terrestris GmbH & Co. KG

Sitz u. Registergericht: Bonn
Amtsgericht Bonn HRA 6835

Steuer-Nr.: 206/5809/0848
Ust.-ID-Nr.: DE223943541

Komplementärin:
terrestris Verwaltungsgesellschaft mbH

vertreten durch:
Torsten Brassat, Marc Jansen,
Hinrich Paulsen, Till Adams

Gesamtsystementwurf GDI-DE Testsuite

Version: 1.0

Technisches Feinkonzept für die Neuentwicklung der GDI-DE Testsuite

(B14.10-3031/17 / 18.062.bdbdi_gdide-testsuite)

erarbeitet für:

Bundesamt für Kartographie und Geodäsie

z. H. Herrn Gerd Weber

Richard-Strauss-Allee 11
60598 Frankfurt am Main

erarbeitet von:

terrestris GmbH & Co. KG

Kölnstraße 99
53111 Bonn

Telefon: +49 228 - 962 899 51
Telefax: +49 228 - 962 899 57

E-Mail: info@terrestris.de
Internet: www.terrestris.de

Ansprechpartner:
Steffi Forberig forberig@terrestris.de

Projektbezeichnung	Neuentwicklung GDI-DE Testsuite	
Projektleiter	Steffi Forberig	
Verantwortlich	Projektleiter	
Erstellt am	02.05.2019	
Zuletzt geändert	11.07.2019 14:57	
Bearbeitungszustand		in Bearbeitung
		vorgelegt
	x	fertig gestellt
Dokumentablage	C:\Users\weber_g\Nextcloud\Testsuite 3\Systemspezifikation\20190711_Gesamtsystementwurf_ohne_Layout_MS2.odt	
V-Modell-XT Version	2.2	

Weitere Produktinformationen

Mitwirkend	[x] SW-Architekt [x] Systemarchitekt [x] Anforderungsanalytiker [x] Informationssicherheitsverantwortlicher [x] Betriebsverantwortlicher [x] Ergonomieverantwortlicher
Erzeugung	Erstellung des Gesamtsystementwurfs

Änderungsverzeichnis

Änderungen			Geänderte	Beschreibung der		
Nr.	**Datum**	**Version**	**Kapitel**	**Änderungen**	**Autor**	**Zustand**
1	23.05.2019	0.2	alle	Ergänzungen	Verena Diewald	in Bearbeitung
2	24.05.2019	0.2	alle	Ergänzungen	Steffi Forberig	in Bearbeitung
3	27.-29.05.2019	0.2	alle	Ergänzungen	Verena Diewald	fertig gestellt
4	17.-25.06.2019	0.3	alle	Ergänzungen	Verena Diewald	fertig gestellt
5	01.07.2019	0.4	Einstiegssei	Titel, Änderungsverzeichnis,	Verena	fertig gestellt

			ten	Prüfverzeichnis	Diewald	
6	11.07.2019	1.0	Deckblatt	Finale Version MS2	Verena Diewald	fertig gestellt

Prüfverzeichnis

Die folgende Tabelle zeigt einen Überblick über alle Prüfungen – sowohl Eigenprüfungen wie auch Prüfungen durch eigenständige Qualitätssicherung – des vorliegenden Dokumentes.

Datum	Geprüfte Version	Anmerkungen	Prüfer	Neuer Produkt-zustand
23.05.2019	0.1		SG	vorgelegt

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die GDI-DE Testsuite ist eine der Nationalen Technischen Komponenten der Architektur der Geodateninfrastruktur Deutschland (GDI-DE) und dient als zentrale Testplattform der Qualitätssicherung innerhalb der GDI-DE.

Mit der GDI-DE Testsuite kann innerhalb der GDI-DE die Qualität für Geodaten und Geodatendienste geprüft werden. Mit Hilfe dieser zentralen Testplattform können Datenanbieter und Dienstebereitsteller ihre Geodaten und Dienste auf Konformität zu nationalen und internationalen Standards, z. B. den Vorgaben der europäischen INSPIRE-Richtlinie prüfen.

Die GDI-DE Testsuite ist seit Ende September 2011 frei nutzbar und unterstützt die Anbieter von Geodaten und Geodatendiensten bei der Bereitstellung ihrer Ressourcen innerhalb der GDI-DE und der Umsetzung der INSPIRE-Richtlinie. Neben einer Webanwendung (https://testsuite.gdi-de.org) und einer Download-Möglichkeit für die lokale Nutzung ist auch eine Schnittstelle für die Ausführung von Tests aus anderen Anwendungen heraus nutzbar.

Zurzeit stehen Tests für die Prüfung von Metadaten, Katalog-/Suchdiensten (CSW), Karten-/Darstellungsdiensten (WMS) und Downloaddiensten (WFS, Atom) bereit.

2 Ausgangssituation und Zielsetzung

Die GDI-DE Testsuite stellt eine der zentralen Komponenten der GDI-DE dar. Sie ist dabei ein wesentliches Werkzeug für die Qualitätssicherung von Geodaten und Geodatendiensten. Hinsichtlich der aktuellen Implementierung sind im Betrieb neue Anforderungen und Bedürfnisse, insbesondere aus Betriebs- aber auch aus Nutzersicht entstanden. Dazu gehört beispielsweise die Integration der Executable Test Suites aus dem INSPIRE Validator auf europäischer Ebene oder auch der Ersatz für das eingesetzte Java-Framework echo3, welches nicht mehr weiter gepflegt wird. Das Lenkungsgremium der GDI-DE hat beschlossen, dass die GDI-DE Testsuite durch eine Neuentwicklung ersetzt werden soll (Beschluss Nr. 97 1).

Ziel dieser Neuentwicklung ist es, ein neues, gleichwertiges System zu erstellen, welches den bisherigen Funktionsumfang (Anforderung 0) auf einer zukunftssicheren Technologiebasis bereitstellt. Darüber hinaus existieren seitens der GDI-DE Anforderungen, die durch die derzeitige Implementierung nicht erfüllt werden können. Diese Funktionalitäten sollen durch das zu entwickelnde System bereitgestellt werden. Ein weiterer Aspekt, der für die Neuentwicklung des GDI-DE Testsuite eine wesentliche Rolle spielt, ist die Herstellung einer erhöhten Ausfallsicherheit gegenüber dem bestehenden System.

Das Projekt zur Neuentwicklung der GDI-DE Testsuite verfolgt dabei die folgenden Projektziele:

- Verbesserung der Benutzerfreundlichkeit:

  Dieses Projektziel soll insbesondere durch die Einbindung der maßgeblichen Benutzergruppen in der Konzeptionsphase erreicht werden. Dazu tragen die Berücksichtigung des Responsive Design sowie die Umsetzung von intelligenten Assistenten in Form von Tool-

tips und die Ausgabe von aussagekräftigen Testberichten zur Erreichung dieses Projektzieles bei.

- Funktionale Erweiterbarkeit und horizontale Skalierbarkeit:

  Dieses Ziel wird vor allen Dingen durch die Einhaltung einer Microservices-Architektur erreicht. Charakteristische für Microservices-Architekturen ist die einfache funktionale Erweiterbarkeit. Ebenfalls beinhalten Microservices-Architekturen das Potenzial einer erhöhten Skalierbarkeit, da im Gegensatz zu anderen Architekturmustern eine horizontale Skalierung bereits auf Microservices-Ebene möglich ist und dadurch flexible Anpassungen möglich sind.

- Verbesserte Wartbarkeit:

  Im Rahmen des Projektes wird eine Verbesserung der Wartbarkeit im Vergleich zur aktuellen Implementierung angestrebt. Dies soll insbesondere durch die Etablierung eines durchgängigen und weitestgehend automatisierten Deployments erreicht werden. Darüber hinaus dienen verschiedene Maßnahmen im Entwicklungsprozess wie Testautomatisierung und Code Reviews der Verbesserung der Wartbarkeit des neuen Systems.

- Hochverfügbarkeit:

  Durch die Bereitstellung des neuen Systems innerhalb eines HA-Clusters soll erreicht werden, dass die neue GDI-DE Testsuite hochverfügbar bereitgestellt wird. Im Rahmen des Entwicklungsprozesses wird dieses Projektziel durch den Einsatz von Docker und Kubernetes, aber auch durch die entsprechende Auslegung der Architektur unterstützt.

Dieses Feinkonzept beschreibt, wie die Anforderungen des Lastenheftes für die Neuentwicklung der GDI-DE Testsuite realisiert werden sollen.

Es entspricht damit einem Pflichtenheft und ist das Produkt „Gesamtsystementwurf“, das im Projekthandbuch aufgeführt ist.

Die Anforderungen des Lastenheftes gliedern sich in funktionale und nichtfunktionale. Die zu realisierenden Anwendungsfälle müssen jedoch sowohl funktionalen als auch nichfunktionalen Anforderungen genügen. Es wurde deshalb bei dem vorliegenden Feinkonzept bewusst auf die Trennung in diese beiden Anforderungskategorien verzichtet, so dass von der üblichen Gliederungsstruktur abgewichen wird.

3 Allgemeine Systemeigenschaften

3.1 Software & Entwicklung

Alle eingesetzten Software-Komponenten besitzen eine Open Source Lizenz (Anforderung 73). Die von uns entwickelten Komponenten werden ebenfalls unter einer Open Source Lizenz zur Verfügung gestellt. Dabei favorisieren wir die Apache License, Version 2.01.

Um die Plattformunabhängigkeit gewährleisten zu können, setzen wir auf die Java-Plattform. Einzige Voraussetzung ist dabei ein Java Compiler (OpenJDK[1] Version 11 LTS), der für alle gängigen Betriebssysteme zur Verfügung steht. Gleiches gilt für die Oberfläche des Systems, welche über einen Webbrowser aufgerufen werden kann (Anforderung 74). Die Entwicklung der Client-Komponente setzt auf eine Node.js[2] Entwicklungsumgebung, die ebenfalls für alle gängigen Betriebssysteme zur Verfügung steht. Java und JavaScript sind seit Jahrzehnten etablierte Programmiersprachen im Web-Enterprise-Umfeld (Anforderung 76, 93). Für diese Programmiersprachen ist auch eine Empfehlung im SAGA-Modul Technische Spezifikationen 5.0.0 ausgesprochen.

Der Betrieb ist mit einem Standardwebserver möglich, indem dieser als Reverse Proxy (z. B. Apache, Nginx, HAProxy Ingress Controller, IIS) für unsere Java-basierten Schnittstellen (z. B. Apache Tomcat oder Jetty) fungiert (Anforderung 75). Eine genauere Spezifikation erfolgt im Betriebskonzept. Die HTTP-Schnittstellen operieren auf JSON bzw. XML und arbeiten mit UTF-8 Encoding. Alle Quellcodedateien, Datenbanken und Konfigurationsdateien werden ebenfalls in diesem Encoding abgelegt (Anforderung 80).

Der Quellcode wird mit sprachspezifischen Kommentaren versehen, auf deren Basis automatisch eine API-Dokumentation erzeugt wird. Für JavaScript wird hierbei die TSDoc-Spezifikation[3], für Java die JavaDoc-Spezifikation[4] benutzt (Anforderung 91). Es werden Klassenköpfe, Methodenköpfe, Funktionsparameter sowie Rückgabewerte kommentiert.

Alle Linter für Java- und JavaScript-Code werden für die verwendeten Module so konfiguriert, dass eine API-Dokumentation zwingend erforderlich ist. Für die Ermittlung der Abdeckungsmetriken (Coverage) wird SonarQube genutzt. Die Ergebnisse werden projektbegleitend gemäß Entwicklungsfortschritt mit dem Prüfprotokoll an den Auftraggeber übergeben (Anforderung 90). Weiter spezifiziert wird dies im Implementierungs-, Integrations- und Prüfkonzept sowie in der Prüfspezifikation Software.

3.2 Betrieb

Die Anforderungen

- 95: 20 Testausführungen gleichzeitig, darüber Fehlermeldung
- 96: 24/7 95% Betrieb, Designentscheidungen im Systementwurf begründen
- 97: Vorgaben GDI-DE Technik 3.4.0 werden eingehalten
- 100: Betrieb in High-Availability-Umgebung, Anforderungen in Systementwurf spezifizieren

werden im Betriebskonzept spezifiziert.

1 https://adoptopenjdk.net/
2 https://nodejs.org/en/about/
3 https://github.com/Microsoft/tsdoc/blob/master/spec/code-snippets/DeclarationReferences.ts
4 https://docs.oracle.com/javase/9/docs/specs/doc-comment-spec.html

4 Systemarchitektur

4.1 Übersicht

Die Systemarchitektur ist in Abbildung 1 schematisch dargestellt.

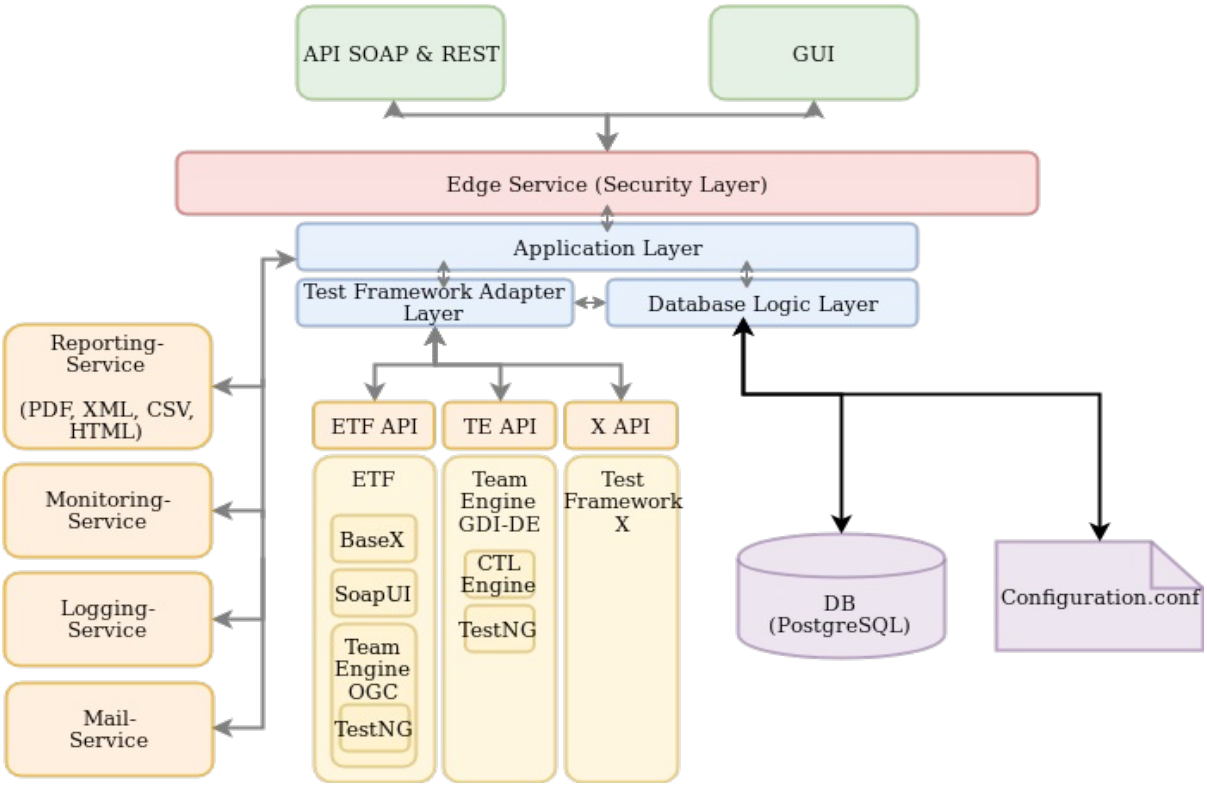

Abbildung 1: Layer-Architektur

In einem höheren Detailliertheitsgrad zeigt Abbildung 2, wie diese auf einzelnen Microservices beruht (Anforderung 89, 94). Ziel dieser Architektur ist die Bereitstellung der einzelnen Services (weiße und grüne Kästen in Abbildung 2) in einer High-Availability (HA) Clusterumgebung, in der die relevanten Services entsprechend der notwendigen Ressourcen skaliert werden können. Höchste Priorität der Architektur hat daher die Trennung inhaltlich und systematisch vereinbarer Services. Grün unterlegte Bestandteile stehen dabei für einzelne Docker-Container, die die entsprechenden Services beinhalten.

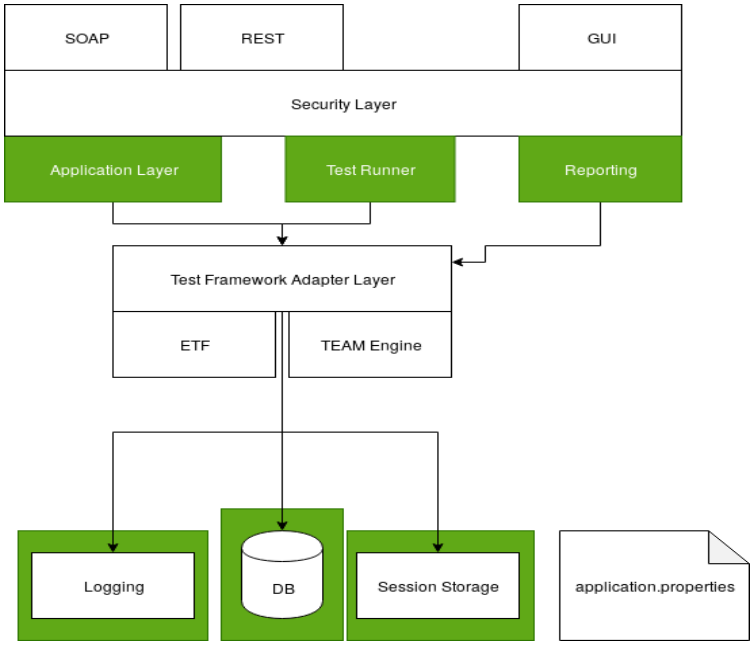

Abbildung 2: Service-Architektur

Die in der folgenden Abbildung 3 mit einem grünen Haken markierten Komponenten werden vollständig von Kubernetes bereitgestellt.

Die mit einem blauen Umring versehenen Komponenten stellen sich wie folgt dar:

- Der Reporting Service wird in der Anwendung als eigenständiger Microservice implementiert (linker blauer Umring). Er dient zur Erstellung und Auslieferung von Testberichten in den verschiedenen Formaten. In der Service-Architektur oben in Abbildung 2 findet sich dieser Service ganz rechts. Er bedient sich über den Test Framework Adapter Layer an den Testergebnissen.
- Der Test-Runner-Service hat die Aufgabe, Testläufe durchzuführen (mittlerer blauer Umring). In der Service-Architektur in Abbildung 2 findet sich dieser Service mittig dargestellt.
- Der rechte blaue Umring in der unteren Abbildung beinhaltet letztendlich zwei Services:
  - Der User Service ist für die Benutzerverwaltung zuständig.
  - Der Application Service kümmert sich um die restlichen Aufgaben der Anwendung (Login, Registrierung, Management der Testkonfigurationen).

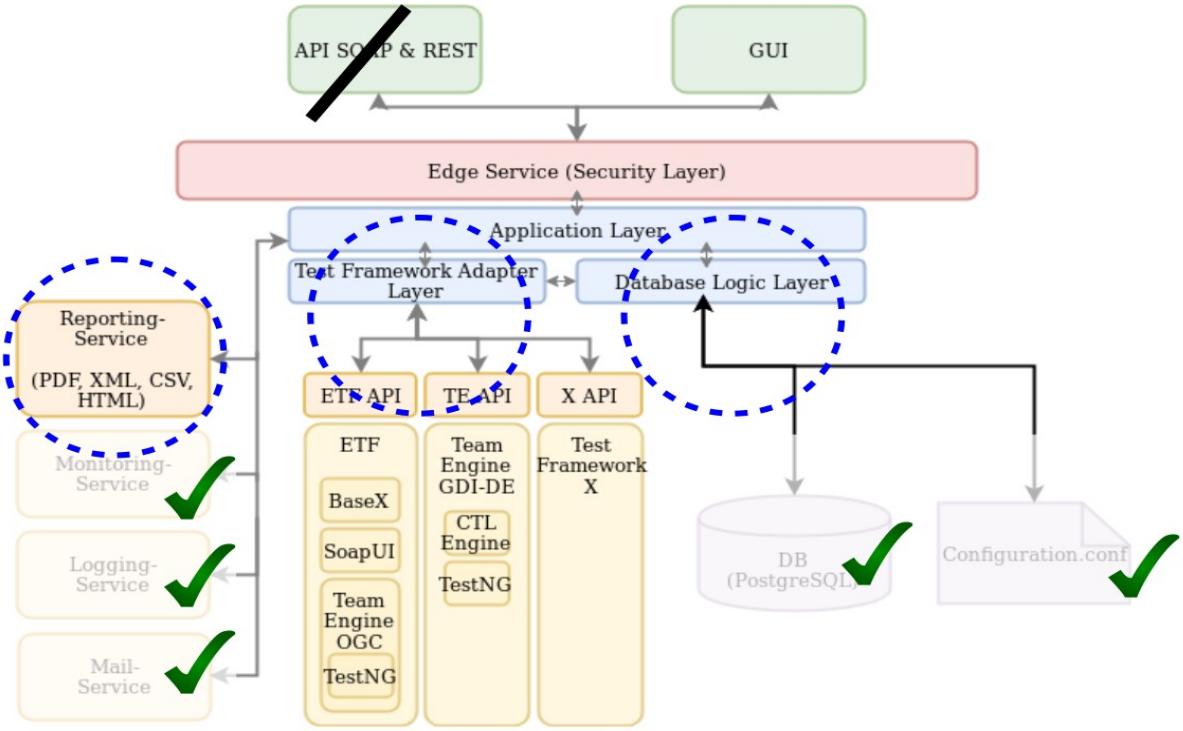

Abbildung 3: Identifizierung der Microservices

4.2 Datenbank

Die Datenhaltung erfolgt durch eine Redis-Session-Storage[5], die es den Komponenten ermöglicht, die Session über mehrere Container verteilt zu nutzen. Für die eigentliche Datenhaltung kommt ein PostgreSQL[6]-Datenbankcontainer zum Einsatz (Anforderung 65).

4.3 Konfiguration

Zentrale statische Konfigurationen können über eine application.properties Datei gesteuert werden und liegen zentral für alle Container erreichbar auf einem Persistent Volume (Anforderung 92).

Neben der application.properties werden an gleicher Stelle die Übersetzungsdateien für die Oberflächenkomponenten abgelegt. Die Übersetzungsdateien liegen im JSON-Dateiformat vor (Anforderung 68).

4.4 Application Layer & Database Logic Layer

Die einzelnen Komponenten der Anwendung werden in vier separate Microservices unterteilt. Der Runner-Service dient dazu, Tests auszuführen. Da dies einen Großteil der Ressourcen brauchen wird, wird diese in einem eigenen Container laufen, so dass sich dieser Teil gut auf die verfügbaren Nodes skalieren lässt. Der Reporting-Service dient zur Erzeugung von Testreports für den Download. Beim Erzeugen von vielen Reports kann dies ebenfalls Last erzeugen, so dass auch

5 https://redis.io/
6 https://www.postgresql.org/

diese Komponente in einem eigenen Container laufen wird und skaliert werden kann. Der User Service kümmert sich um die Benutzerverwaltung. Abschließend gibt es den Application-Service, der die restlichen Schnittstellen bereitstellt (Login, Registrierung, Management der Testkonfigurationen). Dieser kann selbstverständlich ebenfalls skaliert werden, sofern nötig (Anforderung 81). Jeder einzelne Knoten ist in der Lage, Dateien von 5 GB Größe zu verarbeiten (Anforderung 43).

Zusammengenommen bilden diese vier Microservices den im Angebot bzw. im Layer-Architekturmodell aufgeführten Application Layer. Der Runner-Service, der User Service, der Application-Service sowie der Reporting-, Monitoring und Logging-Service werden vom Application Layer angesprochen und gesteuert. Je nach Größe des Application-Services kann dieser in weitere Microservices aufgeteilt werden, sofern dies sinnvoll erscheint.

Der Database Logic Layer bildet die Verknüpfung zwischen dem Application Layer und der Datenbank. Dessen Aufgabe ist die objektrelationale Abbildung (ORM) von Datenbankeinträgen in Java-Objekte und umgekehrt. Dies wird über das Framework Hibernate[7] bewerkstelligt. Damit ist auch eine Datenbankunabhängigkeit gewährleistet, obwohl als Datenbank bereits PostgreSQL festgelegt ist.

4.5 Test Framework Adapter Layer

Es sind mindestens folgende Java-Module geplant, ggf. werden diese bei ausreichender Komplexität noch weiter aufgeteilt:

- Modul mit der Definition der Kern-Testkomponenten. Dort wird auch das Interface definiert, mittels dessen ggf. weitere Testframeworks via SPI (Service Provider Interface) integriert werden können (Anforderung 40, 99).
- Modul mit Implementierung des SPI zur Anbindung der ETF[8]
- Modul mit Implementierung des SPI zur Anbindung der TEAM-Engine[9]
- Spring Boot Modul zur Bereitstellung der Runner-Schnittstellen
- Spring Boot Modul zur Bereitstellung der Reporting-Schnittstellen

Die genaue Ausarbeitung des Test Framework Adapter Layer inklusive der technischen Umsetzung der Testklassenverwaltung sowie der entsprechenden Benutzeroberflächen folgt im Projektverlauf in einer eigenen Feinspezifikation.

7 https://hibernate.org/
8 https://github.com/etf-validator/etf-webapp
9 https://github.com/opengeospatial/teamengine

4.5.1 Datenmodell

Eine schematische Darstellung der Datenflüsse zwischen den Applikationskomponenten ist in Abbildung 4 dargestellt.

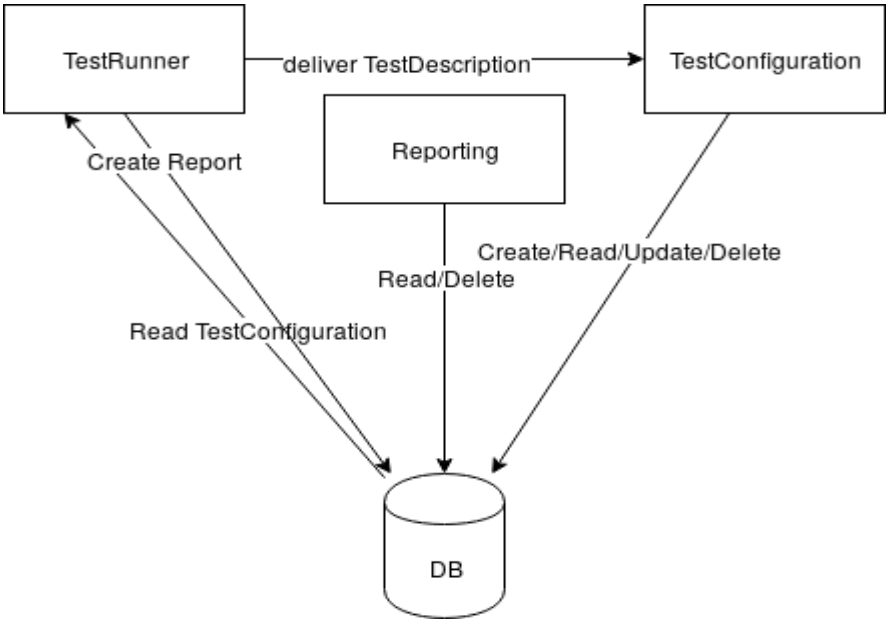

Abbildung 4: Schematische Darstellung des Datenflusses der Applikationskomponenten

Für die TestConfiguration laufen die CRUD-Operationen über den Application Service. Um eine Testkonfiguration erstellen zu können, benötigt dieser die TestDescription, also eine Beschreibung des zu startenden Tests.

Der TestRunner kennt natürlich die TestDescriptions (Metadaten) seiner eigenen Tests. Außerdem liest er eine TestConfiguration ein, um einen Test laufen zu lassen und erzeugt am Ende einen Report.

Das Reporting erzeugt keine eigenen Reports, kann diese aber löschen und lesen. Die Reports werden in einem eigenen Format in der DB abgelegt, die abgeleiteten Formate (HTML, CSV etc.) werden nicht persistiert.

Das Datenmodell des TestRunners ist wie in Abbildung 5 zu sehen strukturiert.

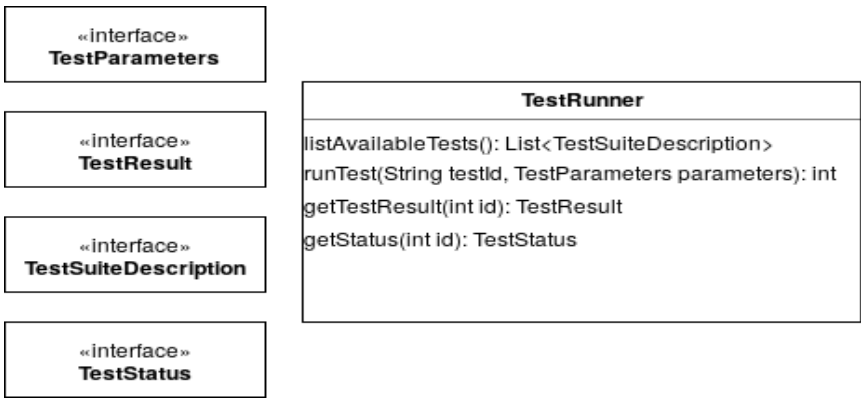

Abbildung 5: Datenmodell für Tests

Testframeworks können durch Implementierung des TestRunner-Interfaces angebunden werden. Dabei werden nicht alle Daten in der Datenbank vorgehalten, sondern lediglich die benutzerspezifischen Daten. Existierende Testklassen/Konformitätsklassen der Testframeworks werden jeweils im Dateisystem vorgehalten und eine Duplizierung der Test-Metadaten somit vermieden.

Für die Implementierung der Anbindung der Testframeworks haben wir zwei alternative Modelle entwickelt. Links in Abbildung 6 zu sehen ist ein externalisierter Ansatz, der die TEAM-Engine bzw. die ETF per REST-Schnittstelle oder in einem separaten Prozess anspricht. Rechts im Diagramm ist ein internalisierter Ansatz dargestellt. Dabei würden die TEAM-Engine bzw. die ETF direkt mittels ihrer Java-API angebunden. Welcher dieser Ansätze für welches Testframework am besten geeignet ist, wird im separaten Feinkonzept für die Tests eruiert.

In jedem Fall laufen die Tests asynchron, sodass diese zum Beispiel nach Abmeldung eines Benutzers im Hintergrund weiterlaufen (Anforderung 51).

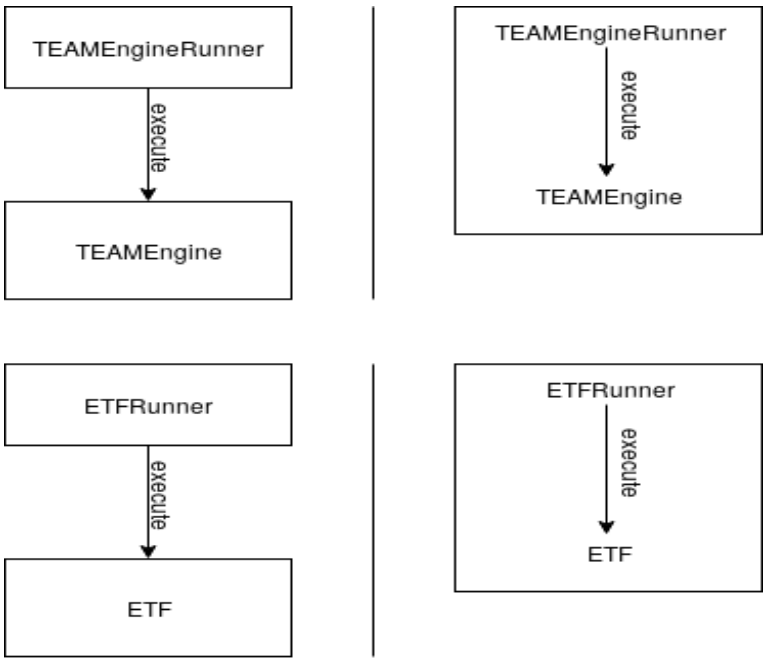

Abbildung 6: Anbindung der Testframeworks

4.5.2 Migrationskonzept

Es werden die bestehenden Tests zum Zeitpunkt der Bereitstellung der Alpha-Version in das neue System integriert. Dies kann durch den AN nach Implementierung der TEAM-Engine- und ETF-Module durch entsprechende Konfiguration der Anwendung ggf. direkt in der Oberfläche (in der Admin-Rolle) geschehen, alternativ durch eine automatische Migration der bestehenden Konfiguration.

Vor der Migration erfolgt eine Abstimmung zwischen AG und AN bezüglich der zu migrierenden Tests.

Nutzerbezogene Daten werden nach Absprache nicht migriert (Anforderung 64).

4.6 Security Layer

Der Zugriff von außen ist zunächst nur über einen Nginx-Container möglich, der über erreichbare HTTPS-Schnittstellen via Reverse Proxy Zugriff auf die einzelnen Komponenten ermöglicht, d. h. ein direkter Zugriff auf interne Endpunkte z. B. die Datenbank ist nicht möglich. Jegliche Kommunikation erfolgt somit verschlüsselt über den HTTPS-Port 443 oder den SMTP-Port 25 (Anforderung 77).

Weiterhin realisiert der Security Layer die Absicherung der REST-Schnittstellen, die Authentifizierung von Nutzern und die Zugriffskontrolle als Umsetzung des Rechte- und Rollenkonzeptes. Für diese Aufgaben wird das Modul Spring Security[10] des Spring Frameworks[11] verwendet. Dies bietet Schutz gegen geläufige Angriffe wie zum Beispiel XSRF-, XSS- und Rainbow Table Angriffe.

10 https://spring.io/projects/spring-security
11 https://spring.io/

4.7 Registrierung und Authentifizierung

Bei der Registrierung legt der Nutzer ein Passwort an, das mithilfe der von Spring Security implementierten Bcrypt-Methode serverseitig verschlüsselt und danach in der Datenbank gespeichert wird (Anforderung 63). Damit ist sichergestellt, dass Passwörter nie im Klartext gespeichert werden. Die Bcrypt-Implementierung versieht den Hash außerdem automatisch mit einem Salt, einer zufällig generierten Zeichenfolge. Diese erhöht die Entropie der Passwörter und verhindert Angriffe mit Rainbow Tables.

Die zu verwendende Methode unterscheidet sich damit von der im Angebot in Kapitel 2.8 beschriebenen SCRAM-SHA-256 Verschlüsselung, ist aber nicht mit Einschränkungen in der Sicherheit verbunden. Diese Methode ist die Standard-Verschlüsselung von Spring Security.

Die Registrierung erfordert die Eingabe eines Captchas zur Unterscheidung zwischen Mensch und Maschine um massenhafte automatisierte Anmeldungen zu vermeiden (Anforderung 14). Außerdem ist ein neuer Account zunächst inaktiv und wird erst freigeschaltet, wenn die Anmeldung über einen per E-Mail versendeten Link bestätigt wird (Double Opt-In Verfahren) (Anforderung 15).

Fachadministratoren und Systemadministratoren müssen beim Login zwingend eine Multifaktor-Authentifizierung benutzen (Anforderung 16). Beim Login wird neben dem Passwort ein zusätzliches, nur einmal gültiges Token benötigt. Zur Generierung dieses Tokens wird eine Mobile App oder eine Browser-Erweiterung verwendet. Die Kommunikation zwischen Authentifizierungstool und Applikation läuft dabei über einen Time-Based One-Time Passwort Algorithmus (TOTP). Sowohl für die Mobile App-als auch für die Browser-Variante gibt es zahlreiche Alternativen. Dem Benutzer bleibt es selbst überlassen, für welche Variante er sich entscheidet.

Das System unterstützt die gleichzeitige Anmeldung von einem User an mehreren Arbeitsplätzen (Anforderung 17).

4.8 Rechte- und Rollenkonzept

Das System unterscheidet zwischen vier Rollen, die Zugriff auf unterschiedliche Funktionen haben: Anonymer Benutzer (*ROLE_ANONYMOUS*), registrierter Benutzer (*ROLE_USER*), Fachadministrator (*ROLE_TECHADMIN*) und Systemadministrator (*ROLE_SYSADMIN*). Die jeweils übergeordnete Rolle erbt alle Berechtigungen der weniger berechtigten Rollen (Anforderung 62).

Die Funktionen, die zur Durchführung der in Kapitel 3 der Leistungsbeschreibung beschriebenen Anwendungsfälle benötigt werden, stehen über die REST-Schnittstellen zur Verfügung. Wird eine Funktion über eine Schnittstelle aufgerufen, wird zunächst geprüft, ob die Rolle des aktuellen Benutzers die notwendigen Privilegien besitzt, die aufgerufene Funktion auszuführen. Die Umsetzung dieser Zugriffskontrolle erfolgt mithilfe des Spring Security Frameworks. Da die GUI die Funktionen über die REST-Schnittstelle aufruft, ist auch diese damit abgesichert.

Durch die Verknüpfung von Funktionen und den Endpoints der API-Schnittstellen wird sichergestellt, dass das System den Nutzern nur die jeweils für den Anwendungsfall notwendigen Daten zur Verfügung stellt (Minimal-Prinzip).

Das System sieht die folgende Zuordnung zwischen Rollen und Funktionen bzw. Anwendungsfällen (siehe Anwendungsfälle Kapitel 3 der Leistungsbeschreibung) vor:

1. Als anonymer Benutzer (*ROLE_ANONYMOUS*) ist keine Authentifizierung mit Zugangsdaten erforderlich. Dem Nutzer stehen die folgenden Funktionen auf der Startseite zur Verfügung:
   - die Startseite aufrufen
   - Benutzerkonto anlegen (3.3) (*Schreiben*)
   - Schnelltest ausführen auf einzelner Ressource (3.9)
   - Abruf eines Testberichts zum Schnelltest, keine persistente Speicherung (3.10) (*Lesen*)
2. Ein registrierter und eingeloggter Benutzer (*ROLE_USER*) hat Zugriff auf folgende Funktionen:
   - die Funktionen des anonymen Benutzers (3.9, 3.10)
   - am System anmelden (3.2)
   - eigenes Benutzerkonto bearbeiten (3.4) und löschen (*Ändern* / *Löschen*)
   - Testkonfiguration anlegen (3.6) und verwalten (3.7) (*Schreiben* / *Ändern*)
   - Freischaltung eines Intervalltests beantragen (3.8)
   - Test ausführen (3.9)
   - Abruf eines Testberichts (3.10) (*Lesen*)
   - Testberichte löschen (3.11) (*Löschen*)
3. Einem Fachadministrator (*ROLE_TECHADMIN*) stehen nach Login mit Multifaktor-Authentifizierung folgende Funktionen zur Verfügung :
   - alle Privilegien des registrierten und anonymen Benutzers (3.2, 3.4, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11)
   - das Passwort eines Benutzerkontos ändern (Anforderung 19) (*Ändern*)
   - alle E-Mail-Adressen der im System registrierten Nutzer abrufen (3.5) (*Lesen*)
   - im Schattenmodus arbeiten (3.12)
   - Testklasse anlegen (3.13) und verwalten (3.15) (*Schreiben* / *Ändern*)
   - Konformitätsklasse anlegen (3.14) und aktualisieren (3.16) (*Schreiben* / *Ändern*)
   - Fachliches Monitoring durchführen (3.17) (*Lesen*)
4. Systemadministratoren (*ROLE_SYSADMIN*) dürfen alle Funktionen ausführen. Dazu gehören zusätzlich zu den Berechtigungen der anderen Rollen:
   - Basisparameter einstellen / System betreiben (3.18)
   - System Monitoring durchführen (3.19)

- Nutzer anlegen/löschen (*Schreiben / Löschen*)

4.9 Reporting Service

Der Reporting Service stellt Klassen zur Generierung von Testberichten auf Basis von Testergebnissen in den Ausgabeformaten PDF, CSV, XML und HTML bereit (Anforderung 52, 53, 54). Die exportierten Dateien enthalten jeweils formatunabhängig die gleichen Informationen (Anforderung 55). Für die Formate PDF, CSV und HTML wird dabei das Framework JasperReports[12] verwendet, welches es erlaubt, Testberichte als Templates außerhalb des Quellcodes abzulegen und somit eine Anpassung der Testberichte ohne Neustart des Systems/Services ermöglicht. Die Templates können, sofern nötig, mit dem Tool Jaspersoft Studio[13] mit Hilfe einer graphischen Nutzeroberfläche angepasst werden. Entgegen des Kriterienkatalogs wird die Komponente MapFish Print nicht verwendet, da in den Reports keine Kartenausgaben erforderlich sein werden (siehe Angebot K 2.5).

Alle Ausgabeformate werden dabei die Informationen aus Anforderung 56 beinhalten.

Für die Ausgabe im XML-Format, wird der Reporting-Service gegen das folgende XML-Schema validieren und serialisieren (siehe Anforderung 54):

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="https://testsuite.gdi-de.org/
testresult" targetNamespace="https://testsuite.gdi-de.org/testresult"
elementFormDefault="qualified">
  <xs:simpleType name="ResultType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="PENDING" />
      <xs:enumeration value="SUCCESS" />
      <xs:enumeration value="FAIL" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="MessageTypeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="request" />
      <xs:enumeration value="response" />
      <xs:enumeration value="text" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="ReportType">
    <xs:sequence>
      <xs:element name="Id" type="xs:positiveInteger" />
      <xs:element name="TestConfigurationId" type="xs:positiveInteger" />
      <xs:element name="TestClassId" type="xs:string" />
```

12 https://community.jaspersoft.com/project/jasperreports-library

13 https://community.jaspersoft.com/project/jaspersoft-studio

```
      <xs:element name="Timestamp" type="xs:dateTime" />
      <xs:element name="Name" type="xs:string" />
      <xs:element name="TestParameters" type="TestParameters" />
      <xs:element name="Test" type="TestType" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TestParameters">
    <xs:sequence>
      <xs:element name="Parameter" type="TestParameter" minOccurs="1" maxOccurs="unbounded" /
>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TestParameter">
    <xs:attribute name="name" type="xs:string" />
    <xs:attribute name="value" type="xs:string" />
  </xs:complexType>
  <xs:complexType name="TestType">
    <xs:sequence>
      <xs:element name="TestId" type="xs:string" />
      <xs:element name="Name" type="xs:string" />
      <xs:element name="Result" type="ResultType" />
      <xs:element name="Requirement" type="xs:string" />
      <xs:element name="Messages" type="MessagesType" />
      <xs:element name="Test" type="TestType" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MessagesType">
    <xs:sequence>
      <xs:element name="Message" type="MessageType" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MessageType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" type="MessageTypeType"></xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="Report" type="ReportType" />
</xs:schema>
```

5 Benutzeroberflächen inkl. Klick-Dummies

5.1 Allgemein

Die Benutzeroberflächen werden als Klick-Dummies über das Online-Tool „Moqups“[14] in einem eigenen geschützten Projektbereich bereitgestellt. Eine Kommentarfunktion erlaubt eine Diskussion über GUI, Workflows und Layout direkt am Entwurf. Diese so bereitgestellten Klick-Prototypen können als Diskussionsgrundlage dienen und mit Kommentaren direkt am Entwurf versehen werden (Anforderung 32). Die Klick-Dummies sollen hierbei vor allem die Usability und Benutzerführung veranschaulichen. Das Layout der Benutzeroberflächen sowie der Oberflächenelemente wird später festgelegt.

Das System wird als Webanwendung mit einer benutzerfreundlichen Oberfläche bereitgestellt. Die Oberfläche entspricht den Vorgaben der DIN EN ISO 9241 zur Mensch-Computer-Interaktion. Im Teil 110 der Norm sind die Grundsätze der Dialoggestaltung formuliert, die für die Softwareentwicklung von großer Bedeutung sind. Diese sind in die vorliegende Konzeption und damit auch in die Gestaltung der Mockups eingeflossen:

(1) Aufgabenangemessenheit
   a) ist die Funktionalität dazu geeignet, dass der Nutzer seine Aufgabe(n) erfüllen kann
   b) wurde darauf geachtet, aus Sicht des Nutzers unnötige Interaktionen zu minimieren

(2) Selbstbeschreibungsfähigkeit
   a) stehen ausreichende Möglichkeiten zur Unterstützung des Nutzers zur Verfügung, wie Hilfen, Rückmeldungen, Tooltips etc.
   b) kann der Nutzer jederzeit erkennen, wo in der Anwendung er sich gerade befindet
   c) ist für den Nutzer erkennbar, welche Aktionen er ausführen kann und wie sie bedient werden müssen

(3) Lernförderlichkeit
   a) ist die Anwendung so einfach und verständlich gestaltet, dass ein Nutzer sie in minimal möglicher Zeit erlernen kann
   b) stehen dazu Anleitungen zur Verfügung
   c) wird der Nutzer durch komplexe Vorgänge geführt, indem er Rückmeldung zu seiner Interaktion erhält

(4) Steuerbarkeit
   a) kann der Nutzer den Dialog selbst steuern, indem er z. B. selbst bestimmen kann, ihn zu starten, zu unterbrechen, die Geschwindigkeit zu beeinflussen

14 https://moqups.com

b) kann der Nutzer jederzeit erkennen, in welchem Prozess er sich gerade befindet

(5) Erwartungskonformität

a) folgt die Gestaltung konsequent erkennbaren Prinzipien (Konsistenz) und entspricht allgemein üblichen Konventionen

b) kann der Nutzer vorhersehen, welche Bearbeitungszeiten Aktionen benötigen

(6) Individualisierbarkeit

a) kann der Nutzer die Anwendung an seine individuellen Bedürfnisse anpassen

(7) Fehlertoleranz

a) toleriert die Anwendung fehlerhafte Eingaben des Nutzers, indem sie ihn darauf hinweist und Korrekturen zulässt

b) bleibt die Funktionalität der Anwendung auch dann erhalten, wenn unerwartete Eingaben oder Aktionen des Nutzers erfolgen

Soweit es zweckmäßig erscheint, wird die Anwendung gemäß den Anforderungen der Barrierefreie-Informationstechnik-Verordnung - BITV 2.0 umgesetzt (Anforderung 1, 82). Dabei orientiert sich der AN, wie mit dem AG abgesprochen, am BITV-Selbstbewertungsformular. Die im Ergebnis des Selbsttests ausgefüllten Fragebögen werden dem AG übergeben.

Im Fokus stehen dabei vor allem die Anforderungen der BITV, die den Zugang zur und die Nutzung der Anwendung durch motorisch und/oder in ihrer Sehleistung beeinträchtigte Personen ermöglicht. Relevant sind für das Projekt vor allem:

- Text auf 200 % vergrößerbar
- Verzicht auf Schriftgraphiken
- Nutzbarkeit ohne Maus
- Vermeidung von Tastaturfallen
- Verzicht auf Flackern
- sinnvolle Dokumententitel
- Angabe der Hauptsprache
- Vermeidung unerwarteter Kontextänderungen bei Fokus
- Vermeidung unerwarteter Kontextänderungen bei Eingabe
- konsistente Navigation
- konsistente Bezeichnung
- korrekte Syntax

Mit dem AG wurde vereinbart, dass auf die Realisierung einer Version mit einfacher Sprache verzichtet werden kann, da die infrage kommende Nutzergruppe eine universitäre oder spezialisierte Fachausbildung besitzt, die für das Verständnis der Inhalte nicht auf einfache Sprache angewiesen ist.

Die Oberflächen der Webanwendung werden nach Umsetzung der funktionalen Workflows und der Usability gemäß den Corporate Design-Vorgaben der GDI-DE angepasst (Anforderung 29). Da es nur in begrenztem Maße Vorgaben zum Corporate Design gibt, ist eine Abstimmung zum Layout anhand von Klickdummies an dieser Stelle nicht sinnvoll. Dies wurde im gemeinsamen Projekttreffen am 13.06.2019 in Frankfurt beschlossen. Das Layout wird in der Programmierung selbst umgesetzt und zwischen AN und AG abgestimmt.

Alle Oberflächen werden, soweit dies für die benötigten Workflows möglich und sinnvoll ist, nach den Gesichtspunkten des Responsive Designs umgesetzt (Anforderung 28). Oberflächen bzw. Oberflächenelemente werden demnach je nach Bildschirmgröße und/oder Device anders dargestellt, um die *Benutzerfreundlichkeit* zu gewährleisten. Besonders die in den o. g. Punkten 2b, 2c und 3a aufgeführten Aspekte sind hierbei zu berücksichtigen. Dies betrifft beispielsweise die umzusetzenden tabellarischen Darstellungen von Testkonfigurationen, -berichten usw. Da sich die geforderten Spalten einer Tabelle (vgl. z. B. Anforderung 9) nicht ohne horizontale Scrollbalken auf kleinen Bildschirmen darstellen lassen, kann an dieser Stelle beispielsweise mit einer reduzierten tabellarischen Ansicht mit einer geringeren Anzahl an initial angezeigten Spalten, die der Nutzer selbst mit weiteren ergänzen kann, gearbeitet werden. Dem Nutzer hierbei die Möglichkeit einzuräumen, selbst entscheiden zu können, welche Informationen er wann angezeigt bekommen möchte, entspricht auch der Forderung nach Individualisierbarkeit (s. o. g. Punkt 6a). Eine genaue Festlegung, welche Bildschirmgrößen und/oder Devices berücksichtigt werden müssen und welche Oberflächen bzw. Oberflächenelemente zweckmäßig anzupassen sind, erfolgt im Projektverlauf durch Abstimmungen zwischen AN und AG.

Sämtliche interaktive Elemente werden über Tooltips erläutert und damit den Anforderungen 2a und 3b der DIN EN ISO 9241 entsprochen. Die Tooltips werden bei Grafiken (z. B. Buttons) auch als Alternativtexte angezeigt. Wie alle sprachlichen Elemente, werden auch die Tooltips in einer Datei außerhalb des eigentlichen Quellcodes vorgehalten und sind dort für die Sprachen Deutsch und Englisch konfigurierbar (Anforderung 24, 67). Für externe Inhalte, z. B. Fehlermeldungen der Testsuites oder sonstiger Dienste, kann eine Mehrsprachigkeit nur gewährleistet werden, wenn diese von den externen Services zur Verfügung gestellt werden (Anforderung 66).

Das System verfügt über eine Online Hilfe. Diese kann entweder als gesonderte Webseite aufgerufen werden, alternativ wird auch eine kontext-sensitive Hilfe angeboten. Dabei klickt der Nutzer mit einem Hilfe-Tool auf ein Objekt und es wird der entsprechende Hilfetext angezeigt (Anforderung 26). Der Inhalt der Online-Hilfe wird über das Code-Repository gepflegt und ist nicht fest im Programm-Code eingebettet.

Das System bietet Suchmasken zum Auffinden von Testkonfigurationen, Testberichten und Benutzerkonten über die in der Leistungsbeschreibung genannten Felder an (Anforderung 22). Bei Frei-

texteingabe unterstützt das System den Benutzer durch eine Autovervollständigung (Anforderung 23). Mittels der oberhalb der tabellarischen Ansichten angeordneten Volltextsuche kann dabei pro Entität auf den dafür konfigurierten Feldern gesucht werden. Derartige Volltextsuchen sind in modernen Webseiten üblich und die Anwendung erfüllt damit die Erwatungskonfomität (s. o. g. Punkt 5a der DIN EN ISO 9241). Zusätzlich bieten die tabellarischen Darstellungen der Entitäten die Möglichkeit, über die Spaltenköpfe der Tabelle auf ein bestimmtes Feld zu filtern.

Für den Prozess „Anlegen einer neuen Testkonfiguration“ kann bei Bedarf ein Schritt-für-Schritt-Assistent integriert werden (Anforderung 25). Insbesondere bieten derartige Assistenten Nutzern bei der erstmaligen Bedienung komplexer Funktionen die Möglichkeit, diese zu erlernen (s. o. g. Punkt 3 Lernförderlichkeit). Der Benutzer kann in jedem Schritt vor- und zurückspringen, um Eingaben vor dem Speichern ggf. nochmals zu verändern und sieht übersichtlich, in welchem Schritt er sich befindet. Den Anforderungen unter o. g. Punkt 2 wird damit entsprochen. Es wird pro Schritt die Eingabe clientseitig überprüft, Eingaben werden aber erst nach der Vervollständigung aller erforderlichen Pflichteingaben einmalig ins System gespeichert. Der Benutzer kann hier selbst entscheiden, ob er den Assistenten oder die ungeführte Formulareingabe nutzen möchte (s. o. g. Punkt 6 der DIN EN ISO 9241).

Für die beiden Prozesse „Test ausführen“ und „Testbericht abrufen“ ist es nicht notwendig, gesonderte Assistenten zur Verfügung zu stellen, da die Benutzerführung der Webanwendung an diesen Stellen eindeutig nachvollziehbar ist.

Sofern im System ein aufgerufener Prozess eine Prozessierungsdauer von 3 Sekunden überschreitet, wird ein Fortschrittsbalken angezeigt. Dazu wird beim Absenden einer Nutzer-System-Interaktion am System abgefragt, wie lange die Bearbeitung voraussichtlich dauern wird. Hierfür ist die Implementierung einer entsprechenden Schnittstelle samt Anwendung auf alle potentiell betroffenen Nutzer-Systeminteraktionen erforderlich. Sobald das Laufzeitverhalten eingeschätzt werden kann, wird vom AN ein Vorschlag unterbreitet, für welche Lese- und Speichervorgänge dies im Detail vorgesehen werden muss, d. h. welche Vorgänge die vorgegebenen 3 Sekunden voraussichtlich überschreiten werden (z. B. Testausführung). Übliche Lese- und Speichervorgänge überschreiten die vorgegebene Dauer i. d. R. nicht (Anforderung 27).

Für die gesamte Umsetzung werden keine iFrames verwendet (Anforderung 31).

Das System ist in aktuellen Versionen von Firefox, Chrome, Internet Explorer (ab Version 11) und Safari lauffähig. Die genannten Browser werden in der vorgegebenen Konfiguration nach SAGA-Modul “Technische Spezifikationen de.bund Version 5.0.0, Kapitel 6“ unterstützt (Anforderung 78, 79). Um die Lauffähigkeit in den verschiedenen Browsern zu gewährleisten, wird für sämtliche Client-Entwicklung auf bewährte Frameworks zurückgegriffen. Als Kern-Framework für den Client und das clientseitige Rendern der interaktiven Oberflächen werden wir das JavaScript Framework ReactJS einsetzen.

Die folgenden Kapitel beschreiben die geforderten Benutzeroberflächen, die jeweils auch im „Moqups“-Tool zur Verfügung stehen.

5.2 Ohne Anmeldung

5.2.1 Startseite

Der Eingang zur Anwendung ist eine Startseite, die die folgenden Komponenten enthält (Anforderung 2):

- Informationen zum Inhalt der GDI-DE Testsuite (aus Altsystem oder durch AG bereit gestellt)
- Informationen zum Datenschutz
- Eine Weiterleitung zur Anmeldung am System
- Eine Weiterleitung zur die Nutzer-Registrierung
- Eine Weiterleitung zum Aufruf der Schnelltestoberfläche
- Sprachauswahl
- Aufrufmöglichkeit Hilfe
- Aufrufmöglichkeit Kontaktinfo und Formular (auf allen Seiten)

5.2.2 Anmelden

Die Anmeldeoberfläche ermöglicht es Benutzern, die bereits ein Benutzerkonto besitzen, sich durch Eingabe von Benutzername und Passwort am System anzumelden.

Zusätzlich werden folgende Weiterleitungen angeboten:

- „Passwort vergessen“-Oberfläche
- „Registrierung“-Oberfläche

Das System ermöglicht es, dass sich ein Benutzer mehrfach mit identischem Benutzerkonto anmelden und die Funktionalität unabhängig von der anderen Instanz nutzen kann. Ist ein Benutzer mehrfach angemeldet, werden entsprechende Warnmeldungen ausgegeben (Anforderung 17).

5.2.3 Anmelden Multi-Factor-Authentifizierung (Fachadministrator/Systemadministrator)

Da sich Fachadministratoren nur über ein Multi-Factor-Authentifizierungsverfahren (MFA) am System anmelden dürfen, wird Benutzern, die als Fachadministrator im System registriert sind, eine entsprechende zusätzliche Anmeldeoberfläche zur Verfügung gestellt (Anforderung 16).

Das genaue Vorgehen zum MFA findet sich in Kapitel 4.7.

5.2.4 Passwort vergessen

In dieser Oberfläche hat der Benutzer die Möglichkeit, über Eingabe der E-Mail-Adresse in das entsprechende Feld eine E-Mail zum Zurücksetzen des Passworts zu generieren.

5.2.5 Registrierung

In dieser Oberfläche steht dem Benutzer ein Formular zur Verfügung, sich neu am System zu registrieren (Anforderung 14). Das Formular umfasst die folgenden Felder:

- Benutzername
- Passwort
- Passwort wiederholen
- Vorname
- Name
- E-Mail-Adresse
- Sprache
- Bestätigung der Nutzungsbedingungen
- Bestätigung Datenschutzbestimmungen und Zustimmung zur Verarbeitung personenbezogener Daten
- Captcha-Bestätigung

Bei der Eingabe wird clientseitig bereits, soweit möglich, auf korrekte und vollständige Eingabe geprüft und dem Nutzer eine Rückmeldung über fehlende Eingaben gegeben. Die Eingabe besitzt einen Test zur Unterscheidung zwischen Mensch und Maschine (Captcha).

Nach Abschicken des Formulars wird eine E-Mail zur Bestätigung der Registrierung mit einem Link an den Benutzer verschickt. Mit Anklicken des Links erfolgt die Freischaltung des Kontos (Anforderung 15).

5.2.6 Schnelltest

Über diese Weboberfläche können nicht-registrierte Nutzer einen Schnelltest mit eingeschränktem Funktionsumfang online ausführen. Dabei können lediglich vordefinierte Standard-Testkonfigurationen für jede Testklasse ausgewählt und verwendet werden. Ein Testbericht wird als Ergebnis angezeigt und kann heruntergeladen werden., Dieser kann aber nicht gespeichert werden, da kein Benutzerkonto vorliegt (Anforderung 60, 61).

5.3 Angemeldet als Benutzer

5.3.1 Testkonfigurationen

Für einen angemeldeten Benutzer öffnet sich die Webanwendung als Startpunkt im ersten Menüreiter „Testkonfigurationen“ (Anforderung 3). Diese Oberfläche enthält die tabellarische Übersicht zu allen Testkonfigurationen des angemeldeten Benutzers (Anforderung 6, 45). Die Tabelle enthält dabei mindestens die geforderten Informationen laut Leistungsbeschreibung und bietet über Schaltflächen in den Spaltenköpfen die Möglichkeit, die Tabelle nach einzelnen Spalten zu

sortieren und zu filtern, sofern dies für den Spalteninhalt sinnvoll ist (Anforderung 8, 12). Die Tabelle enthält keine Scrollbalken; überschreitet die Anzahl der Zeilen den verfügbaren Platz, kommt deshalb ein Paging zum Einsatz (Vermeidung vertikaler Scrollbalken). Zur Vermeidung horizontaler Scrollbalken wird der Inhalt der Tabelle bei Bedarf angepasst und die Anzahl initialer Spalten reduziert.

Für jede Testkonfiguration gibt es in der Tabelle die Möglichkeit, den letzten Testbericht sowie eine Tabelle mit allen Testberichten zu dieser Konfiguration aufzurufen. Der Benutzer wechselt über diese Aktionen in den zweiten Menüreiter „Testberichte“.

Jede Testkonfiguration kann über entsprechende Schaltflächen in der Tabelle gelöscht oder bearbeitet werden. Jede Testkonfiguration kann über eine entsprechende Schaltfläche in der Tabelle ausgeführt werden (Anforderung 7, 46). Ist ein Testdurchlauf einmal gestartet, kann er über eine Schaltfläche jederzeit durch den Benutzer wieder gestoppt werden (Anforderung 47). Der Testdurchlauf wird somit abgebrochen und der Ausgangszustand ist wieder hergestellt.

Ein durch den Benutzer gestarteter Testdurchlauf wird auch dann weiter ausgeführt, wenn der Benutzer sich vom System abmeldet (Anforderung 51). Nach erneuter Anmeldung an der Anwendung kann der Benutzer sich mit Hilfe der Menüs 'Testkonfiguration' und 'Testberichte' über den Status der Testausführung informieren.

Über eine separate Schaltfläche kann der Benutzer eine neue Testkonfiguration anlegen (Anforderung 3).

Durch eine Mehrfach-Selektion von Zeilen können in einem Schritt über die Schaltfläche „Selektierte Testkonfigurationen löschen“ mehrere Testkonfigurationen gelöscht werden.

Die Header-Leiste der Anwendung ermöglicht es einem angemeldeten Benutzer jederzeit, sich über eine Schaltfläche vom System abzumelden (Anforderung 3).

5.3.2 Testkonfiguration anlegen/bearbeiten

In dieser Oberfläche kann der Benutzer eine neue Testkonfiguration erstellen (Anforderung 5, 44) bzw. eine vorhandene Konfiguration bearbeiten. Das Eingabeformular besitzt folgende Felder:

- ID: wird automatisch generiert (nach Speichern bzw. im Bearbeitungsmodus)
- Name: Freitext
- Beschreibung: Freitext
- Testklasse: Auswahlbox mit den bestehenden Testklassen
- Konformitätsklassen: Anzeige und Auswahlmöglichkeiten der zur Testklasse gehörenden Konformitätsklassen verpflichtend/optional
- Dienstequalität: Auswahlmöglichkeit, ob Dienstequalität mit getestet werden soll (nur bei Testklassen, die Dienste testen)
- Datensatz/Dienst: Eingabemöglichkeit URLs / Dateien

- bei Eingabe mehrerer Ressourcen entstehen beim Speichern entsprechend viele einzelne Testkonfigurationen
- bei abgesicherten Diensten zusätzlich Eingabe von Authentifizierung-Parametern (Anforderungen 38, 39)

- E-Mail-Benachrichtigung: Auswahlmöglichkeit, ob eine Benachrichtigungs-E-Mail verschickt werden soll, wenn der Test durchgeführt wurde und ein Testergebnis vorliegt: nein/ja/bei Fehler
- Ausführungswiederholung: Angabe eines Zeitraums und Intervalls, in dem Tests regelmäßig ausgeführt werden sollen.
  Eine Ausführungswiederholung mit einem höheren Intervall als eine Testausführung pro Tag ist ohne Zustimmung des Dienstbetreibers nicht möglich. Mit Zustimmung des Dienstbetreibers gibt es hier nur die Einschränkung, dass das kleinste Intervall auf eine Stunde festgelegt ist (Anforderung 48, 49). Es wird ein Vorschlag vorgelegt, wie diese Anforderung bestmöglich umgesetzt werden kann.

Der Benutzer hat in der Oberfläche die Möglichkeit, sich zwischen einer geführten Eingabe per Assistenten oder für die ungeführte Formulareingabe zu entscheiden (Anforderung 25). Nutzt er den Assistenten zum Anlegen der Testkonfiguration, wird er Schritt-für-Schritt durch das Formular geführt und kann jederzeit zu einzelnen Schritten zurückkehren.

Für beide Varianten gilt, dass sich Formularinhalte in Abhängigkeit zu eingebenen Daten verändern können, da beispielsweise die Auswahl einer Testklasse festlegt, welche Konformitätsklassen angeboten werden. Ebenso ist geplant, anhand der hochgeladenen zu testenden Ressource bereits eine eingeschränkte Auswahl an möglichen Testklassen anzubieten.

5.3.3 Testberichte

Die Webanwendung bietet dem Benutzer im Menüreiter „Testberichte“ die Möglichkeit, die tabellarische Übersicht über alle Testberichte der eigenen Testkonfigurationen aufzurufen. Die Tabelle enthält dabei mindestens die geforderten Informationen laut Leistungsbeschreibung. Das Verhalten der Tabelle ist identisch mit der in 5.3.1 Testkonfigurationen beschriebenen und erfüllt damit die Anforderungen 9, 12 und 59.

Für jeden Testbericht gibt es über die Download-Funktion die Möglichkeit, sich den Testbericht im gewünschten Format herunterzuladen. Zur Verfügung stehen die Formate HTML, CSV, PDF und XML (Anforderung 13, 52, 53, 54). Die Inhalte und die Struktur der heruntergeladenen Formate sind jeweils gleich und enthalten alle in der Leistungsbeschreibung geforderten Elemente (Anforderung 55, 56).

Für jeden Testbericht kann über die entsprechende Spalte die Detailansicht zu einem Testbericht aufgerufen werden (Anforderung 10).

5.3.4 Detailansicht zum Testbericht

In dieser Oberfläche erhält der Benutzer eine Detailansicht zum Testergebnis. Diese Ansicht enthält die in der Leistungsbeschreibung geforderten Angaben (Anforderung 10):

- Übersicht der Ergebnisse je Konformitätsklasse (Anzahl bestanden, mit Fehler, mit Warnung)
- Auflistung der durchgeführten Einzeltests
- Beschreibung der ATS je Einzeltest
- Ergebnis der Einzeltests (Erfolgsmeldung, Fehlermeldung)
- Abrufbare Links zur Ressource, Anfrage und Antwort zu jedem Test
- Zusätzlich bei Schemavalidierungstests Referenz zur Fehlerfundstelle im XML-Dokument

Die Detailansicht besitzt eine Möglichkeit zur Filterung nach den Statuslevels bestanden/mit Warnung/mit Fehler (Anforderung 11). Zudem kann der Benutzer den Detailgrad des Testberichts einstellen (Anforderung 58). Fehlermeldungen sind immer nach dem gleichen Muster aufgebaut, wie laut Leistungsbeschreibung gefordert (Anforderung 57). Die genaue Ausformung eines Testberichts wird im Rahmen des Feinkonzepts für den Test Framework Adapter Layer ausgearbeitet.

Ein Testbericht kann auch in dieser Oberfläche über eine entsprechende Schaltfläche in den geforderten Formaten heruntergeladen werden.

5.3.5 Kontoverwaltung

Das System stellt angemeldeten Benutzern eine Oberfläche zur Verfügung, in der die eigenen Benutzerkonto-Daten verändert werden können und das Benutzerkonto gelöscht werden kann (Anforderung 18).

Folgende Daten können im Formular durch den Benutzer verändert werden:

- Benutzername
- Passwort
- Vorname
- Nachname
- E-Mail-Adresse
- Sprache

Über eine separate Schaltfläche kann das Benutzerkonto komplett gelöscht werden.

5.4 Angemeldet als Fachadministrator

Einem Benutzer, der im System die Rolle Fachadministrator besitzt, stehen sämtliche Oberflächen zur Verfügung, die auch ein normaler Benutzer sieht (siehe Kapitel 5.3 zu angemeldetem Benutzer).

Beim Anlegen von Testkonfigurationen wird dem Fachadministrator ergänzend ermöglicht, eine Testkonfiguration als Standardkonfiguration für den Schnelltest zu kennzeichnen.

Darüber hinaus stehen dem Fachadministrator weitere Oberflächen zur Verfügung.

5.4.1 Testklassenverwaltung

Das System besitzt eine Oberfläche zur Verwaltung von Testklassen (Anforderung 4). Diese kann über den entsprechenden Menüreiter angesteuert werden. Neben einer tabellarischen Übersicht gibt es an dieser Stelle die Möglichkeit zur Neuanlage von Testklassen.

Die genaue Ausarbeitung der Oberflächen zur Testklassenverwaltung erfolgt im Rahmen der Feinspezifikation zum Test Framework Adapter Layer im weiteren Projektverlauf. Da es sich bei den Fachadministratoren um eine eher kleinen Personenkreis handelt, wird vorgeschlagen, in dieser Phase einen Workshop zusammen mit ihnen durchzuführen, um gemeinsam Umsetzungsvorschläge zur Testklassenverwaltung zu diskutieren und festzulegen.

5.4.2 Nutzerverwaltung

Dem Fachadministrator steht über den Menüreiter „Nutzerverwaltung“ eine Oberfläche mit einer tabellarischen Übersicht zu allen vorhandenen Benutzerkonten zur Verfügung (Anforderung 20). Die Tabelle enthält dabei mindestens die geforderten Informationen laut Leistungsbeschreibung. Das Verhalten der Tabelle ist identisch mit der in 5.3.1 Testkonfigurationen beschriebenen und erfüllt damit die Anforderung 12.

Pro Benutzereintrag in der Tabelle kann der Fachadministrator über eine Schaltfläche das jeweilige Benutzerkonto bearbeiten. Innerhalb des Benutzerkontos darf er nur das Passwort für einen Benutzer verändern.

Pro Benutzereintrag in der Tabelle kann der Fachadministrator über eine Schaltfläche das jeweilige Benutzerkonto löschen. Dabei werden auch sämtliche zugehörige Testkonfigurationen und -berichte des Benutzers gelöscht.

Durch eine Mehrfach-Selektion von Zeilen können in einem Schritt über die Schaltfläche „Selektierte Benutzer löschen“ mehrere Benutzer gelöscht werden.

Über eine separate Schaltfläche können neue Benutzer angelegt werden (Anforderung 19).

Pro Benutzereintrag in der Tabelle kann der Fachadministrator über die Schaltfläche „Schattenmodus“ in die Benutzeransicht des jeweiligen Benutzers wechseln und hat somit Zugriff auf dessen Testkonfigurationen und Testberichte sowie sämtliche Funktionalitäten dieser Oberflächen (Anforderung 17, 30).

Sobald der Fachadministrator sich im Schattenmodus befindet, wird er in der Oberfläche über einen zusätzlichen Dialog/Popup permanent auf diesen Modus hingewiesen, über eine Schaltfläche innerhalb dieses Dialogs kann der Schattenmodus jederzeit beendet werden. Auch der normale Benutzer wird im Falle des gleichzeitigen Schattenmodus durch einen Fachadministrator entsprechend darauf hingewiesen.

Über eine separate Schaltfläche ist es möglich, alle E-Mail-Adressen der Benutzer des Systems auszugeben. Diese Funktionalität wird nicht über einen „mailto“-Link zur Verfügung gestellt (Anforderung 21).

5.4.3 Monitoring

Dem Fachadministrator steht über den Menüreiter „Monitoring“ eine Oberfläche zur Verfügung, auf der Kennzahlen zur fachlichen Nutzung visualisiert sind (Anforderung 50). Der Fachadministrator kann einen bestimmten Zeitpunkt auswählen und es werden die entsprechenden Werte angezeigt (standardmäßig werden die aktuellen Werte angezeigt). Hier können neben der freien Eingabe von Zeiträumen auch übliche Zeitintervalle wie beispielsweise „letzte 24h“, „vergangene Woche“ etc.) zur Direktauswahl angeboten werden.

Wir schlagen die Ausgabe folgender Kennzahlen vor:

- Anzahl angemeldeter Benutzer
- Anzahl registrierter Benutzer
- Anzahl aktiver Benutzer
- Anzahl laufende Tests
- Anzahl Testläufe pro Testklasse (davon bestanden/mit Warnung/mit Fehler)
- Anzahl aktivierter Intervalltests
- Anzahl Testklassen
- Laufzeiten von Tests
- Anzahl an Testläufen nach Ziel-URLs/Ziel-Domains/IP

Die Oberfläche unterstützt die grafische Anzeige bestimmter Kennzahlen für ein vom Fachadministrator definiertes Zeitintervall (Tag, Monat, Jahr).

Der Fachadministrator kann sich einen Bericht im CSV- oder PDF-Format mit allen Kennzahlen herunterladen.

5.5 Angemeldet als Systemadministrator

Einem Benutzer, der im System die Rolle Systemadministrator besitzt, stehen sämtliche Oberflächen zur Verfügung, die auch ein normaler Benutzer (siehe Kapitel 5.3 zu angemeldetem Benutzer) sowie ein Fachadministrator (siehe Kapitel 5.4 zu angemeldetem Fachadministrator) sieht.

Darüber hinaus steht dem Systemadministrator in der Nutzerverwaltung noch die Möglichkeit zur Verfügung, die Fachadministratoren zu verwalten, d. h. Benutzern die Rolle Fachadministrator zuzuweisen oder zu entziehen. Er verfügt auch über die Möglichkeit, zu einem Benutzer eine Übersicht über dessen im System gespeicherten personenbezogenen Daten zu erzeugen (Nachweis- und Auskunftspflicht gemäß DSGVO).

6 IT-Sicherheitsanforderungen und Schutzbedarf

Dieses Thema wird im Sicherheitskonzept behandelt und weiter ausgeführt.

7 Schnittstellenübersicht

Wir stellen REST-Schnittstellen mit folgender Funktionalität zur Verfügung:

- Testklassen/Konformitätsklassen anlegen/modifizieren/entfernen (Fachadministrator)
- Benutzerverwaltung (Fachadministrator)
- Testkonfigurationen anzeigen, verwalten und ausführen (Benutzer)
- Testberichte verwalten (Benutzer)

Die Schnittstellen decken alle Funktionen, die angemeldeten Nutzern zur Durchführung von Tests und zum Abrufen von Testergebnissen zur Verfügung stehen, ab (Anforderung 87). Dabei unterstützen die Schnittstellen Anfragen im UTF-8 Encoding und geben dies auch in Antworten aus.

Die REST-Schnittstellen werden nach Open API[15] spezifiziert (Anforderung 88).

8 Lieferumfang

Der Lieferumfang ist den Kapiteln 5 und 6.3 des Projekthandbuches zu entnehmen.

9 Abnahmekriterien und Vorgehen zur Ausgangsprüfung

Die Abnahme der funktionalen Anforderungen an das System erfolgt auf Basis der beschriebenen Anwendungsfälle (laut Lastenheft) im Benutzerhandbuch. Die durchgeführten Tests sind in der Abnahmetestdokumentation beschrieben. Diese ist die Grundlage für das Abnahmeprotokoll.

Für alle automatisierten Tests liegen die entsprechenden Dokumentationen vor (siehe Prüfspezifikation Software).

Alle weiteren Nutzungsdokumentationen sind bereitgestellt.

15 https://swagger.io/specification/

10 Liste der Anforderungen (Lastenheft)

Folgende Liste enthält eine Kurzfassung der Anforderungen aus dem Lastenheft, auf die im vorliegenden technischen Feinkonzept Bezug genommen wurde. Zusätzlich enthält die Tabelle eine Verlinkung in die Kapitel des vorliegenden Feinkonzepts, so dass sie auch wie eine Art Register verwendet werden kann. Mit der Spalte 'Usecase' wird die Verbindung zu dem Teil der Leistungsbeschreibung hergestellt, in dem die Anwendungsfälle schematisch skizziert sind.

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
0	Vollständiger Funktionsumfang der bisherigen Lösung ist Bestandteil des neuen Systems (SOAP-API nicht mehr erforderlich)	3.2, 3.3, 3.4, 3.6, 3.7, 3.9, 3.10, 3.11, 3.13, 3.14, 3.15, 3.16	2
1	Webanwendung mit Benutzeroberfläche (gemäß Vorgaben der DIN EN 9241 und Beachtung BITV)	3.2, 3.3, 3.4, 3.6, 3.7, 3.9, 3.10, 3.12, 3.13, 3.14, 3.15, 3.16, 3.17	5.1
2	Startseite (Informationen zu Inhalt GDI-DE Testsuite, Datenschutz, Anmeldung/ Nutzerregistrierung)	3.2, 3.3	5.2.1
3	Startseite für angemeldete Benutzer	3.2, 3.3, 3.6, 3.7, 3.12, 3.15, 3.16, 3.17	5.3.1

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
4	Webseite zur Verwaltung von Test- und Konformitätsklassen	3.8, 3.13, 3.14, 3.15	4.5 5.4.1
5	Benutzeroberfläche zur Erstellung einer Testkonfiguration	3.6, 3.8	5.3.2
6	Benutzeroberfläche „Testmanagement“ (Übersicht aller Testkonfigurationen)	3.7	5.3.1
7	Funktion „Testkonfiguration ausführen“	3.7 3.9	5.3.1
8	Sortierbarkeit der Tabelle mit den Testkonfigurationen	3.7	5.3.1
9	Übersicht Testberichte (tabellarische Übersicht aller Testberichte eines Nutzers	3.10, 3.11	5.1 5.3.3
10	Detailinformationen aus „Übersicht Testberichte“ abfragbar	3.10, 3.11	5.3.3 5.3.4
11	Filterfunktion der Detailansicht zu Testberichten nach Status des Einzeltest	3.10, 3.11	5.3.4
12	Sortierbarkeit der Tabellen über die Kopfzeile (spaltenweise, auf- oder absteigend)	3.5, 3.7, 3.10, 3.11, 3.12	5.3.1 5.3.3 5.4.2
13	Auslieferung von Testberichten in Webseitenform	3.10, 3.11	5.3.3
14	Funktionalität zur Registrierung eines neuen Benutzerkontos	3.3	4.7 5.2.5
15	Double Opt-In Verfahren zur Registrierung neuer Benutzerkonten	3.3	4.7 5.2.5
16	Multi-Factor Authentifizierungsverfahren für Login Fachadministrator	3.2	4.7 5.2.3
17	gleichzeitige Anmeldung von zwei Nutzern an einem Benutzerkonto	3.2, 3.12	5.2.2 5.4.2

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
18	Funktionalität zur Bearbeitung/zum Löschen des eigenen Benutzerkontos für registrierte Nutzer	3.4	5.3.5
19	Funktionalität zum Anlegen, Bearbeiten, Löschen von Benutzerkonten für Fachadministratoren	3.4	5.4.2
20	tabellarische Übersicht „Benutzerkonten“ für Fachadministratoren	3.4	5.4.2
21	Funktionalität für den Fachadministrator zum Auslesen aller E-Mail-Adressen der Nutzer	3.5	5.4.2
22	Suchfunktion zum Auffinden von Testkonfigurationen, Testberichten und Benutzerkonten	3.4, 3.7, 3.11	5.1
23	Suche besitzt für alle Eingabefelder eine Autofill-Funktion	3.4, 3.7, 3.11	5.1
24	Tooltips für alle interaktiven Elemente	3.2, 3.3, 3.4, 3.5, 3.6, 3.7 3.9, 3.10, 3.11, 3.12, 3.13, 3.14, 3.15	5.1
25	Assistenten für die Prozesse „Test ausführen“, „Bericht abrufen“, „Anlegen einer Testkonfiguration“	3.6 3.9, 3.10	5.1 5.3.2

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
26	Online-Hilfe	3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.9, 3.10, 3.11, 3.13, 3.14, 3.15, 3.16, 3.17	5.1
27	Fortschrittsanzeige bei Nutzer-System-Interaktion ab 3 sec Dauer	3.2, 3.3, 3.4, 3.5, 3.6, 3.9, 3.10, 3.12, 3.13, 3.14	5.1
28	Responsive Design		5.1
29	Nutzung des Corporate Design der GDI-DE		5.1
30	Schattenmodus für Fachadministrator	3.12	5.4.2
31	Es dürfen keine iFrames verwendet werden.		5.1
32	Bereitstellung eines Click-Prototypen		5.1
33	Abdeckung aller QoS Anforderungen aus den INSPIRE Guidance Dokumenten	3.6, 3.9, 3.13, 3.14, 3.15	4.5

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
34	Konzept für Aktualisierung von Testklassen im laufenden Betrieb	3.14, 3.15	4.5
35	Funktionalität zur Erstellung neuer Konformitätsklassen	3.13, 3.14	4.5
36	Unterstützung von ETS auf Basis von TestNG und CTL	3.13, 3.14, 3.15	4.5
37	Mechanismus für die Aktualisierung von Testklassen im laufenden Betrieb für den Fachadministrator	3.15	4.5
38	Unterstützung des Testens geschützter Dienste	3.9, 3.13, 3.15	5.3.2
39	Ermöglichung der Angabe mehrerer Ressourcen zu einer Testklasse in einer Testkonfiguration und deren Test	3.6, 3.7, 3.13	4.5 5.3.2
40	Schnittstellen zur Integration weiterer ETS's	3.13, 3.14, 3.15	4.5
41	Funktionalität zur Erstellung neuer Testklassen	3.13	4.5
42	Ermöglichung von Abhängigkeiten zwischen Konformitätsklassen innerhalb einer Testklasse	3.13, 3.14, 3.15	4.5
43	Ermöglichung von Tests von Ressourcen bis zu 5 GB Größe	3.9	4.4
44	Funktionalität zum Anlegen neuer Testkonfigurationen	3.6	5.3.2
45	Funktonalität zur Auflistung aller Testkonfigurationen eines Benutzerkontos	3.7	5.3.1
46	Funktionalität zum Ausführen von Testkonfigurationen	3.9	5.3.1
47	Funktion zum Abbrechen gestarteter Testdurchläufe	3.9	5.3.1
48	Anlegen und Ausführen von Testkonfigurationen nur mit vorgegebenen Werten für Abfrageintervalle ohne Zustimmung des Dienstbetreibers	3.7, 3.8, 3.9	5.3.2

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
49	Ermöglichung des Einstellens von Testwiederholungsintervallen in der Testkonfiguration für registrierte Benutzer und mit Zustimmung des Dienstbetreibers	3.7, 3.8, 3.9	5.3.2
50	Funktionalität für den Fachadministrator zur Visualisierung von Kennzahlen	3.17	5.3.2
51	Abmeldung vom System ohne Abbruch gestarteter Testdurchläufe	3.9	4.5.1 5.3.1
52	Auslieferung von Testberichten als komma-separierte Dateien (*.csv)	3.10	4.9 5.3.3
53	Auslieferung von Testberichten als PDF-Dokument (*.pdf)	3.10	4.9 5.3.3
54	Auslieferung von Testberichten als XML-Dokument	3.10	4.9 5.3.3
55	Testberichte enthalten formatunabhängig immer den gleichen Informationsgehalt	3.10	4.9 5.3.3
56	Testberichte sind immer gleich aufgebaut	3.10	4.9 5.3.3
57	Fehlermeldungen in Testberichten sind immer nach dem gleichen Muster aufgebaut.	3.10	4.5
58	Einstellmöglichkeiten zur Auswahl des Inhaltes von zu erzeugenden Testberichten	3.10, 3.11	5.3.4
59	Funktionalität zur Auflistung aller Testberichte eines Benutzerkontos	3.10, 3.11	5.3.3
60	Schnelltestmodus mit verringertem Funktionsumfang	3.9	5.2.6
61	Erreichbarkeit des Schnelltestmodus' über die Startseite	3.9	5.2.6
62	vollständige und korrekte Implementierung des Rollenkonzeptes	3.2, 3.3, 3.4	4.8

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
63	Verschlüsselung der gespeicherten Zugangsdaten	3.3, 3.4, 3.13, 3.14, 3.15	4.7
64	verlustfreie Übernahme der Daten des bestehenden Systems	3.2	4.5.2
65	Nutzung des DBMS PostgreSQL		4.2
66	sprachwahlabhängige Anzeige der Testergebnisse		5.1
67	vollständige Unterstützung für Mehrsprachigkeit		5.1
68	Konfigurationsmechanismus für die Pflege der Übersetzungen	3.18	4.3
69	Monitoringkonzept für die Komponenten der GDI-DE Testsuite	3.16, 3.19	Siehe Betriebshandbuch
70	Logging-Komponente, die menschenlesbare Logs erzeugt	3.16, 3.19	Siehe Betriebshandbuch
71	mehrstufiges Log-System, das alle relevanten Verarbeitungsschritte protokolliert	3.16, 3.19	Siehe Betriebshandbuch
72	Fehlermeldungen bei Testausführung müssen eindeutiges Ableiten der Fehlerursache ermöglichen	3.16, 3.19	4.5
73	Alle eingesetzten Softwarepakete besitzen eine Open Source Lizenz.		3.1
74	Das System ist plattformunabhängig.		3.1
75	Ermöglichung des serverseitigen Betriebs mit einem Standardwebserver		3.1
76	Entwicklung mit verbreiteten Programmiersprachen und Frameworks		3.1
77	Kommunikation nach außen über HTTPS Port 443 sowie SMTP Port 25		4.6
78	Absicherung des vollständigen Funktionierens in den Browser-Versionen: IE ab V10, aktuelle Versionen von Firefox, Chrome und Safari		5.1
79	Erfüllung der Vorgaben des SAGA-Moduls „Technische Spezikationen“ de.bund 5.0.0, Kap. 6 hinsichtlich aktiver Inhalte und JavaScript		Nicht mehr relevant
80	fehlerfreie Verarbeitung und Darstellung von UTF-8 Daten		3.1

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
81	Treffen und hinreichende Dokumentation von Maßnahmen zur Höherskalierung der gleichzeitigen Testausführungen sowie der Größe von zu testenden Ressourcen		4.4
82	Erfüllung der Anforderungen an Barrierefreiheit gem. SAGA-Modul „Technische Spezifikation“ de.bund 5.0.0, Kapitel 7		5.1
83	Der vollständige Funktionsumfang ist mit Unit-, System- und Integrationstests zu versehen.		Siehe Prüfspezifikation Software
84	Umsetzung der Bausteine und Maßnahmen gem. zum Ausschreibungszeitraum aktuellen BSI IT-Grundschutz für den Schutzbedarf „normal“		Siehe Sicherheitskonzeption
85	Durchführung gemäß V-Modell XT 2.2		Siehe Projekthandbuch
86	Konzept zur Entwicklungsstrategie für das System		Siehe Projekthandbuch
87	SOAP-Schnittstelle	3.2, 3.6, 3.7, 3.9, 3.10, 3.11, 3.13, 3.14, 3.15	Nicht mehr relevant

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
88	REST-Schnittstelle	3.2, 3.6, 3.7, 3.9, 3.10, 3.11, 3.13, 3.14, 3.15	7
89	Erfüllung der Anforderungen an eine diensteorientierte Architektur gem. SAGA-Modul „Technische Spezifikation“ de.bund 5.0.0, Kapitel 5		4.1
90	Dokumentation des Quellcodes in ausreichender Qualität		3.1
91	Darlegung der Quellcode-Dokumentation im Feinkonzept		3.1
92.1	Bereitstellung automatisierter Verfahren für das Deployment von geändertem Quellcode in Form von Softwarepaketen in die Staging- und die Produktivumgebung		Siehe Implementierungs-, Integrations- und Prüfkonzeption
92	Einstellbarkeit der Basisparameter des Systems mittels einer Datei außerhalb des Quellcodes	3.18	4.3
93	Das System muss auf verbreiteten, zukunftssicheren und nachhaltigen Technologien basieren.		3.1
94	modularer Aufbau und Kompatibilität mit dem Microservices-Architekturmuster		4.1
95	20 Testausführungen gleichzeitig, darüber Fehlermeldung		3.2
96	24/7 95% Betrieb, Designentscheidungen im Systementwurf begründen		3.2
97	Vorgaben GDI-DE Technik 3.4.0 werden eingehalten		3.2
98	Monitoringsystem für Staging- und Produktivsystem	3.19	Siehe Betriebshandbuch
99	Erfüllung der Anforderungen an eine einfache funktionale Erweiterbarkeit		4.5

Nr.	Kurzbeschreibung	Use-case	Referenz-Kapitel
100	Betrieb in HA-Umgebung, Anforderungen in Systementwurf spezifizieren		3.2, siehe Betriebs-handbuch

Tabelle 1: Anforderungsliste

11 Glossar

Begriff	**Erläuterung**
Abstract Test Suite (ATS)	Als Abstract Test Suite (ATS) wird eine Beschreibung von Tests zur Überprüfung von Anforderungen eines Standards bezeichnet. ATS sind bereits so formuliert, dass sie durch eine technische Implementierung als Executable Test Suite (ETS) umgesetzt werden können.
Assistent	Der Begriff Assistent bezeichnet im Projektkontext eine Weboberfläche, mittels derer ein Anwender durch mehrere Dialoge für eine ergonomische Dateneingabe geführt wird
Atom Feed	Atom ist ein Standard zum plattformunabhängigen Austausch von Informationen. Ein Atom Feed stellt dabei eine Instanz einer solchen Informationssammlung dar. Atom Feeds können im Rahmen der INSPIRE Richtlinie für die Implementierung von INSPIRE Download-Diensten genutzt werden.
Basisparameter	Der Begriff Basisparameter bezeichnet im Kontext der GDI-DE Testsuite alle Konfigurationen, die für die Skalierung und Verteilung der GDI-DE Testsuite im Betrieb relevant sind.
Container	Containervirtualisierung (auch: Betriebssystemvirtualisierung) ist eine Methode, um mehrere Instanzen eines Betriebssystems (als sog. „Gäste“) isoliert voneinander auf einem Hostsystem zu betreiben.
Einzeltest	Einzeltests bestehen aus Testschritten. Ein Einzeltest beschreibt den Vorgang zur Überprüfung von genau einer Anforderung eines Standards. Mehrere Einzeltests bilden zusammen Konformitätsklassen
Executable Test Suite (ETS)	Als Executable Test Suite (ETS) wird die Implementierung einer ATS bezeichnet. Eine ETS kann durch verschiedene technische Lösungen implementiert werden, die wiederum durch spezifische Testengines ausgeführt werden können.
GDI-DE	Mit GDI-DE wird die Geodateninfrastruktur in Deutschland bezeichnet. Diese besteht aus den Geodateninfrastrukturen des Bundes und der Länder. Die GDI-DE Testsuite ist eine der zentralen Komponenten der GDI-DE.

Begriff	**Erläuterung**
HA-Cluster	Hochverfügbarkeitscluster (engl. High-Availability-Cluster – HA-Cluster) werden zur Steigerung der Verfügbarkeit bzw. für bessere Ausfallsicherheit eingesetzt. Tritt auf einem Knoten des Clusters ein Fehler auf, werden die auf diesem Knoten laufenden Dienste auf einen anderen Knoten migriert. Sowohl die Hardware als auch die Software eines HA-Clusters muss frei von Single-Point-of-Failures (Komponenten, die durch einen Fehler das gesamte System zum Ausfall brächten) sein.
INSPIRE	Die Infrastructure for Spatial Information in the European Community (INSPIRE) beschreibt den gesetzlichen Rahmen zur Umsetzung einer Geodateninfrastruktur in Europa. Die Vorgaben von INSPIRE werden in Deutschland durch die GDI-DE umgesetzt.
JSON	Die JavaScript Object Notation, kurz JSON, ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen.
Konformitätsklasse	Eine Konformitätsklasse bildet einen Teilbereich einer Testklasse ab. Sie ist eine Zusammenstellung von Einzeltests, die bestanden werden müssen, damit die getestete Ressource der Konformitätsklasse entspricht.
Linter	Ein Linter ist eine Software zur statischen Code-Analyse.
Microservices	Microservices sind ein Architekturmuster der Informationstechnik, bei dem komplexe Anwendungssoftware aus unabhängigen Prozessen komponiert wird, die untereinander mit sprachunabhängigen Programmierschnittstellen kommunizieren. Die Dienste sind weitgehend entkoppelt und erledigen eine kleine Aufgabe. So ermöglichen sie einen modularen Aufbau von Anwendungssoftware.
OGC Catalog Service for the Web (CSW)	Ein OGC CSW ist ein Katalogdienst, der zum Suchen und Finden von Datensätzen und Diensten anhand der im Katalogdienst eingebundenen Metadaten geeignet ist.
OGC Compliance Testing Language (CTL)	Die OGC CTL ist eine Auszeichnungssprache zur technischen Umsetzung von Tests. Es können einzelne Testschritte implementiert werden und Einzeltests zu Konformitätsklassen zusammengefasst werden.

Begriff	**Erläuterung**
OGC TEAM Engine	Die OGC TEAM Engine ist eine Testengine, die vom OGC genutzt wird, um Software-Implementierungen auf Konformität zu den Standards des OGC zu überprüfen. Die OGC TEAM Engine nutzt vornehmlich in OGC CTL implementierte ETS. Darüber hinaus gibt es Schnittstellen zum Einbinden von TestNG-basierten ETS.
OGC Web Feature Service (WFS)	Ein OGC Web Feature Service ist ein Geodienst, der die Bereitstellung von Geodaten im GML Format vorsieht und durch das OGC spezifiziert wurde. Web Feature Services bieten GET- und POST-Schnittstellen zum Abfragen von Geodaten.
OGC Web Map Service (WMS)	Ein OGC Web Map Service ist ein Geodienst, der der Bereitstellung von Karten dient und durch das OGC spezifiziert wurde. Web Map Services bieten GET- und POST-Schnittstellen zum Abfragen von Karten
Open Geospatial Consortium (OGC)	Das Open Geospatial Consortium (OGC) ist eine gemeinnützige Organisation, die sich zum Ziel gesetzt hat, die Entwicklung von raumbezogener Informationsverarbeitung auf Basis allgemeingültiger, offener Standards zum Zweck der Interoperabilität festzulegen.
Open Source	Als Open Source wird Software bezeichnet, deren Quelltext öffentlich und von Dritten eingesehen, geändert und genutzt werden kann.
Rainbow Table	Die Rainbow Table (engl. für Regenbogentabelle) ist eine Datenstruktur, die eine schnelle, speichereffiziente Suche nach der ursprünglichen Zeichenfolge (in der Regel ein Passwort) für einen gegebenen Hashwert ermöglicht.
Ressource	Im Kontext der GDI-DE Testsuite bezeichnet eine Ressource einen Web-Dienst oder eine Datei, die anhand der vorhandenen Testklassen mittels der GDI-DE Testsuite überprüft werden kann.
Reverse Proxy	Der Reverse Proxy holt Ressourcen für einen Client von einem oder mehreren Servern. Die Umsetzung der Adresse ist atypisch und der Richtung des Aufrufes entgegengesetzt (deutsch „umgekehrter Proxy“). Die wahre Adresse des Zielsystems bleibt dem Client verborgen. Das unterscheidet ihn vom typischen Proxy, der mehreren Clients eines proprietären (in sich abgeschlossenen) Netzes den Zugriff auf ein externes Netz gewährt.

Begriff	**Erläuterung**
SAGA	Die Standards und Architekturen für E-Government-Anwendungen (SAGA) sind eine Sammlung verpflichtender Konventionen bei der Umsetzung von ITK-Projekten in der öffentlichen Verwaltung.
Service Provider Interface	Service Provider Interface (SPI) ist eine API, die von Dritten implementiert oder erweitert werden kann. Damit können Komponenten erweitert und ausgetauscht werden.
Testklasse	Testklasse bezeichnet die Gesamtheit aller Konformitätsklassen, die zum Testen der Konformität zu einem bestimmten Standard notwendig sind.
Testkonfiguration	Mit Testkonfiguration wird eine Konfiguration für Testausführungen zu einer oder mehrerer spezifischen Ressource(n) bezeichnet und umfasst eine Testzusammenstellung bestehend aus einer oder mehreren Konformitätsklassen einer Testklasse.
Testschritt	Ein Testschritt ist die kleinste Einheit einer Einzeltestspezifikation. Ein Einzeltest besteht aus mindestens einem Testschritt. Testschritte werden in einer ATS beschrieben und in einer ETS technisch implementiert.
XML	Die Extensible Markup Language (dt. Erweiterbare Auszeichnungssprache), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten im Format einer Textdatei, die sowohl von Menschen als auch von Maschinen lesbar ist.
XSRF	Eine Cross-Site-Request-Forgery (meist CSRF oder XSRF abgekürzt) ist ein Angriff auf ein Computersystem, bei dem der Angreifer eine Transaktion in einer Webanwendung durchführt.
XSS	Cross-Site-Scripting (XSS) bezeichnet das Ausnutzen einer Computersicherheitslücke in Webanwendungen, indem Informationen aus einem Kontext, in dem sie nicht vertrauenswürdig sind, in einen anderen Kontext eingefügt werden, in dem sie als vertrauenswürdig eingestuft werden.