schleifen verwendet werden. In Robot Framework 3 können If-Anweisungen durch den Aufruf integrierter Keywords wie "Run Keyword If", "Set Variable If" erreicht werden. Die folgende Abbildung 2.6 zeigt Beispiele für die Verwendung von For- und If-Anweisungen in Keywords.

```
*** Keywords***
Varargs Keyword
    [Arguments]    @{args}
    Log to console    --------
    FOR    ${item}    IN    @{args}
        Log to console      ${item}
    END


Number Detect
    [Arguments]    ${number}
    Run Keyword If    ${number}>10    Log    Number is large than 10!
    ...    AND    Return From Keyword    ${True}
    [Return]    ${False}
```

Abbildung 2.6: Robot Framework Keywords

Robot Framework verfügt über integrierte Keywords, die häufig in Test-Frameworks verwendet werden, wie z. B. Should Be True, Should Be Equal usw. Diese integrierten Keywords können ohne expliziten Verweis aufgerufen werden.

2.2.2 Ausführung und Logs

Robot-Dateien können als Testsuite ausgeführt werden. Die Ausführung erfolgt durch einen Befehl wie "robot testsuite_name.robot". Am Ende des Befehls können einige allgemeine Parameter angehängt werden. Ein Beispiel ist "--dryrun", mit dem nur geprüft wird, ob die Testsuite selbst ordnungsgemäß ausgeführt werden kann, ohne den eigentlichen Python-Code aufzurufen.

Während der Ausführung der Tests werden die Ergebnisse für Erfolg (PASS) und Misserfolg (FAIL) für jeden Testfall ausgedruckt. Die Tests erzeugen auch Berichte und Logs. Der Benutzer kann die Ergebnisse jedes Schrittes im Testfall in den Logs sehen.

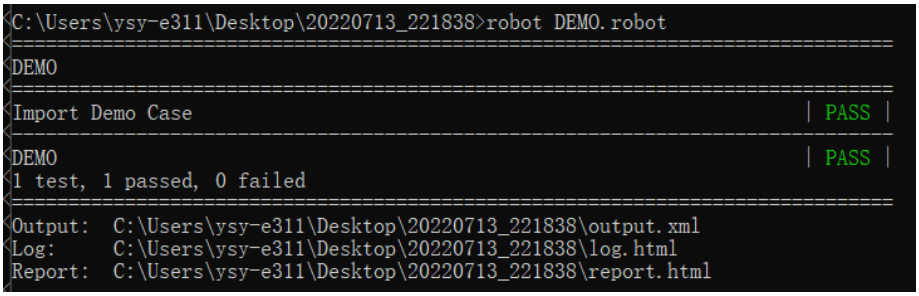

Abbildung 2.7:. Testausführung Robot Framework

2.3 Django-Framework

Django ist ein fortschrittliches Python-Web-Framework, das eine schnelle Entwicklung und ein sauberes, praktisches Design fördert. Django wurde 2005 unter der BSD-Lizenz veröffentlicht und zielt darauf ab, die Entwicklung von datenbankgestützten Websites zu vereinfachen. Der Fokus liegt auf der Wiederverwendbarkeit und „Pluggability“ von Komponenten, agiler Entwicklung und der DRY-Regel (Don't Repeat Yourself). In Django sind sogar Konfigurationen und Datenmodelle in Python implementiert. [Dja22]

Der Kern des Django-Frameworks umfasst: Ein objekt-relationaler Mapper, der als Medium zwischen dem Datenmodell (definiert als Python-Klasse) und der relationalen Datenbank dient. Ein URL-Verteiler, der auf regulären Ausdrücken basiert. Ein Sichtsystem für die Bearbeitung von Anfragen. Eine Template-System , um Daten auf das Web-frontend zu bringen.

Zum Kern-Framework von Django gehören außerdem die folgenden: [Wik02]

- Webserver zur Entwicklung und zum Testen
- Formularsystem: für die Umwandlung zwischen HTML-Formularen und datenbankfähigen Daten.
- Cache Framework mit verschiedenen Caching-Optionen verfügbar
- Middleware-Unterstützung, die eine Einmischung in alle Phasen der Anfrageverarbeitung ermöglicht.
- Integriertes Verteilungssystem: Die Komponenten der Anwendung können über vordefinierte Signale miteinander kommunizieren.
- Serialisierungssystem: Dient der Umwandlung von Django-Modellinstanzen und XML/JSON-Formaten.
- System zur Erweiterung der Fähigkeiten der Template-System

3 Analyse

Der Zweck dieses Kapitels ist die Bestimmung der Szenarien und Anforderungen für den Testfallgenerator. Zunächst werden die Nutzungsszenarien aufgelistet und zusammengefasst. Anschließend werden die spezifischen Funktionen festgelegt, die implementiert werden sollen.

3.1 Szenarien

In diesem Abschnitt werden die Anwendungsszenarien des Testfallgenerators mit natürlicher Sprache beschrieben. Die Szenarien sind nach Typ in Testfallverwaltung, Testumgebungsverwaltung, Testdurchführung, Ergebnisanzeige und Verwaltungen von Testschritt/Testumgebungselements unterteilt.

3.1.1 Testfällen verwalten und bearbeiten

Nachdem sich ein Benutzer (Testingenieur) bei seinem Konto angemeldet hat, wird im Browser eine Seite zur Verwaltung von Testfällen angezeigt, auf der alle von ihm erstellten

Testfälle aufgelistet sind. Nachdem der Benutzer auf die Schaltfläche "Create New Testcase" geklickt und den Namen des neuen Testfalls eingegeben hat, wird eine Seite zur Bearbeitung und Gestaltung des Testfalls angezeigt und ein neuer Testfall mit leerem Inhalt und Standardeinstellungen erstellt. Benutzer können Testfälle bearbeiten, löschen und exportieren, indem sie auf einen Testfall in der Liste klicken.

Wenn der Benutzer auf einen bestehenden Testfall klickt, wird die Seite zur Bearbeitung von Testfällen angezeigt, genau wie bei der Erstellung eines neuen Testfalls. Auf der Seite zur Bearbeitung von Testfällen können Benutzer auf die Option Speichern klicken, um ihre Änderungen zu übernehmen und zur Testfallverwaltung zurückzukehren. Ber Benutzer kann auch auf Abbrechen klicken, um die Änderungen zu verwerfen und den Vorgang zu beenden.

Bei der Bearbeitung von Testfällen kann der Benutzer die Liste der voreingestellten Testschritte und das Testfallbearbeitungspanel mit Testschritten, Setup und Teardown sehen. Der Benutzer wählt einen voreingestellten Testschritt aus und klickt auf " Add", woraufhin der Testschritt in das Innere des Hauptteils, des Setup- oder Teardown-teil des Testfalls eingefügt wird. In Bearbeitungspanel für Testfälle kann Benutzer die Reihenfolge der hinzugefügten Testschritte ändern oder einzelne Testschritte entfernen. Wenn der Benutzer einen einzelnen Testschritt auswählt, werden die für diesen Testschritt erforderlichen Parameter in einem Bearbeitungsfeld angezeigt, und ihre Werte können bearbeitet werden.

Die Seite zur Testfallverwaltung ermöglicht auch den Exportieren und Importieren von Testfällen, die in ein strukturiertes Dateiformat konvertiert werden können. Wenn der Benutzer einen Testfall auswählt und auf die Export klickt, wird eine strukturierte Konfigurationsdatei erstellt und auf den Computer des Benutzers heruntergeladen. Wenn der Benutzer auf die Importoption klickt, kann er die Datei auf dem lokalen Computer auswählen und hochladen, und einen neuen Testfall hinzufügen.

3.1.2 Testumgebung verwalten

Mit der Testumgebung kann der Benutzer die Informationen über das zu testende System (SUT) und die mit dem Automatisierungstest verbundenen Elemente in einer Sammlung konfigurieren. Diese Informationen werden nur gelesen, wenn der Testfall ausgeführt wird. Die Testfälle und ihre Testschritte selbst müssen diese Informationen nicht direkt enthalten, wodurch die Testumgebung von den Testfällen entkoppelt werden kann. Die Testumgebung besteht aus den Objektentitäten des SUT oder der Automatisierungsumgebung. Jede dieser Entitäten kann logisch auf eine Netzelemententität abgebildet werden. Jedes Objekt hat einen festen Satz von Parameternamen, und die Werte werden vom Benutzer eingegeben. Bei der Testausführung, wenn die Parameter eines Testschritts im Testfall als Standard oder leer gelassen werden, liest der Test die Konfiguration in der Testumgebung. Jeder Parameter

des Testumgebungselements hat auch einen Variablennamen im Test, den der Benutzer in den Testfall eingeben kann. Beispielsweise kann die ID der ersten konfigurierten Basisstation durch "basestation.1.id" dargestellt werden.

Nachdem der Benutzer auf die Registerkarte Testumgebung geklickt hat, wird eine Seite zur Verwaltung der Testumgebung angezeigt. Ähnlich wie Testfälle können auch Testumgebung von Benutzer erstellt, bearbeitet und gelöscht werden. Wenn der Benutzer auf "New Test Environment" oder "Edit" klickt, wird eine Seite zur Bearbeitung der Testumgebung angezeigt.

Auf der Seite zur Bearbeitung der Testumgebung kann Benutzer vordefinierte Testumgebungselemente hinzufügen. Sobald der Benutzer unter einem Element auf „Add“ klickt, wird ein entsprechendes Objekt zur Testumgebung hinzugefügt. Wenn ein Konfigurationsobjekt ausgewählt wird, werden die darin enthaltenen Parameter angezeigt. Der Benutzer kann den Wert dieser Parameter ändern. Zusätzlich kann der Benutzer mehrere Einheiten desselben Typs von Testumgebungselementen hinzufügen, jede mit einer eindeutigen Nummer zu ihrer Identifizierung.

Ähnlich wie auf der Bearbeitungsseite von Testfällen kann der Benutzer auf der Bearbeitungsseite von Testumgebung die Änderung speichern oder abbrechen. Nach dem Speichern oder Abbrechen kann der Benutzer zur Verwaltungsseite der Testumgebung zurückkehren.

3.1.3 Test Set, Testdurchführung und Ergebnisanzeige

Mit der Konfiguration des Testsatzes (Test Set) können Benutzer seinen Testfall und seine Testumgebung zu ausführbaren Testsuiten kombinieren. Ähnlich wie bei Testfällen und Testumgebungen kann der Benutzer auf der Webseite Testsätze hinzufügen, bearbeiten und löschen. Im Gegensatz zu den ersten beiden kann der Benutzer den Testsatz jedoch auch ausführen und die Ergebnisse einsehen.

Auf der Verwaltungsseite von Testsätzen kann der Benutzer alle seinen Testsätze einsehen. Jeder Testsatz ist mit dem aktuellen Status und dem Ergebnis des letzten Laufs (PASS oder FAIL) gekennzeichnet. Nachdem der Benutzer auf einen Testsatz geklickt hat, werden dessen Details angezeigt, einschließlich der verwendeten Testfälle/ Testumgebung und der Ergebnisse der letzten Testergebnisse. Die Benutzer können auch die von dem Testsatz verwendeten Testfälle/Testumgebung ändern. In den Details des Testsatzes können Benutzer auf Execute (Ausführen) klicken, um die Ausführung der Testfälle auszulösen. Danach aktualisiert das System das Log des Testsatzes mit seinem Status und seinen Ergebnissen. Zusätzlich zu den letzten Testläufen kann der Benutzer auch alle Einträge zu diesem Testsatz einsehen, indem er auf "All Logs" klickt. Dazu gehören die Start- und Endzeiten der Tests, die Ergebnisse und die von den Tests erzeugten Logs. Darüber hinaus kann der Benutzer die generierte Testsuite über die

Exportoption herunterladen. Die Testsuite wird mit den im Server enthaltenen Testschritten verpackt, so dass der Benutzer sie auch auf anderen Rechnern ausführen können.

3.1.4 Testschritte und Testumgebungselemente aktualisieren

Das System sollte auch eine Benutzeroberfläche für die Verwaltung der Testschritte und Testumgebungselemente bieten. Sobald sich der Wartungspersonal bei seinem Konto anmeldet, kann er alle Testschritte und Testumgebungselemente im System einsehen. Durch Klicken auf die entsprechende „Add“ Option kann er neue Testschritte oder Testumgebungselemente hinzufügen. Durch Anklicken eines Eintrags in der Liste der Testschritte/Testumgebungselemente kann er den entsprechenden Eintrag einsehen und bearbeiten.

Die für die Testschritte erforderlichen Keywords/Codes sollten von den Wartungspersonal entweder innerhalb oder außerhalb des Systems aktualisiert werden. Im Idealfall sollte die Aktualisierung des Testschrittcodes eine Änderung der entsprechenden Daten im System auslösen.

3.1.5 Überblick

Durch die Zusammenfassung der oben genannten Szenarien lassen sich die wichtigsten funktionalen Punkte des Systems zusammenfassen, darunter.

- Testfällen ansehen, hinzufügen, löschen und bearbeiten
- Testumgebungen ansehen, hinzufügen, löschen und bearbeiten
- Testsätze ansehen, hinzufügen, löschen und bearbeiten
- Testsätze ausführen
- Testergebnisse ansehen
- Testschritte verwalten, enthält Hinzufügen, löschen und bearbeiten
- Testumgebungselementen verwalten, enthält Hinzufügen, löschen und bearbeiten

Die nachstehende Abbildung 3.1 zeigt ein knappes Anwendungsfalldiagramm auf der Grundlage dieser Zusammenstellung.

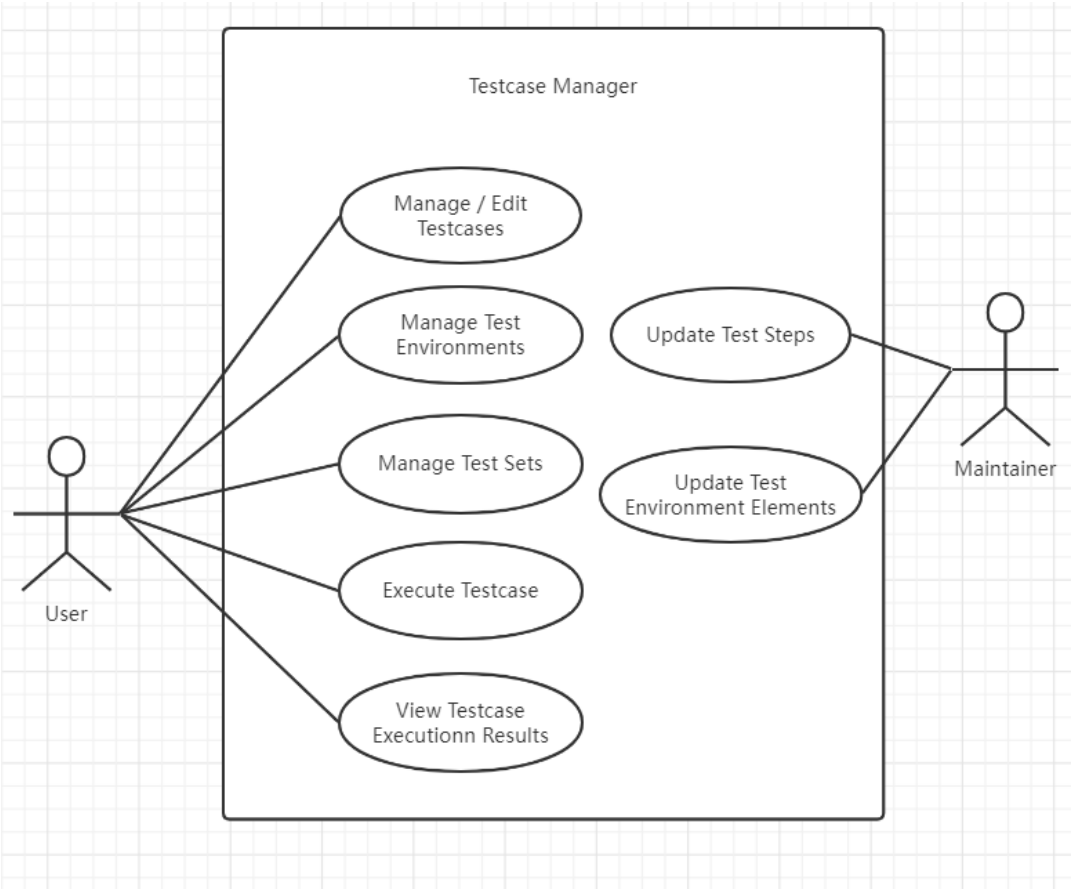

Abbildung 3.1: Anwendungsfalldiagramm

3.2 Anforderungsanalyse

Die Anforderungsanalyse bezieht sich auf die gesamte Arbeit, die zur Festlegung des Zwecks, des Umfangs, der Definition und der Funktionalität eines neuen Systems beim Aufbau eines neuen oder bei der Änderung eines bestehenden Systems oder Produkts erforderlich ist. Dazu gehören die Berücksichtigung der Anforderungen verschiedener Interessengruppen, die Ermittlung widersprüchlicher Anforderungen, die Festlegung von Kompromissen zwischen widersprüchlichen Anforderungen und die Analyse, Dokumentation, Validierung und Verwaltung von Software- und Systemanforderungen. [Kot98]

Dieser Abschnitt beschreibt die funktionalen und nicht-funktionalen Anforderungen an den Testfallgenerator auf der Grundlage der zuvor beschriebenen Szenarien.

3.2.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die Funktionen und Verhaltensweisen, die das System erfüllen soll.

Benutzeranmeldung

Das System sollte es den Benutzern ermöglichen, sich in das System einzuloggen.

Der Benutzer soll:

- Zugriff auf die vom System bereitgestellte Benutzeroberfläche über die URL
- Benutzernamen und sein Passwort eingeben und auf Login klicken.

Das System soll:

- die URL bereitstellen, unter der die Benutzeroberfläche aufgerufen werden kann
- für nicht eingeloggte Benutzer den Login-Seite anzeigen.
- den Benutzernamen und das Passwort des Benutzers überprüfen und den Benutzer anmelden, wenn sie korrekt sind.
- die Anmeldung ablehnen und dem Benutzer eine Fehlermeldung anzeigen, wenn der Benutzername oder das Passwort falsch sind.
- nach der Benutzeranmeldung eine Registerkarte für den Zugriff auf die Seite zur Verwaltung von Testfällen, Testumgebungen und Testsätzen bereitstellen.

Testfälle ansehen

Das System sollte es den Benutzern ermöglichen, ihre Testfälle im System zu sehen.

Das System soll:

- eine Liste aller Testfälle im aktuellen System des Benutzers und deren abhängige Optionen bereitstellen.

Testfall erstellen

Das System sollte es dem Benutzer ermöglichen, neue Testfall im System zu erstellen.

Der Benutzer soll:

- durch einem Klick auf die Option, einen neuen Testfall erstellen.
- Namen des neuen Testfalls festlegen und Senden klicken.

Das System soll:

- eine Option zum Erstellen neuer Testfälle bieten
- nachdem der Benutzer „neuer Testfälle“ geklickt hat, wird die Seite zur Benennung eines neuen Testfalls angezeigt.
- Nachdem der Benutzer den Namen des neuen Testfalls angegeben hat, wird der neue Testfall für den Benutzer mit diesem Namen im System gespeichert.

Testfall löschen

Das System sollte es dem Benutzer ermöglichen, ihre Testfälle zu löschen.

Der Benutzer soll:

- einen Testfall aus der Liste auswählen und auf die Option Exportieren klicken.
- auf das Dialogfenster klicken, um die Löschung zu bestätigen.

Das System soll:

- Löschenoptionen für jeden Testfall in der Liste anbieten.
- Wenn der Benutzer auf "Löschen" klickt, erscheint ein Dialogfenster zur Bestätigung des Löschvorgangs.
- Testfälle werden erst dann aus dem System gelöscht, wenn der Benutzer die Löschung in einem Dialogfenster bestätigt.

Testfall importieren

Das System sollte es den Benutzern ermöglichen, Testfälle zu importieren.

Der Benutzer soll:

- auf der Seite Testfallverwaltung auf Neuen Testfall importieren klicken.
- in dem Dialogfenster, die zu importierende Datei auswählen und auf Hochladen klicken.

Das System soll:

- eine Option zum importieren neuer Testfälle bieten.
- nachdem der Benutzer „Neue Testfall importieren“ geklickt hat, wird ein Dialogfenster für den Datei-Upload eingeblendet.
- versuchen, einen neuen Testfall mit den vom Benutzer hochgeladenen Daten zu erstellen.
- den Testfall im System speichern, wenn die Erstellung erfolgreich war.
- eine Fehlermeldung an den Benutzer senden, wenn das Lesen der vom Benutzer hochgeladenen Datei oder das Erstellen eines neuen Testfalls fehlschlägt.

Testfall exportieren

Das System sollte es den Benutzern ermöglichen, ihre Testfälle zu exportieren.

Der Benutzer soll:

- einen Testfall aus der Liste auswählen und auf die Option Exportieren klicken.
- Speicherort der exportierten Datei auf dem lokalen Computer auswählen.

Das System soll:

- Exportoptionen für jeden Testfall in der Liste anbieten.
- Nachdem der Benutzer auf die Exportoption geklickt hat, wird die dem Testfall entsprechende serialisierte Datei erstellt und auf den Computer des Benutzers heruntergeladen.

Testfall bearbeiten

Das System sollte es den Benutzern ermöglichen, ihre Testfälle zu bearbeiten.

Der Benutzer soll:

- den zu bearbeitenden Testfall in der Liste auswählen und auf die Bearbeiten Option klicken.
- Testschritte zu dem Testfall hinzufügen und löschen.
- die Parameter der einzelnen Testschritte konfigurieren.
- den Testschritt bei Bedarf als Einrichtungs- oder Abrüstungsschritt markieren.
- die Reihenfolge der Testschritte im Testfall ändern, falls erforderlich.
- Name des Testfalls ändern, falls erforderlich.
- auf Speichern oder Abbrechen, um die Bearbeitung des Testfalls zu beenden.

Das System soll:

- Bearbeitungsoptionen für jeden Testfall bieten.
- eine Benutzeroberfläche zur Bearbeitung von Testfällen anbieten, die die Testschritte im Testfall und ihre Parameter liest und anzeigt, nachdem der Benutzer auf Bearbeiten geklickt hat.
- eine Liste von Testschritten bereitstellen, auf die der Benutzer klicken kann, um den entsprechenden Testschritt hinzuzufügen.
- in jedem Testschritt des Testfalls eine Schaltfläche zum Löschen bieten, auf die der Benutzer klickt, um den Schritt aus dem Testfall zu entfernen.
- Option zur Bezeichnung von Testschritten als Setup/Teardown bereitstellen.
- ein Bearbeitungsfenster für jeden Testschritt des Testfalls anbieten, damit der Benutzer die Parameterwerte des Testschritts eingeben oder ändern kann.
- Auf-/Abwärts-Option für jeden Testschritt bieten, die die Reihenfolge der entsprechenden Testschritte ändert.

- ein Eingabefeld in die Testfallbearbeitungsseite einfügen, um den Namen des Testfalls zu ändern.
- Wenn der Benutzer auf Speichern klickt, werden der Testschritt und seine Parameter auf der Seite sowie der Name des Testfalls auf den im System gespeicherten Testfall aktualisiert.
- Wenn der Benutzer auf Abbrechen klickt, werden die Änderungen nicht gespeichert und die Testfallverwaltungsseite wird an den Benutzer zurück-gegeben.

Testumgebung ansehen

Das System sollte es den Benutzern ermöglichen, ihre Testumgebungen zu ansehen.

Das System soll:

- eine Liste aller Testumgebungen im aktuellen System des Benutzers und deren abhängige Optionen bereitstellen.

Testumgebung erstellen

Das System sollte es den Benutzern ermöglichen, neue Testumgebung zu erstellen.

Der Benutzer soll:

- durch einem Klick auf die Option, einen neuen Testumgebung erstellen.
- Namen der neuen Testumgebung festlegen und Senden klicken.

Das System soll:

- eine Option zum Erstellen einer neuen Testumgebung anbieten.
- nachdem der Benutzer die Option geklickt hat, wird die Seite zur Benennung einer neuen Testumgebung angezeigt.
- Nachdem der Benutzer den Namen der neuen Testumgebung angegeben hat, wird die neue Testumgebung für den Benutzer mit diesem Namen im System gespeichert.

Testumgebung löschen

Das System sollte es den Benutzern ermöglichen, ihre Testumgebungen zu löschen.

Der Benutzer soll:

- Eine Testumgebung aus der Liste auswählen und auf die Option Exportieren klicken.

- auf das Dialogfenster klicken, um die Löschung zu bestätigen.

Das System soll:

- Löschenoptionen für jede Testumgebung in der Liste anbieten.
- Wenn der Benutzer auf "Löschen" klickt, erscheint ein Dialogfenster zur Bestätigung des Löschvorgangs.
- Testumgebung werden erst dann aus dem System gelöscht, wenn der Benutzer die Löschung im Dialogfenster bestätigt.

Testumgebung bearbeiten

Das System sollte es den Benutzern ermöglichen, ihre Testumgebungen zu bearbeiten.

Der Benutzer soll:

- die zu bearbeitende Testumgebung in der Liste auswählen und auf die Bearbeiten Option klicken.
- Testumgebungselemente zur Testumgebung hinzufügen und löschen.
- die Parameter der einzelnen Testumgebungselement aktualisieren.
- Name der Testumgebung ändern, falls erforderlich.
- auf Speichern oder Abbrechen, um die Bearbeitung der Testumgebung zu beenden.

Das System soll:

- Bearbeitungsoptionen für jeden Testumgebung bieten.
- eine Benutzeroberfläche zur Bearbeitung von Testumgebung anbieten, die die Testumgebungselemente im Testumgebung und ihre Parameter liest und anzeigt, nachdem der Benutzer auf Bearbeiten geklickt hat.
- eine Liste von Testumgebungselemente bereitstellen, auf die der Benutzer klicken kann, um den entsprechenden Element zur Testumgebung hinzuzufügen.
- in jedem Element eine Schaltfläche zum Löschen bieten, die der Benutzer anklickt, um das Element aus der Testumgebung zu entfernen.
- ein Bearbeitungsfenster für jede Testumgebungselement anbieten, damit der Benutzer die Parameterwerte der Testumgebungselement eingeben oder ändern kann.
- ein Eingabefeld zum Ändern des Namens der Testumgebung bereitstellen.
- Wenn der Benutzer auf Speichern klickt, werden die Elemente und ihre Parameter auf der Seite sowie der Testumgebungsname mit der im System gespeicherten Testumgebung aktualisiert.
- Wenn der Benutzer auf Abbrechen klickt, werden die Änderungen nicht gespeichert und die Testumgebungverwaltungsseite wird an den Benutzer zurückgegeben.

Testsatz ansehen

Das System sollte es den Benutzern ermöglichen, ihre Testsätze zu ansehen

Das System soll:

- eine Liste aller Testsätz im aktuellen System des Benutzers und deren abhängige Optionen bereitstellen.
- den Echtzeit-Ausführungsstatus und den groben Fortschritt des Testsatzes anzeigen, wenn der Testsatz gerade ausgeführt wird.
- für jeden Testsatz den Laufstatus und die letzten Laufergebnisse anzeigen.

Testsatz erstellen

Das System sollte es den Benutzern ermöglichen, neuen Testsatz zu erstellen.

Der Benutzer soll:

- durch einem Klick auf die Option, einen neuen Testsatz erstellen.
- dem Testsatz einen Namen, einen Testfall und eine Testumgebung zuordnen und dann auf Speichern klicken.

Das System soll:

- eine Option zum Erstellen neuen Testsatz bieten
- das Dialogfenster für die Details des Testsatzes anzeigen, nachdem der Benutzer auf die Option geklickt hat.
- Wenn der Benutzer auf Speichern klickt, wird der neue Testsatz im System gespeichert.

Testsatz bearbeiten

Das System sollte es den Benutzern ermöglichen, ihren Testsatz zu bearbeiten.

Der Benutzer soll:

- Anzeigen und Ändern der Konfiguration des Testsatzes durch Anklicken von Testsatz in der Liste.
- dem Testsatz einen Namen, einen Testfall und eine Testumgebung zuordnen und dann auf Speichern klicken.

Das System soll:

- das Dialogfenster für die Details des Testsatzes anzeigen, nachdem der Benutzer in der Liste auf Testsatz geklickt hat.

- alle Testfälle und die Testumgebung des Benutzers im Dialogfenster für die Details des Testsatzes bereitstellen.
- der Testsatz im System aktualisiert, wenn der Benutzer auf aktualisieren klickt.

Testsatz ausführen

Das System sollte es den Benutzern ermöglichen, ihren Testsatz auszuführen.

Der Benutzer soll:

- den Testsatz in der Liste auswählen und auf die Option Ausführen klicken.

Das System soll:

- für jeden Testsatz eine Option zum Ausführen anbieten.
- die Testfälle und die Testumgebung des Testsatzes einlesen, die zur Ausführung verfügbare Testsuite zusammenstellen, ausführen und den Laufstatus des Testsatzes aktualisieren.
- den Laufstatus aktualisieren und die Ergebnisse ablesen, nachdem der Testsatz beendet ist, und ein Testlog erstellen.

Testsatz als ausführbare Testsuiten exportieren

Das System sollte es den Benutzern ermöglichen, ihren Testsatz als ausführbares Testsuite-Paket zu exportieren.

Der Benutzer soll:

- den Testsatz in der Liste auswählen und auf die Option Exportieren klicken.

Das System soll:

- für jeden Testsatz eine Option zum Exportieren anbieten.
- die Testfälle und die Testumgebung des Testsatzes einlesen, die Testsuite zusammenstellen und verpacken, dann auf den Computer des Benutzers heruntergeladen.

Ergebnisse der Testdurchführung ansehen

Das System sollte es den Benutzern ermöglichen, die Ergebnisse ihres Testsatzlaufs zu sehen.

Der Benutzer soll:

- den Testsatz in der Liste aufklappen, um einen Überblick über die Ergebnisse der

letzten Läufe zu sehen.
- auf Option klicken, um alle Läufe des Testsatzes zu sehen, einschließlich Laufzeiten, Ergebnisse und alle detaillierte Logs.

Das System soll:

- einen Überblick über die Ergebnisse der letzten Läufe im Erweiterungspanel des Testsatzes bieten.
- die Möglichkeit bieten, alle Logs des Testsatzes anzuzeigen
- Nachdem der Benutzer auf Option „Alle Logs anzeigen“ geklickt hat, zeigt die neue Detailseite alle vergangenen Läufe des Testsatzes an, einschließlich der Zeit, des Ergebnisses und des Links zur Logdatei jedes Laufs.

Testschritte verwalten

Das System sollte es dem Wartungspersonal ermöglichen, die Testschritte zu verwalten.

Der Wartungspersonal soll:

- Testschritte hinzufügen, löschen oder ändern

Das System soll:

- Oberfläche oder Schnittstellen zum Hinzufügen, Löschen und Ändern von Testschritten bieten.
- neue und geänderte Testschritt in das System aktualisieren.

Testumgebunselemente verwalten

Das System sollte es dem Wartungspersonal ermöglichen, die Testumgebunselemente zu verwalten.

Der Wartungspersonal soll:

- Testumgebunselemente hinzufügen, löschen oder ändern.

Das System soll:

- Oberfläche oder Schnittstellen zum Hinzufügen, Löschen und Ändern von Testumgebunselemente bieten.
- neue und geänderte Testumgebunselement in das System aktualisieren.

Benutzern verwalten

Das System sollte es dem Wartungspersonal ermöglichen, die Benutzerkonten zu verwalten.

Der Wartungspersonal soll:

- Benutzerkonten hinzufügen, löschen oder ändern

Das System soll:

- Oberfläche oder Schnittstellen zum Hinzufügen, Löschen und Ändern von Benutzerkonten bieten.
- neue und geänderte Benutzerkonto in das System aktualisieren.

3.2.2 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben die Qualität der vorgeschlagenen Funktionalität zum Zeitpunkt der Implementierung. Als solche haben sie einen erheblichen Einfluss auf den Ressourcenverbrauch, die Entwicklung und die Wartung. Darüber hinaus tragen diese Anforderungen zur Akzeptanz des Systems durch die Benutzer bei. [Ste05]

Zuverlässigkeit

Zuverlässigkeit ist eine Grundvoraussetzung für die Akzeptanz eines Systems. Korrektes Verhalten muss immer gewährleistet sein. Im Falle eines Fehlers muss auch der Übergang in einen sicheren Zustand gewährleistet sein. Aufgrund der Zuverlässigkeit des Systems sollte es nur selten in einen fehlerhaften Zustand geraten. Falls dies doch geschieht, kann das Wartungspersonal durch Maßnahmen wie den Neustart des Dienstes das System wieder in den korrekten Zustand versetzen.

Erweitbarkeit

Das System sollte so weit wie möglich auf der Ebene des Softwarecodes erweiterbar bleiben. Diese Erweiterbarkeit basiert nicht nur auf der bestehenden Funktionalität, sondern umfasst auch Verbesserungen in Bezug auf Systemzuverlässigkeit, Sicherheit und Bequemlichkeit.

Sicherheit

Obwohl dieses System hauptsächlich intern von der Industrie genutzt wird, darf die Sicherheit des Systems nicht außer Acht gelassen werden. Bei diesem System sollte die Datensicherheit gewährleistet sein. Die Benutzer sollten nur ihre eigenen Testfälle und Testumgebungsinformationen pflegen und verwalten können, und die Konfiguration von Testschritten, Testumgebungselementen und Benutzerkonten sollte nur vom Wartungs-personal geändert werden können. Wenn die Benutzer das Testverfahren und die Testumgebung für jeden beliebig ändern könnten, hätte dies unvorhergesehene Folgen.

3.3 Zusammenfassung

In diesem Kapitel werden Szenarien mit unterschiedlichen Merkmalen entwickelt und vorgestellt, die durch das hier vorgestellte System ermöglicht werden.

Für den Benutzer (Testingenieur) werden die grundlegenden Funktionspunkte für die Nutzung des Systems definiert, einschließlich der Verwaltung von Testfällen, der Verwaltung der Testumgebung, der Ausführung von Tests, der Anzeige von Ergebnissen usw. Andererseits kann der Wartungspersonal (Entwickler) die Art der für die Testfälle erforderlichen Testschritte und Testumgebungselemente aktualisieren, um sie auf dem neuesten Stand zu halten.

Ein intuitives, leicht zu bedienendes System ist für Benutzer und Wartungspersonal von entscheidender Bedeutung. Eine eigenständige Anwendung ist für dieses System möglicherweise nicht geeignet, wobei Faktoren wie die Aktualisierung von Testschritten und die Wiederverwendung von Testfällen und -umgebungen zu berücksichtigen sind, was zu Problemen führt, wie z. B. nicht rechtzeitig aktualisierte Testschritte und die Notwendigkeit, Testfälle und Testumgebung nach einem Computerwechsel neu auszufüllen oder zu importieren. Daher ist hier eine zentralisierte serverbasierte und webbasierte Implementierung des Systems angebracht.

4 Entwurf und Architektur

Im vorangegangenen Kapitel wurden die Anforderungen an das System auf der Grundlage des Szenarios entwickelt und spezifiziert. Auf der Grundlage dieser Anforderungen wird in diesem Kapitel das System mit Hilfe von Software-Engineering-Methoden entworfen.

4.1 System Überblick

Der zu entwickelnde Entwurf besteht aus einem Webserver, einem Testausführungs-modul und Testschritten (Bibliotheken), die die Benutzeroberfläche mit dem zu testenden System über eine automatisch generierte Testsuite verbinden. Die nachstehende

Abbildung 4.1 zeigt die Bestandteile des Systems.

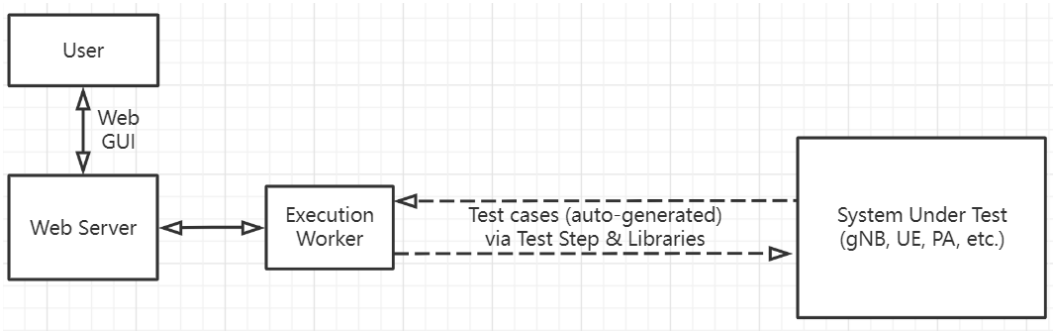

Abbildung 4.1: System Überblick

4.2 Datenarchitektur

Die folgenden Einheiten sind in der Datenarchitektur des Systems enthalten.

Benutzer

Der Benutzer repräsentiert den Testingenieur und kann Testfälle verwalten und bearbeiten, die Testumgebung konfigurieren, Testfälle ausführen (Testset), Testergebnisse anzeigen usw.

Testumgebungselement

Die Grundeinheit, aus der eine Testumgebung besteht und die den Typ des Testobjekts oder des automatisierten Testwerkzeugs darstellt, das beim Testen verwendet wird. Das Testumgebungselement enthält eine Liste von Parametern für die oben genannten Objekte. Testumgebungselement kann vom Wartungspersonal erstellt, geändert oder gelöscht werden. Benutzer kann Testumgebungselement zu Testumgebung hinzufügen oder ihn aus der Testumgebung entfernen.

Testumgebung

Zeigt eine Testumgebungskonfiguration an, die aus einem oder mehreren Testumgebungselementen und entsprechenden Werten besteht. Die Testumgebung gehört dem Benutzer und kann von ihm erstellt, geändert und gelöscht werden.

Testschritt

Testschritte sind die Grundeinheiten von Testfällen. Im System wird der Testschrittname verwendet, um das entsprechende Keyword in der Testschrittbibliothek zu finden. Ein Testschritt enthält eine Liste von Parametern, die für die entsprechenden Schlüsselwörter

in der Testschrittbibliothek erforderlich sind. Testschritt kann vom Wartungspersonal erstellt, geändert oder gelöscht werden. Benutzer kann ein Testschritte zu Testfälle hinzufügen oder ihn aus dem Testfall entfernen.

Testfall

Testfälle sind mit dem Benutzer verbunden und können von ihm erstellt, geändert und gelöscht werden. Ein Testfall besteht aus einem oder mehreren Testschritten und den Werten der zugehörigen Parameter.

Testsatz

Testsatz bezeichnet eine Testsuite, die Testfälle und Testumgebungen für dieselben Benutzer enthält, und direkt zur Ausführung verwendet werden kann. Ein Testsatz gehört zu einem Benutzer und kann von diesem hinzugefügt, geändert oder gelöscht werden.

Test Log

Das TestLog ist eine Aufzeichnung der einzelnen Tests, die zu einem bestimmten Testsatz gehört. Im TestLog werden der Zeitpunkt des Tests, die Testergebnisse und die Logdatei (über einen Link) aufgezeichnet.

Die oben genannten Datenentitäten wurden zusammengetragen und gemäß der Definition des Entity-Relationship-Modells modelliert. Die folgende Abbildung 4.2 zeigt ein einfaches ERM-Diagramm, das sich aus der Zusammenstellung und Modellierung der Daten und Beziehungen des Systems ergibt.

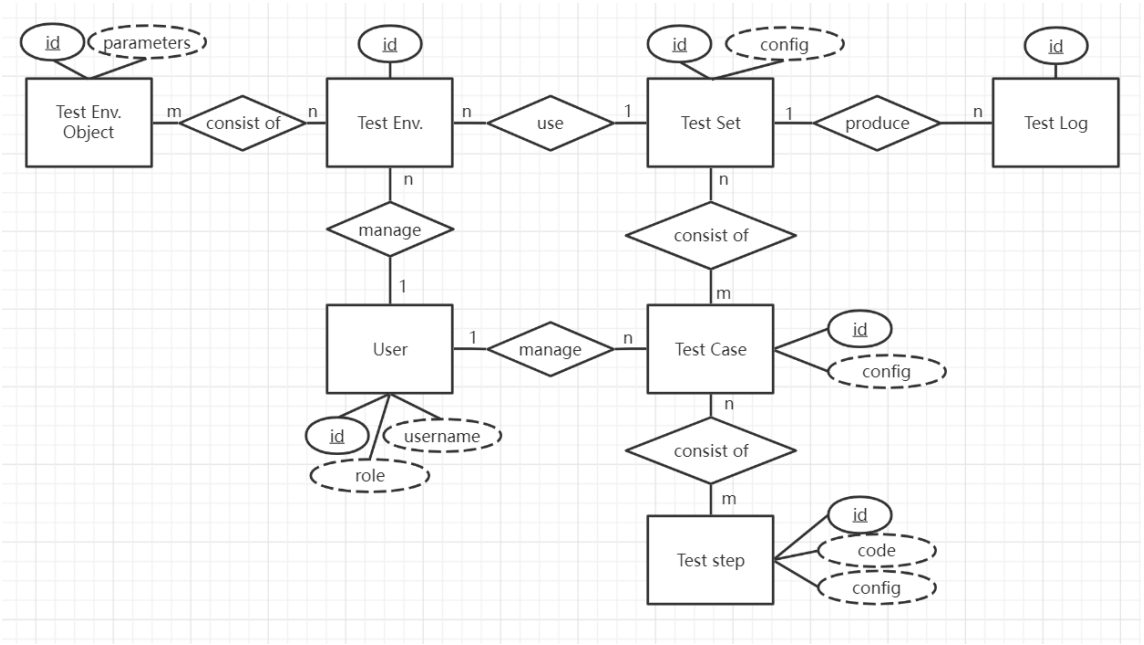

Abbildung 4.2: einfaches ERM Diagramm des Testfallgenerators

4.3 Model-View-Controller und Django MTV

Das Model-View-Controller-Konzept, kurz MVC, wurde erstmals von Trygve Reenskaug in den Jahren 1978-1979 vorgestellt. [Ree79] Der Zweck dieses Konzepts ist die Aufteilung einer interaktiven Anwendung in drei separate Teile: Modell, Sichten (View) und Controller. Modelle stellen die funktionalen Kernkomponenten und die Abbildung der verantwortlichen Objekte auf die Datenbank dar (ORM). Die Sicht stellt die Präsentation des Modells für den Benutzer dar, d. h. die grafische Oberfläche. Der Controller liegt zwischen den ersten beiden und ist für die Weiterleitung und Verarbeitung der Benutzereingaben zuständig.

Durch die Modellierung der oben genannten Entitäten wie Testfälle, Testschritte und Testumgebungen können darauf aufbauend die entsprechenden Sichten und Controller entwickelt werden. Mit Hilfe der Views und Controller können die grafischen Oberflächen und die Businesslogik der oben genannten funktionalen Szenarien implementiert werden, um die entsprechenden funktionalen Anforderungen zu erfüllen.

In Django ist das MVC-Konzept in drei Teilen implementiert - Model, Template, View - es kann als "MTV" bezeichnet werden. Das Template ist für die Darstellung der HTML-Seite für den Benutzer verantwortlich. Sie bietet eine Reihe von Vorlagensyntaxen, die es ermöglichen, Daten aus der Sicht an die Webschnittstelle zu übergeben. Views sind Python Callback-Funktionen für URLs, die Benutzereingaben (d.h. HTTP-Anfragen) entgegennehmen und verarbeiten. Gegebenenfalls können Views auf Modelle und Templates zugreifen, um Daten abzurufen/zu ändern und Seiten als Ausgabe an den Nutzer zu senden. Insgesamt unterscheidet sich das MTV-Konzept vom Namen her von MVC, aber es ist dennoch ein Konzept, das mit dem MVC übereinstimmt. Abbildung 4.3 zeigt eine einfache Darstellung des Django MTV-Komponentendiagramms. [Dja22a]

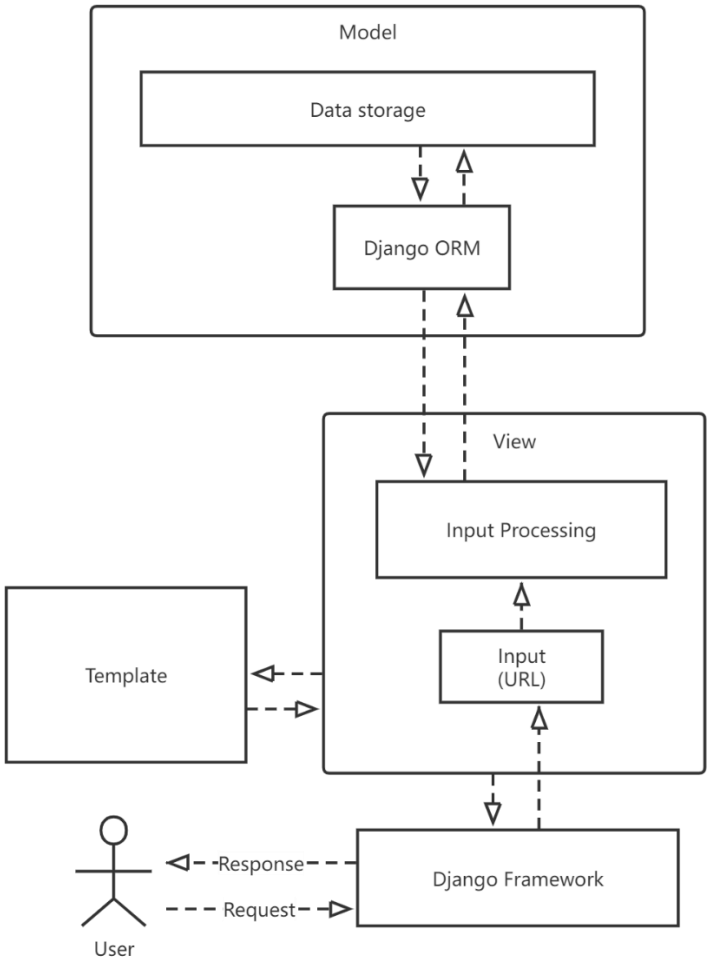

Abbildung 4.3: Komponentenübersicht der Django MTV

4.3.1 Model

Das Modell enthält hauptsächlich die Daten jedes in der Datenarchitektur genannten Entitätstyps sowie die Programmlogik zu deren Abfrage und Bearbeitung. Das bedeutet, dass Aktionen wie das Hinzufügen von Testschritt zu einem Testfall, das Ändern der Reihenfolge von Testschritt, das Bearbeiten der Testumgebung usw. auf der Modellebene implementiert werden. Daher muss die Modellebene verschiedene Schnittstellen zu den Views und Templates bereitstellen, damit die Benutzer schließlich auf diese definierten Funktionen zugreifen können.

Neue Dateneinträge werden im Modell durch Vorgänge wie das Hinzufügen von Testfällen oder Testumgebungen erstellt. Ebenso werden die entsprechenden Einträge im Modell aktualisiert, nachdem sie vom Benutzer bearbeitet wurden. Das System muss die entsprechenden Benutzeroperationen in Datenänderungen im Modell umsetzen. Darüber hinaus müssen bei der Ausführung der bestehenden Kombination aus Testfall

und Testumgebung (d.h. Testsatz) der Status des Laufs und der Ergebnisstatus ebenfalls im Modell gespeichert werden, um dem Benutzer angezeigt zu werden.

In dieser Arbeit werden die Entitäten durch die von Django bereitgestellte Klasse models.Model modelliert. Diese modellierten Klassen werden in einer Datenbank unter Verwendung der von Django bereitgestellten Datenbankschnittstelle gespeichert. Die oben genannten Dateneinheiten werden in modellierten Klassen und Datenbank Tabelle wie folgt dargestellt. Die Datenbank verwendet MariaDB (kompatibel mit MySQL).

User

In dieser Arbeit wird die mit Django mitgelieferte Klasse django.contrib.auth.models.User zur Darstellung der Benutzerentität verwendet. Die Datentabelle User enthält eine ID, einen Benutzernamen und ein verschlüsseltes Passwort. Die Klasse django.contrib.auth.models.User enthält auch andere Eigenschaften wie email, is_staff, usw., die hier weggelassen werden. [Dja22c]

Attribut	Typ	Länge	Idend.
id	INT	11	ja
username	VARCHAR	150	
password	VARCHAR	128	

Tabelle 4.1: Modell User

TestStep

Die Testschritt enthält den Namen des Schlüsselworts, seinen Katalog, eine Liste seiner Parameter (durch Kommas getrennt) und einen Beschreibungstext. Der Name des Testschritts ist sein Identifikator.

Attribut	Typ	Länge	Idend.
keyword_name	VARCHAR	191	ja
category	VARCHAR	191	
parameters	LONGTEXT		
description	LONGTEXT		

Tabelle 4.2: Modell TestStep

TestCase

Die Tabelle TestCase stellt die Testfälle dar. Die Datentabelle der Testfälle enthält die ID, die Benutzer-ID, den Namen des Testfalls, die Konfiguration, die Testschritte, die Einrichtungs- und Abrüstungs-testschritte.

Um das Lesen und Schreiben auf der Anwendungsschicht zu vereinfachen, wird die

Beziehung zwischen Testfällen und Testschritten nicht streng nach den Standards relationaler Datenbanken modelliert. Die Felder test_steps, setup_steps und teardown_steps des Testfalls speichern die Testschritte und ihre Parameterwerte als JSON-Format.

Attribut	Typ	Länge	Idend.
id	BIGINT	20	ja
user_id	INT	11	
case_name	VARCHAR	191	
testcase_config	LONGTEXT		
test_steps	LONGTEXT		
setup_steps	LONGTEXT		
teardown_steps	LONGTEXT		

Tabelle 4.3: Modell TestCase

EditTestCase

EditTestCase ist eine temporäre Dateneinheit, die erzeugt wird, wenn der Benutzer einen Testfall erstellt oder bearbeitet. Der Unterschied zwischen einem EditTestCase und einem TestCase besteht in der Hinzufügung der TestCase-ID, zu der er gehört. TestCase, der dem EditTestCase entspricht, wird erst aktualisiert, wenn der Benutzer auf Speichern klickt.

Attribut	Typ	Länge	Idend.
id	BIGINT	20	ja
user_id	INT	11	
from_case_id	BIGINT	20	
case_name	VARCHAR	191	
testcase_config	LONGTEXT		
test_steps	LONGTEXT		
setup_steps	LONGTEXT		
teardown_steps	LONGTEXT		

Tabelle 4.4: Modell EditTestCase

TestEnvironmentElement

Die Datentabelle für das Testumgebungseement enthält die ID, den Namen und eine Liste von Parametern (durch Kommas getrennt)

Attribut	Typ	Länge	Idend.
id	BIGINT	20	ja
Element_name	VARCHAR	191	
parameters	LONGTEXT		

Tabelle 4.5: Modell TestEnvironmentElement

TestEnvironment

Die Datentabelle für die Testumgebung enthält die ID, den zugehörigen Benutzer-ID, ihren Namen und die Elemente der Testumgebung. Um das Lesen und Schreiben auf der Anwendungsschicht zu vereinfachen, wird die Beziehung zwischen Testumgebung und Testumgebungselement nicht streng nach den Standards relationaler Datenbanken modelliert. Die Felder elements des TestEnvironments speichern die Testumgebungs-elemte und ihre Parameterwerte als JSON-Format.

Attribut	Typ	Länge	Idend.
id	BIGINT	20	ja
user_id	INT	11	
test_env_name	VARCHAR	191	
elements	LONGTEXT		

Tabelle 4.6: Modell TestEnvironment

EditTestEnvironment

Ähnlich wie EditTestCase ist EditTestEnvironment eine temporäre Dateneinheit, die erstellt wird, wenn der Benutzer die Testumgebung erstellt oder bearbeitet. Die Entität wird gespeichert, sobald der Benutzer eine Operation durchführt, wie z. B. das Hinzufügen eines Testumgebungselements, das Aktualisieren eines Elementparameters usw.

EditTestEnvironment fügt die TestEnvironment-ID hinzu, zu der es gehört. Die entsprechende Testumgebung wird erst aktualisiert, wenn der Benutzer auf Speichern klickt.

Attribut	Typ	Länge	Idend.
id	BIGINT	20	ja
user_id	INT	11	
from_env	BIGINT	20	
test_env_name	VARCHAR	191	
elements	LONGTEXT		

Tabelle 4.6: Modell TestEnvironment

TestSet

Die Datentabelle für den Testsatz enthält die ID, den Namen, die Konfiguration, die Testumgebung und die Testfälle, denen er zugeordnet ist.

Attribut	Typ	Länge	Idend.
id	BIGINT	20	ja
test_set_name	VARCHAR	191	
test_set_config	LONGTEXT		
test_env_id	BIGINT	20	
testcase_id	BIGINT	20	

Tabelle 4.6: Modell TestSet

TestLog

Die Datentabelle TestLog enthält die Aufzeichnungen der einzelnen Testdurchführungen. Sie enthält die ID, die ID des zugehörigen Testsatzes, die Start-/Endzeit, den Status und das Ergebnis (PASS oder FAIL).

Attribut	Typ	Länge	Idend.
id	BIGINT	20	ja
testset_id	BIGINT	20	
start_time	DATETIME		
End_time	DATETIME		
status	VARCHAR	16	
verdict	VARCHAR	16	

Tabelle 4.6: Modell TestLog

4.3.2 View / Django Template

Die Sicht liefert dem Benutzer Informationen über das Modell. Dazu ruft die Sicht die Daten des entsprechenden Objekts über eine Schnittstelle ab und zeigt sie an. Jede Sicht verfügt über eine Kontrollkomponente, die Benutzereingaben entgegennimmt und interpretiert. Die Sicht muss außerdem eine einfach zu bedienende, intuitive Schnittstelle bieten, die dem Benutzer die Interaktion mit dem System erleichtert.

Im Szenario dieser Arbeit wird die Oberfläche der Sicht durch das Web Template bereitgestellt. Django bietet eine Template-Engine, die es dem Controller (view.py)

ermöglicht, Daten im Kontext an das Template zu übergeben. Die Web-Templates präsentieren dem Nutzer die Daten wie gewünscht. [Dja22d]

Auf der Grundlage der zuvor beschriebenen benutzerbezogenen Anwendungsszenarien wurde eine Reihe von Templates erstellt, die für die Darstellung der View verwendet werden. Die folgende Abbildung 4.4 zeigt alle Webvorlagen in den Templates und die entsprechenden Sichten für jede Vorlage.

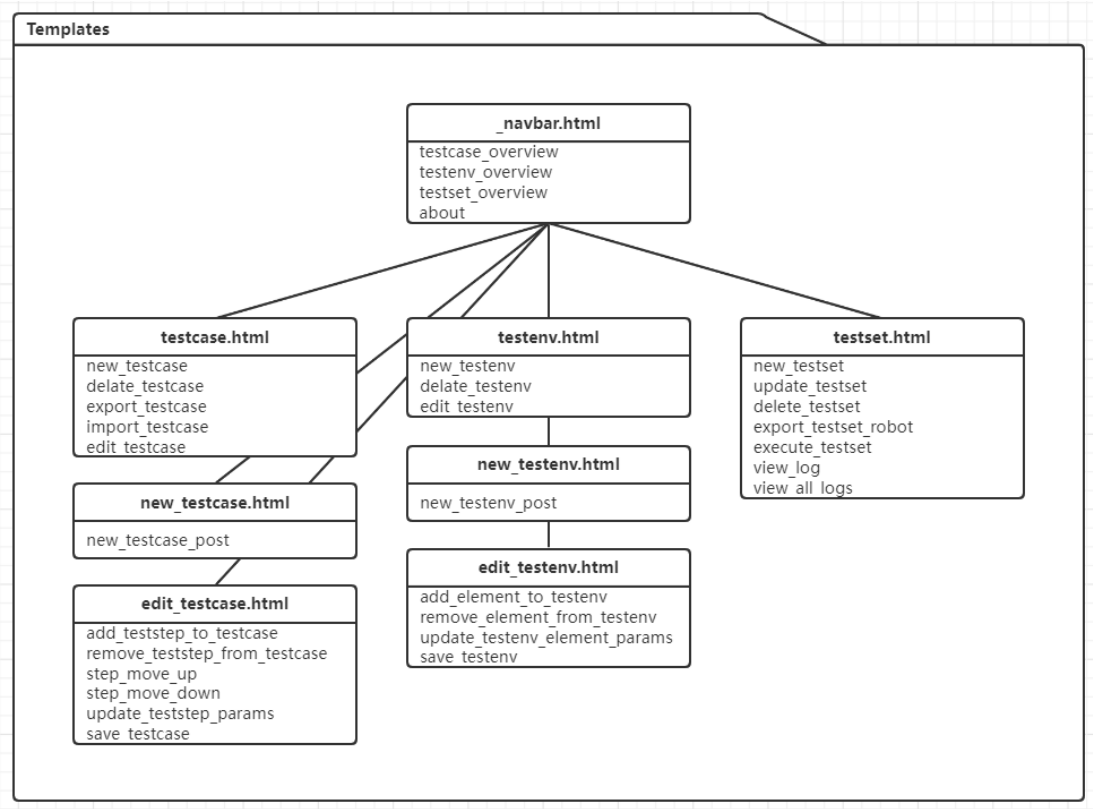

Abbildung 4.4: Templates von Testcase Manager

4.3.3 Controller / Django Views

Der Controller nimmt Benutzerinteraktionen entgegen, prüft und verarbeitet sie und kann eine oder mehrere Oberflächen verwalten.

In Django wird die Controller-Funktion hauptsächlich von view.py übernommen. Obwohl der Name anders lautet, enthält die django-View-Schicht die Logik für den Zugriff auf das Modell und die Auswahl der entsprechenden Vorlage. Durch die Zuordnung der zu beantwortenden URLs zu den entsprechenden Funktionen kann das System die Beantwortung und entsprechende Verarbeitung von Benutzeroperationen realisieren. Die Benutzeraktion wird konkret durch eine HTTP GET/PUT/POST-Anfrage dargestellt.

Die Anfrage wird erstellt, nachdem er auf einen Link oder eine Schaltfläche auf einer Webseite geklickt hat. [Dja22e]

Wenn beispielsweise ein Benutzer sich anmeldet, gibt er einen Benutzernamen und ein Passwort ein, klickt dann auf Anmelden und sendet eine POST-Anfrage. Die in der View angegebene Callback-Funktion holt sich den Benutzernamen und das Passwort der Anfrage und prüft die Gültigkeit des Benutzers und des Passworts über die Model-Schnittstelle. Nach der Bestätigung, dass das Kennwort des Benutzers gültig ist, bezieht View über das Modell eine Liste von Testfällen für den Benutzer und gibt die Übersichtsseite für Testfälle an den Benutzer zurück. Die Seite enthält das HTML-Template testcases.html und den Kontext mit Liste von Testfällen für die Testfallübersicht, die schließlich als Webseite im Browser des Benutzers dargestellt wird.

5 Realisierung

Auf der Grundlage des Entwurfs im vorherigen Kapitel liegt der Schwerpunkt nun auf der Implementierung und Ausführung einer auf Python und Django basierenden Webanwendung. Die Webanwendung kann Tests für das Robot-Framework generieren und ausführen. Die Anwendung ermöglicht es den Benutzern, Testfälle und Testumgebungen zu verwalten, Test auszuführen und die Ergebnisse der Läufe anzuzeigen. Außerdem ist es erforderlich, dass das Wartungspersonal die Testschritte mit den entsprechenden Bibliotheken und Elementen der Testumgebung aktualisieren kann.

Die Anwendung basiert auf Python Version 3.7 und Django Version 3.2. Die Testfälle werden auf Basis von Robot-Framework Version 3.2.2 erzeugt. Das Projekt wird unter dem Betriebssystem Windows 10/11 entwickelt und debuggt. Die entwickelten Anwendungen können auch auf Servern mit Linux-Systemen eingesetzt werden. In dieser Arbeit wurde die implementierte Anwendung auf einem Server mit Ubuntu 20.04 eingesetzt und in einem Webbrowser demonstriert.

5.1 Implementierung

Die Projektstruktur des implementierten Systems ist in Abbildung 5.1 dargestellt und enthält Konfigurationsdateien, Anwendungspakete, Webvorlagen, statische

Webressourcen, Dateiexportordner und Testausführungsordner.

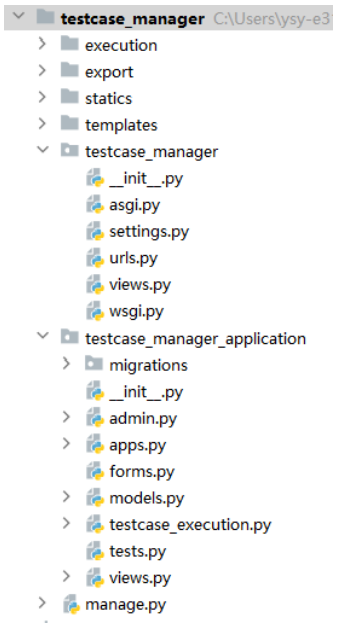

Abbildung 5.1: Projektstruktur Testcase Manager

Das Anwendungspaket testcase_manager_application enthält den Hauptlogikcode, einschließlich der Modellklassen und Views. Darüber hinaus wird der für die Ausführung der Testsatz erforderliche Logikcode in das Modul testcase_execution.

Das Verzeichnis testcase_manager ist der Ort, an dem die Konfiguration des Django-Frameworks abgelegt wird. Die in diesem Projekt verwendete Konfiguration besteht aus zwei Hauptabschnitten. Zuerst wird setting.py für die globale Konfiguration verwendet, einschließlich statischer Web-Dateipfade, Anwendungspaketeinführungen und Datenbankverbindungen. Zweitens wird urls.py verwendet, um URLs und die dazu-gehörigen View-Callback-Funktionen zu spezifizieren, d.h. um allen möglichen Benutzer-aktionen im System Views zuzuordnen.

Das Verzeichnis Templates enthält alle verfügbaren Webvorlagendateien, die im vorherigen Entwurf festgelegt und definiert wurden. Der Ordner statics enthält statische Ressourcendateien, einschließlich Webicons, JavaScript-Skripte, CSS-Dateien, Font-Dateien, usw. Der Ordner export ist das temporäre Verzeichnis, in das die Benutzer die Zip-Datei beim Exportieren von Testsatz verpacken werden. Im Ordner execution werden die Testsatz-Laufdateien und die von ihnen erzeugten Logs gespeichert.

5.1.1 Modellklassen

Die implementierte Modell-Klasse basiert auf der Klasse django.db.models.Model. Auf der Grundlage des vorherigen Entwurfs wurden acht neue Modellklassen erstellt, zusätzlich zu der mit Django gelieferten Klasse User.

Neben den Eigenschaften der Modellklassen selbst umfassen sie auch Kernfunktionen wie das Hinzufügen von Testschritten zu Testfällen, das Ändern von Elementparametern in der Testumgebung usw. Die folgende Abbildung 5.1 zeigt das Klassendiagramm (mit Modul pyreverse automatisch generiert).

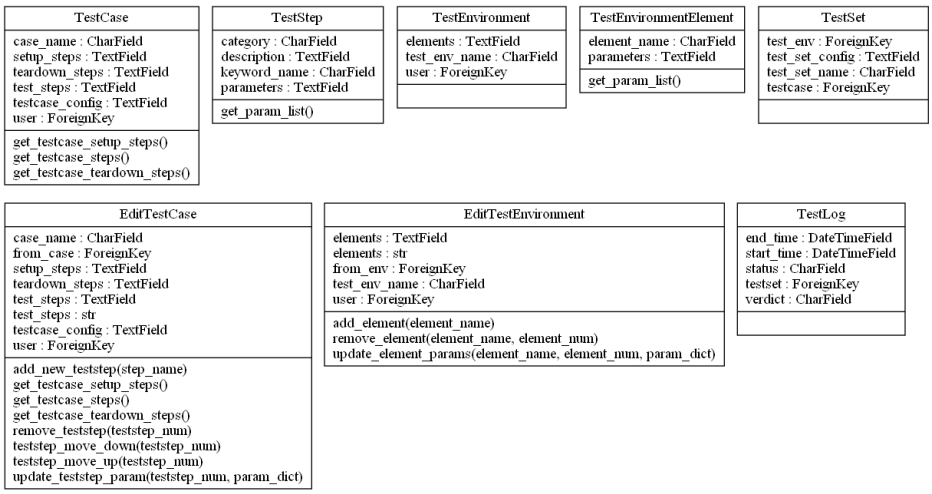

Abbildung 5.1: Klassendiagramm Implementierte Modellklassen

5.1.2 Realisierung von Szenarien

Login

Die Anmeldeseite wird von der View-Funktion user_login verwaltet und wird Benutzern angezeigt, die sich noch nicht angemeldet haben. user_login prüft den vom Benutzer eingegebenen Benutzernamen und das Passwort und akzeptiert die Anmeldung des Benutzers oder lehnt sie ab.

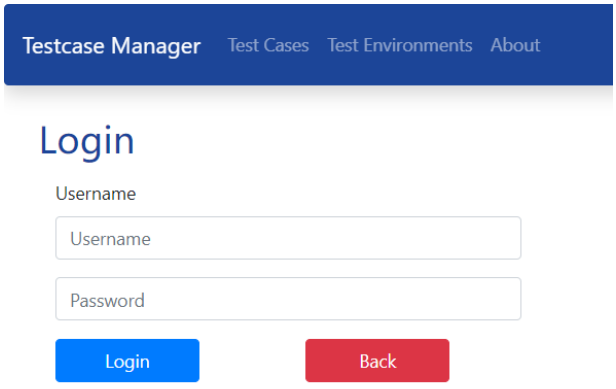

Abbildung 5.2: Login Seite

Verwaltung, Hinzufügen und Löschen von Testfällen

Wenn ein eingeloggter Benutzer die Webseite öffnet, zeigt das System standardmäßig die Testfallübersichtsseite an. Die Seite Testfallübersicht wird von Funktion testcase_overview und Template testcase.html verwaltet. Abbildung 5.3 unten zeigt die Testfälle, die dem Benutzer admin gehören. Auch das Erstellen, Bearbeiten, Importieren, Exportieren und Löschen von Testfällen sind hier implementiert.

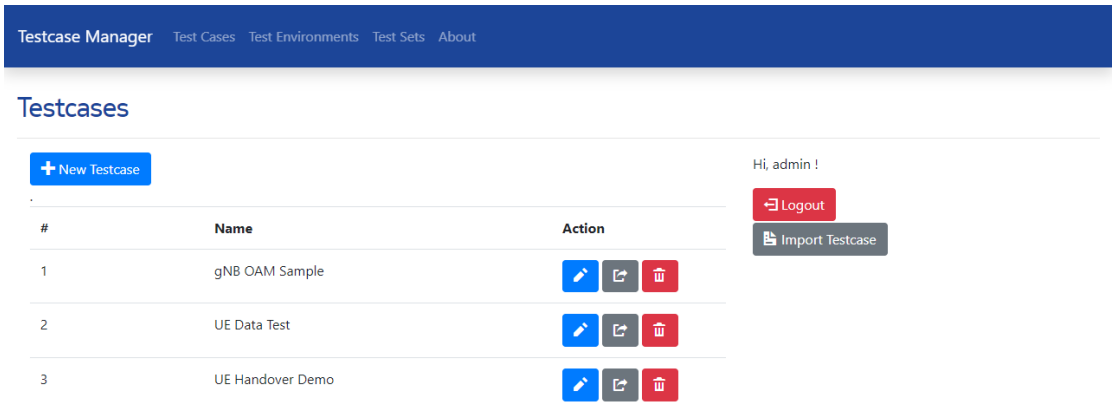

Abbildung 5.3: Testfallübersichtsseite

Oben in testcase.html befindet sich die Schaltfläche "Neuer Testfall", die auf views. new_testcase verweist und eine new_testcase.html zurückgibt, in die der Benutzer den Namen des neuen Testfalls eingeben und dann wählen kann, ob er zur Bearbeitung oder zum Abbruch übergehen möchte.

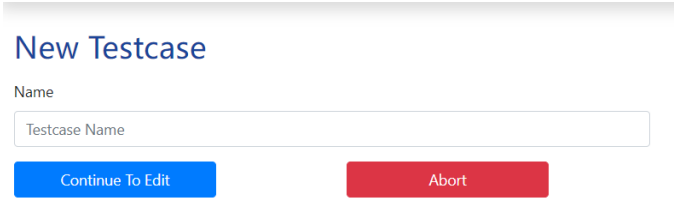

Abbildung 5.4: Neue Testfall

Testfall bearbeiten

Die Option Bearbeiten auf dem Testfall führt zur Bearbeitungsseite des entsprechenden Testfalls. Die Views-Funktion views.edit_testcase erzeugt ein temporäres Objekt EditTestCase, das den aktuellen Benutzer und den (tatsächlichen) Testfall enthält. Hier

werden das Hinzufügen/Entfernen von Testschritten, das Verändern der Reihenfolge von Testschritten, das Aktualisieren der Parameter von Testschritten und das Speichern von Testfällen implementiert, wie in folgende Abbildung 5.5 gezeigt.

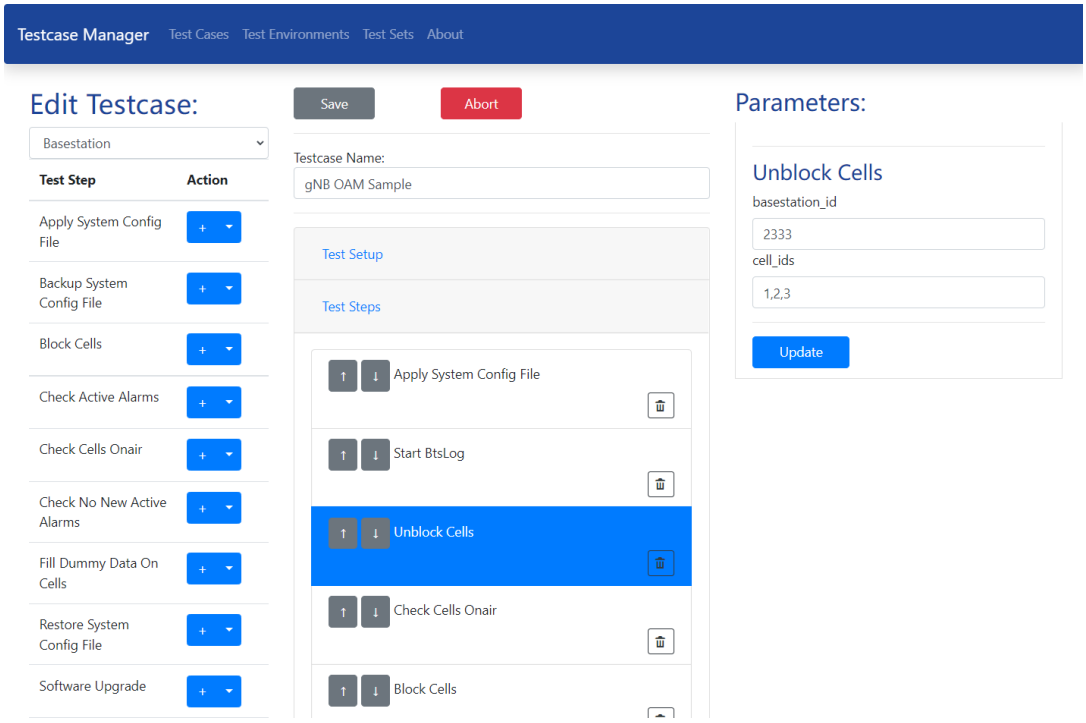

Abbildung 5.5: Testfallbearbeitung

Die Testschrittauswahl bietet eine Liste aller verfügbaren Testschritte. Der Benutzer wählt oben im Auswahlfeld eine Kategorie aus und die Testschritte dieser Kategorie werden unten angezeigt. Diese Auswahl- und Anzeigeaktion wird durch eine JavaScript-Funktion innerhalb der Template implementiert. Die Pfeile auf der Schaltfläche Hinzufügen des Testschritts ermöglichen es dem Benutzer, den Testschritt zum Setup/Teardown des Testfalls hinzuzufügen. In der Spalte Testschritt kann der Benutzer einzelnen Testschritt innerhalb des Testfalls auswählen. Die Javascript-Funktion in der Template hebt den ausgewählten Eintrag hervor und zeigt alle Parameter des Testschritts im Parameter-Editor auf der rechten Seite an.

Verwaltung der Testumgebung

Die Verwaltungsseite für die Testumgebung wird durch die View-Funktion testenv_overview und die Template testenv.html bereitgestellt. Die Seite enthält alle Testumgebungen des aktuellen Benutzers. Hier werden die Neu-, Bearbeitungs- und Löschfunktionen der Testumgebung implementiert, wie in Abbildung 5.6 unten gezeigt.

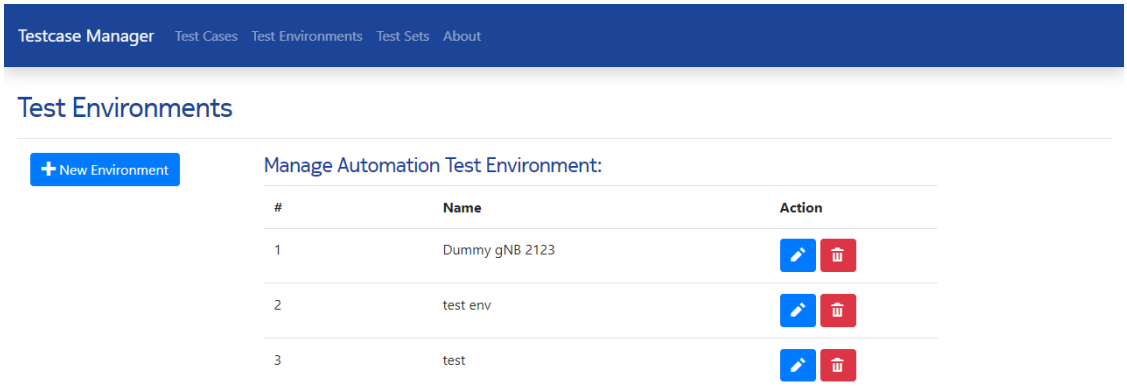

Abbildung 5.6: Testumgebungen verwalten

Die Seite Neue Testumgebung ist ähnlich aufgebaut wie die Seite Neuer Testfall. Wie in Abbildung 5.7 unten dargestellt

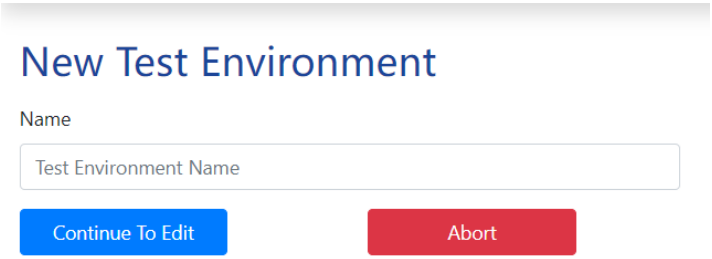

Abbildung 5.7: Neue Testumgebung

Testumgebung bearbeiten

Die Bearbeitungsseite für die Testumgebung wird durch die View-Funktion edit_testcase und eine gleichnamige Template bereitgestellt. Hier werden die Funktionen des Hinzufügens/Entfernens von Testumgebungselementen, des Änderns von Testumgebungs-elementparametern und des Änderns von Testumgebungsnamen implementiert. Die Bearbeitungsseite für die Testumgebung ist in Abbildung 5.8 wie folgt dargestellt.

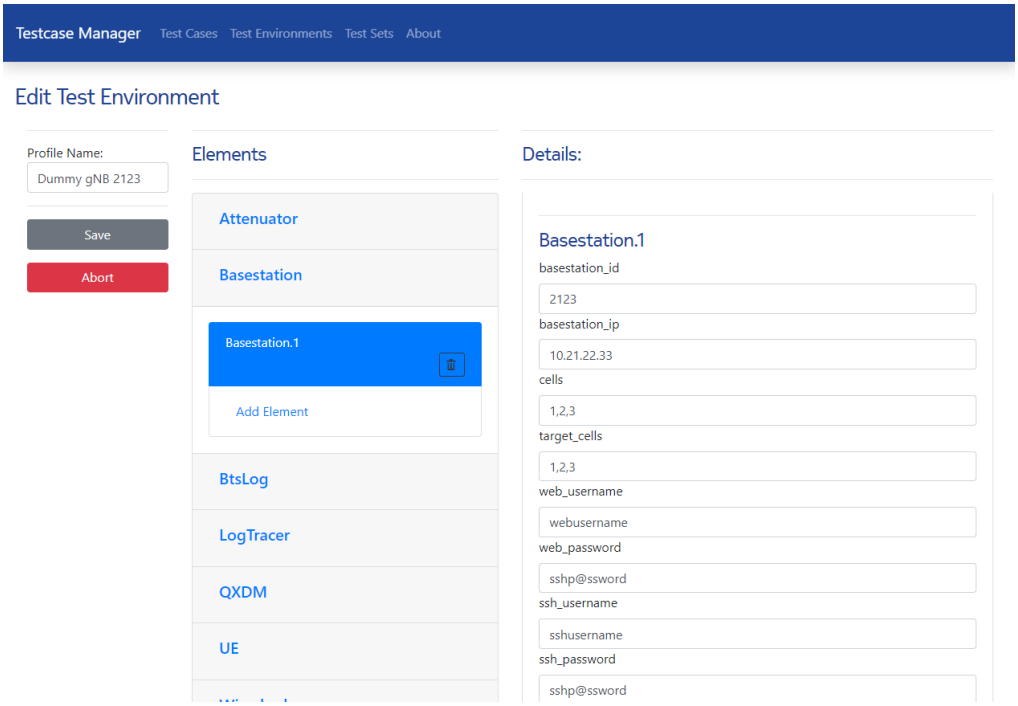

Abbildung 5.8 Testumgebungsverwaltung

Anders als auf der Seite zur Bearbeitung von Testfällen wird das Hinzufügen/Entfernen von Testumgebungselementen unterhalb des Elementtyps angezeigt. Benutzer können das entsprechende Elementkarte von SUT/Testautomatisierungsumgebung erweitern, um die zur Testumgebung hinzugefügten Elemente zu verwalten.

Übersicht der Testsets

Die Seite Testsätze wird durch die View-Funktion testset_overview und die Template testset.html bereitgestellt. Die Seite zeigt alle Testsätze des aktuellen Benutzers und seine letzten 3 Ausführungen an. Hier sind die Funktionen zum Hinzufügen, Ändern, Löschen und Ausführen von Testsätzen sowie der entsprechende Export und die Anzeige von Ausführungsprotokollen implementiert. Die implementierte Seite ist in Abbildung 5.9 wie folgt dargestellt

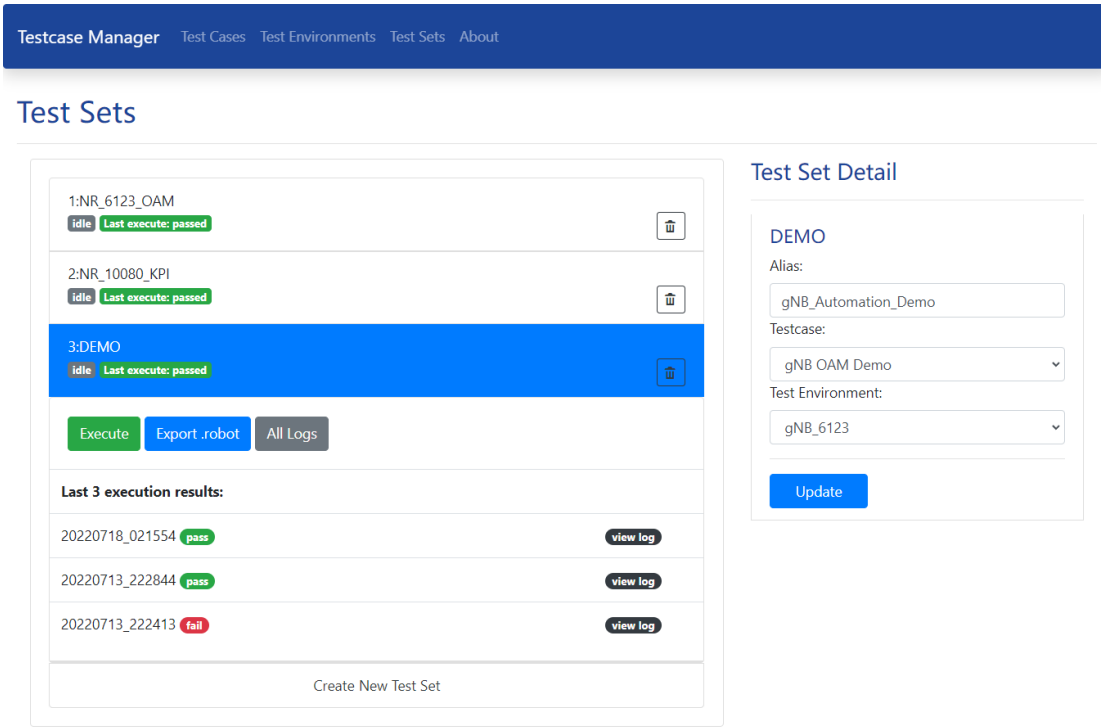

Abbildung 5.9: Testsätze verwalten

Die View-Funktion testset_overview prüft anhand des Status der Testlogeinträge, ob der Testsatz derzeit ausgeführt wird. Falls ein Test durchgeführt wird, wird die entsprechende Testsatzkarte in Abbildung 5.10 wie folgt dargestellt.

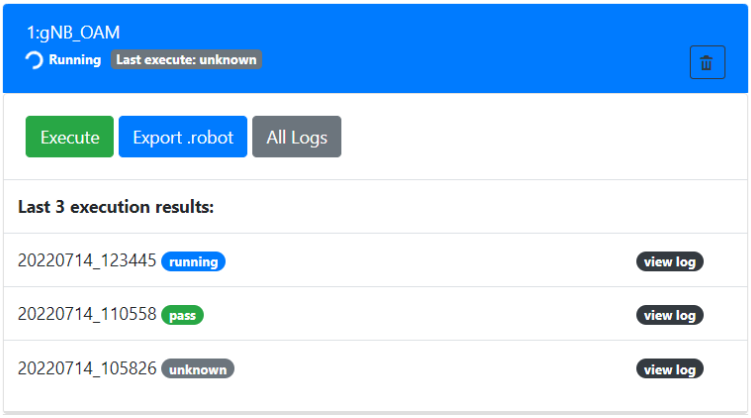

Abbildung 5.10: Laufende Testsatz

Durch Anklicken der Option „All Logs“ auf der Testsatzkarte kann der Benutzer die Seite mit allen AusführungsLogs für diesen Testsatz anzeigen. Hier sind die Funktionen zur Anzeige von Start und Ende der Ausführung, der Ergebnisse und des Log-Links implementiert. Diese Seite wird durch die View-Funktion view_all_logs und die Template

log_list.html bereitgestellt, wie in folgende Abbildung 5.11 dargestellt.

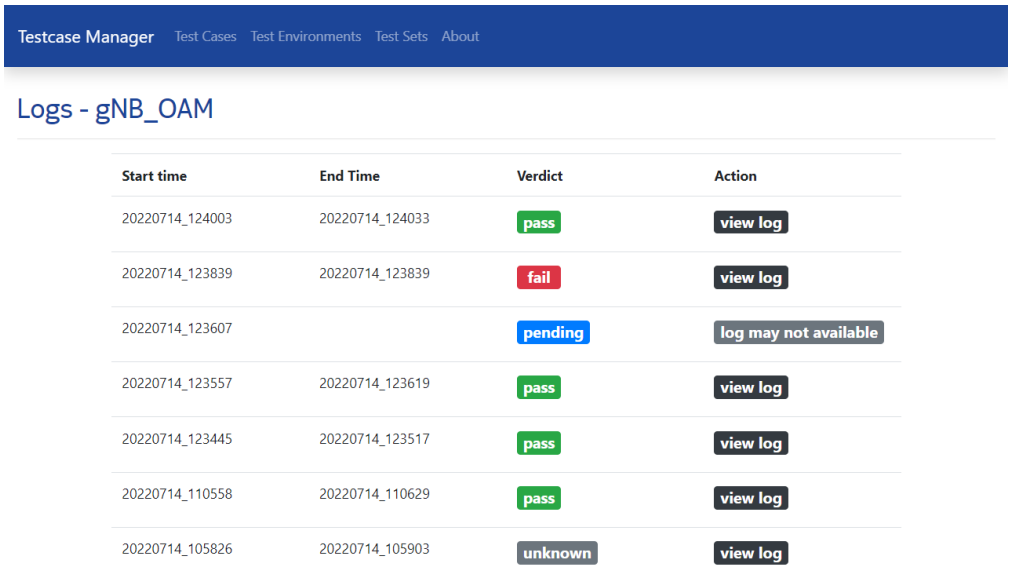

Abbildung 5.11: Alle Ausführungsprotokolle für den Testsatz.

Abbildung 5.12 unten zeigt die Logs, die automatisch von Robot-Framework während der Testausführung generiert werden. Im Log sind die entsprechenden Testschritte auf der Seite zur Bearbeitung des Testfalls (Abb. 5.5) zu sehen. Zur Demonstration dieser Testschritte wird der Dummycode verwendet.

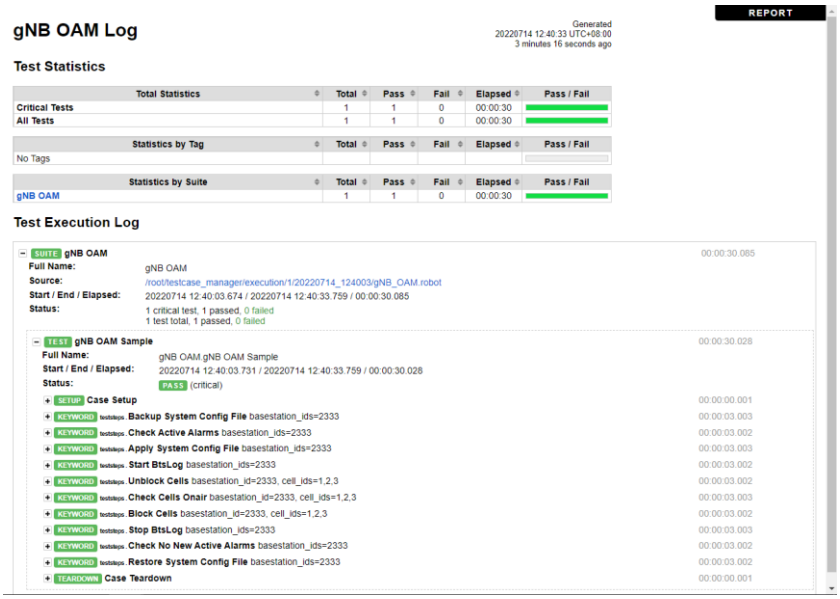

Abbildung 5.12: Ausführungsprotokoll der automatisch generierten Testsuite.

Verwaltung von Benutzern, Testschritten und Testumgebungselementen

Im Gegensatz zu den benutzerbezogenen Funktionen ist die Verwaltung von Benutzern, Testschritten und Testumgebungselementen in Django Admin implementiert, das mit Django geliefert wird. Django Admin liest Metadaten aus Modellen, um eine schnelle, modellzentrierte Oberfläche bereitzustellen, über die vertrauenswürdige Benutzer die Inhalte Ihrer Website verwalten können. [Dja22b]

Durch die Verwaltung von Benutzerkonten in Django Admin können Administratoren Datenberechtigungen für einzelne Benutzerkonten festlegen. Dieses Datenrecht ist nur in Django Admin wirksam. So können Administratoren festlegen, wer Testschritte und Testumgebungen verwalten oder normale Benutzer hinzufügen darf. Benutzer mit den entsprechenden Rechten sind das Wartungspersonal und können die entsprechenden Daten im Modell aktualisieren. Folgende Abbildung 5.13 zeigt ein Beispiel für die Bearbeitung eines Testschritts in Django Admin.

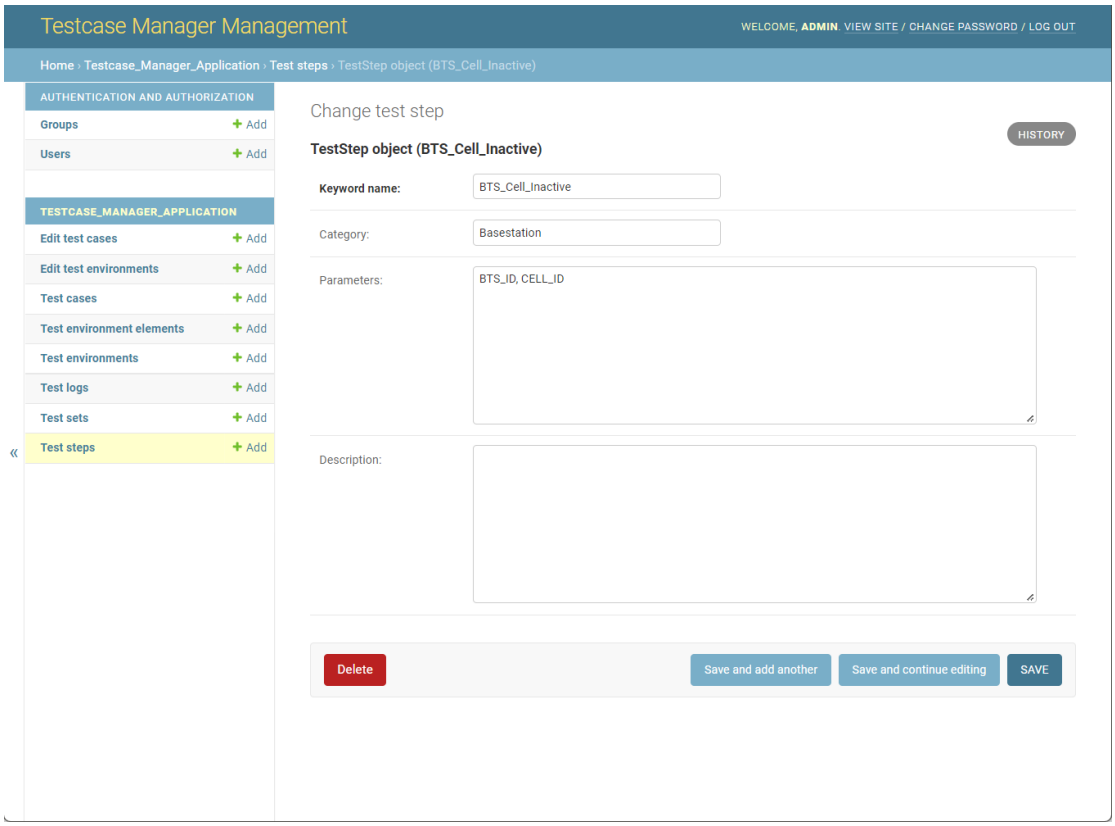

Abbildung 5.13: Bearbeitung eines Testschritts in Django Admin

5.2 Evaluation

In diesem Abschnitt werden die vom System implementierten Funktionen evaluiert. Dazu werden die in den vorangegangenen funktionalen Anforderungen beschriebenen Szenarien mit dem implementierten System verglichen. Außerdem wurde die Umsetzung der nichtfunktionalen Anforderungen im Hinblick auf die implementierten Systeme bewertet.

5.2.1 Abdeckung der Anforderungen

Im Folgenden wird die Umsetzung der funktionalen und nicht-funktionalen Anforderungen überprüft. Es ist zu beachten, dass einige Anforderungen im implementierten System nicht vollständig erfüllt werden.

Funktionale Anforderungen

- Benutzeranmeldung:
  Diese Anforderung wurde erfüllt. Benutzer kann sich mit ihren Anmeldedaten am System anmelden, um die Funktionen des Systems zu nutzen.

- Testfälle ansehen:
  Diese Anforderung wurde erfüllt. Benutzer kann seine Testfälle ansehen

- Testfall erstellen
  Diese Anforderung wurde erfüllt. Benutzer kann neue Testfälle erstellen.

- Testfall löschen
  Diese Anforderung wurde erfüllt. Benutzer kann Testfälle löschen, und es gibt einen Dialog zur Bestätigung des Löschens.

- Testfall importieren
  Diese Anforderung wurde erfüllt. Benutzer kann Dateien hochladen und Testfälle importieren, wobei das Dateiformat überprüft wird.

- Testfall exportieren
  Diese Anforderung ist teilweise erfüllt, da der Benutzer das Verzeichnis, in dem die exportierten Testfälle gespeichert werden sollen, noch nicht auswählen konnte.

- Testfall bearbeiten

  Diese Anforderung wurde erfüllt. Benutzer kann Testfälle bearbeiten, einschließlich des Hinzufügens und Löschens von Testschritten, des Änderns ihrer Reihenfolge und des Änderns ihrer Parameter.

- Testumgebung ansehen

  Diese Anforderung wurde erfüllt. Benutzer kann seine Testumgebung ansehen

- Testumgebung erstellen

  Diese Anforderung wurde erfüllt. Benutzer kann neue Testumgebung erstellen.

- Testumgebung löschen

  Diese Anforderung wurde erfüllt. Benutzer kann Testumgebung löschen, und es gibt einen Dialog zur Bestätigung des Löschens.

- Testumgebung bearbeiten

  Diese Anforderung wurde erfüllt. Benutzer kann Testumgebung bearbeiten, einschließlich des Hinzufügens und Löschens von Elementen und des Änderns ihrer Parameter.

- Testsatz ansehen

  Diese Anforderung ist teilweise erfüllt. Der Ausführungsstatus des Testsatzes wird nur zu dem Zeitpunkt ermittelt, zu dem die Seite aktualisiert wird, und wird nicht in Echtzeit aktualisiert. Es gibt auch keine geeignete Möglichkeit, den Ausführungs-fortschritt des aktuellen Testsatzes im Roboter-Framework abzurufen.

- Testsatz erstellen

  Diese Anforderung wurde erfüllt. Benutzer kann neue Testsatz erstellen.

- Testsatz bearbeiten

  Diese Anforderung wurde erfüllt. Benutzer kann Testsatz bearbeiten, bei denen Testfälle und Testumgebungen zugewiesen werden.

- Testsatz ausführen

  Diese Anforderung wurde erfüllt. Benutzer kannTestsatz ausführen.

- Testsatz als ausführbare Testsuiten exportieren

  Diese Anforderung wurde erfüllt. Benutzer kann Testsatz als .zip Datei exportieren.

- Ergebnisse der Testdurchführung ansehen

  Diese Anforderung ist teilweise erfüllt. Benutzer kann alle Ausführungen des Testsatzes einsehen, einschließlich Laufzeiten und Ergebnisse. Die Logs der Testdurchführung enthalten jedoch nur die automatisch von Robot-Framework generierten Logs, andere Logs, die von den SUT/Testwerkzeugen generiert und erfasst werden, sind noch nicht im System verfügbar.

- Testschritte verwalten

  Diese Anforderung wurde erfüllt. Wartungspersonal kann auf Django Admin Testschritte hinzufügen, löschen oder ändern.

- Testumgebunselemente verwalten

  Diese Anforderung wurde erfüllt. Wartungspersonal kann auf Django Admin Testumgebunselemente hinzufügen, löschen oder ändern.

- Benutzern verwalten

  Diese Anforderung wurde erfüllt. Wartungspersonal kann auf Django Admin Benutzerkonten hinzufügen, löschen oder ändern.

Nicht-Funktionale Anforderungen

Zunächst ist zu ermitteln, ob das System die Anforderungen an die Zuverlässigkeit erfüllt. Zur Zeit sind keine schwerwiegenden Fehlerzustände im System festgestellt worden. Bei der Erstellung von Testfällen werden die Parameter in den Testfällen jedoch nicht überprüft. Dies führt möglicherweise dazu, dass die resultierenden Testfälle fehlschlagen. Auch die Namen von Testsätzen / Testfällen / Testumgebungen erlauben die ungeprüfte Eingabe beliebiger Zeichen, was zu unvorhergesehenen Situationen führen kann, z. B. zu Fehlern bei der Verarbeitung von Dateipfaden.

In Bezug auf die Erweiterbarkeit bietet das MVC- oder MTV-Pattern die Möglichkeit, in Zukunft weitere Funktionen hinzuzufügen und zu verbessern. Die Funktionalität kann durch die Änderung von Modellen, Sichten oder Templates leicht erweitert werden.

In Bezug auf die Sicherheit stellen das Anmeldesystem und die von Django bereitgestellten Funktionen grundsätzlich sicher, dass Benutzer nur ihre eigenen Testfälle, Testumgebungen und Testsets bearbeiten können. Darüber hinaus können nur Benutzer mit bestimmten Berechtigungen Testschritte, Testumgebungselemente und Benutzerkonten verwalten. Die Sicherheit kann jedoch noch verbessert werden, z. B. durch einen reinen HTTPS-Modus die Verbindungssicherheit verstärken.

6 Zusammenfassung

6.1 Auswertung der Ergebnisse

Die Evaluierungsergebnisse zeigen, dass es grundsätzlich möglich ist, einen Testfallgenerator für RAN-Test zu erstellen. Die demonstrierten Umsetzungsergebnisse können dies bestätigen. Für die Umsetzung dieser Arbeit wurde das auf Python basierende Django-Framework verwendet. Das MVC/MTV-Pattern von Django vereinfacht die Projektentwicklung erheblich. Zusätzlich bietet es auch ein gewisses Maß an Skalierbarkeit und Flexibilität. In Kapitel 3 wird hauptsächlich eine Anforderungsanalyse durchgeführt, um die möglichen Szenarien aufzulisten. Auf dieser Grundlage wurden in Kapitel 4 Modelle und Sichten (Templates) entworfen. Deshalb ist es intuitiv, die Unterschiede zwischen Implementierung und Anforderungen/Entwurf vor und nach dem Implementierungsprozess zu vergleichen.

Außerdem können die einzelnen Testschritte der Testfälle selbst aufgrund der Unterschiede zwischen manuellen und automatisierten Testmethoden für die Tester noch schwer verständlich sein. Ausgehend von diesem System können die Entwickler der TA-Team später abstraktere Testschritte anbieten, um das Verständnis weiter zu vereinfachen.

Das bestehende System bedarf jedoch noch zahlreicher Verbesserungen, um mehr benutzerfreundlich, sicher und robust zu sein. Der Code des implementierten Systems entspricht so weit wie möglich den Grundsätzen der Wiederverwendbarkeit, der geringen Kopplung und der hohen Lesbarkeit. Allerdings lassen sich im Front-End (Web-Template) noch einige Tricks anwenden, um die Benutzererfahrung weiter zu verbessern. Auf der Oberfläche zur Bearbeitung von Testfällen kann nach besseren Möglichkeiten gesucht werden, um Testschritte hinzuzufügen und die Reihenfolge der Testschritte zu ändern, wie z.B. Drag&Drop.

6.2 Ausblick

In Anbetracht der Benutzerfreundlichkeit für das Wartungspersonal (Entwickler) kann in späteren Iterationen des Systems eine Art automatische Aktualisierung der Testschritte implementiert werden. Bspw. durch die Verknüpfung von Code-Repositories und die Auslösung automatischer Updates auf der Grundlage von Code-Commits.

Künftige Versionen des Systems sollten auch die Möglichkeit bieten, die Ausführung von Testschritten in Testfällen zu schleifen, zu verzögern und zu parallelisieren. Es kann auch in Betracht gezogen werden, die Ausführung mehrerer Testfälle in einem einzigen Testsatz zuzulassen. Außerdem können einige zusätzliche Optionen für die Ausführung der Testsuite verwendet werden, z. B. Schleifenzyklen, Dryrun und andere Parameter usw.

Was die Ausführung der Tests angeht, genügt es nicht, die gesamte Testausführung auf einem einzigen Server durchführen zu lassen, wenn das System eine breitere Anwendung erreichen wird. In zukünftigen Versionen des Systems kann die Integration einer verteilten Architektur in Betracht gezogen werden. Die eigentliche Ausführung der Tests kann anderen Servern im Netz übertragen werden.

Es bleibt abzuwarten, inwieweit das in dieser Arbeit implementierte System die Arbeitsbelastung von Testern und Entwicklern von Testautomatisierungen verringern wird. Durch kontinuierliche Entwicklung, Integration, Iteration und Einholung von Nutzerfeedback werden ausreichende Benutzerfreundlichkeit, Stabilität und Sicherheit erreicht. Dadurch können Testingenieure genügend robuste Testfälle schreiben, auch ohne das mit einem automatischen Testframework verbundene Fachwissen.

Ähnliche Systeme können über den Bereich der RAN-Automatisierungstests hinaus eingesetzt werden. Das in dieser Arbeit vorgeschlagene System kann auch in anderen Branchen eingesetzt werden, die einen hohen Grad an Testautomatisierung erfordern.

7 Literaturverzeichnis

[3GP22] 3GPP, 3GPP Specification series: 25 series, 3GPP: https://www.3gpp.org/DynaReport/25-series.htm, 2022, Stand 18.05.2022.

[3GP221] 3GPP, 3GPP Specification series: 36 series, 3GPP: https://www.3gpp.org/DynaReport/36-series.htm, 2022, Stand 18.05.2022.

[5GP16] 5G PPP Architecture Working Group, "View on 5G Architecture", 5GPPP: http://doi.org/10.5281/zenodo.3265031, Version 3.0, 2020.

[5GN19] 5G Networks, 5G Terminology: The gNB, Online im Internet: https://www.5g-networks.net/5g-technology/5g-terminology-the-gnb/, 2019, Stand 16.05.2022.

[Che20] Kai Chen, Anforderungen und Herausforderungen beim automatisierten (Chinesisch), Online im Internet: http://www.cww.net.cn/article?id=475469, 2020, Stand 10.05.2022.

[Dja22] Django Software Foundation, Design philosophies | Django Documentation, Online im Internet: https://docs.djangoproject.com/en/3.2/misc/design-philosophies/, 2022, Stand 20.06.2022.

[Dja22a] Django Software Fundation, Django FAQ | Django Documentation, Online im Internet: https://docs.djangoproject.com/en/4.0/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names, 2022, Stand 20.06.2022.

[Dja22b] Django Software Fundation. (2022, Juli) Django Admin| Django Documentation, Online im Internet: https://docs.djangoproject.com/zh-hans/3.2/ref/contrib/admin/, 2022, Stand 06.07.2022.

[Dja22c] Django Software Fundation, django.contrib.auth | Django Documentation, Online im Internet: https://docs.djangoproject.com/en/3.2/ref/contrib/auth/ , 2022, Stand 04.07.2022.

[Dja22d] Django Software Fundation, Templates | Django Documentation, Online im Internet: https://docs.djangoproject.com/en/3.2/topics/templates/, 2022,

Stand 03.07.2022.

[Dja22e] Django Software Fundation, Writing views | Django Documentation, Online im Internet: https://docs.djangoproject.com/en/3.2/topics/http/views/, 2022, Stand 03.07.2022.

[Gan16] Mary Gannon, What are attenuators? Online im Internet : https://www.connectortips.com/faq-what-are-attenuators/, 2016, Stand 17.05.2022.

[Goo22] Google, Android Debug Bridge (adb) - Android Developers. Online im Internet :. https://developer.android.com/studio/command-line/adb, 2022, Stand 24.05.2022.

[HBT22] HBTE, Programmable RF Attenuator. Online im Internet : https://www.hbtetech.com/products/programmable-rf-attenuator.html, 2022, Stand 17.05.2022.

[Jon21] Dan Jones und Corinne Bernstein, What is a radio access network?, Online im Internet : https://www.techtarget.com/searchnetworking/definition/radio-access-network-RAN, 2021, Stand 08.05.2022.

[Kot98] Gerald Kotonya, *Requirements Engineering: Processes and Techniques*.: Chichester. https://www.amazon.de/-/en/Gerald-Kotonya/dp/0471972088 , 1. Auflage, 1998.

[Lau06] Pekka Laukkanen, "Data-Driven and Keyword-Driven Test", Helsinki University of Technology, Masterarbeit: http://eliga.fi/Thesis-Pekka-Laukkanen.pdf, 2006.

[Pet20] Larry Peterson und Oguz Sunay, *5G Mobile Networks: A Systems Approach*.: Morgan & Claypool Publishers, https://www.amazon.com/5G-Mobile-Networks-Approach-Synthesis/dp/1681738880 , 3. Auflag, 2020.

[Ree79] Trygve Reenskaug, "MODELS - VIEWS - CONTROLLERS", https://folk.universitetetioslo.no/trygver/1979/mvc-2/1979-12-MVC.pdf , 1979.

[Rob22] Robot Framework Foundation, Robot Framework. Online im Internet: https://robotframework.org/, 2022, Stand 26.05.2022.

[Ste05] Andrew Stellman und Jennifer Greene, *Applied Software Project*

Management.: O'Reilly Media, 2005. https://www.amazon.de/-/en/Andrew-Stellman/dp/0596009488 , 1. Auflage, 2005.

[Wik02] Wikipedia, Django_(web_framework) - Wikipedia, Online im Internet : https://en.wikipedia.org/wiki/Django_(web_framework), 2022, Stand 03.07.2022.

Selbstständigkeitserklärung gem. § 14 Absatz 5 BPO

Hiermit versichere ich, ...Shiyi Ying....., dass ich die vorliegende Bachelorarbeit mit dem Titel

Testfallgenerator für die Automatisierungstest von 4G/5G-Funkzugangsnetz

selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Zwickau, den 22.07.2022

..... Ying Shiyi .....

.....

Ort, Datum Unterschrift