Technische Universität Braunschweig

PLRI PETER L. REICHERTZ INSTITUT FÜR MEDIZINISCHE INFORMATIK

AAL on FHIR

Projekt ARGOS

Software-Entwicklungspraktikum (SEP)
Sommersemester 2015

Testspezifikation

Auftraggeber
Technische Universität Braunschweig
Peter L. Reichertz Institut für Medizinische Informatik
Prof. Dr. Reinhold Haux
Mühlenpfordtstraße 23
38106 Braunschweig

Betreuer: Jonas Schwartze, M.Sc.

Auftragnehmer:

Name	E-Mail-Adresse
Angelina Céline Korell	a.korell@tu-braunschweig.de
Fabian Wohninsland	f.wohninsland@tu-braunschweig.de
Florian Bahr	f.bahr@tu-braunschweig.de
Nora Krogoll	n.krogoll@tu-braunschweig.de
Rafael Fornal	r.fornal@tu-braunschweig.de

Braunschweig, 15. Juli 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Tests sind im Entwicklungsprozess eines Softwareprodukts ein unverzichtbares Werkzeug, um die Implementierung des Produkts gegen ihre Spezifikation zu verifizieren und die Realisierung gestellter Qualitätsanforderungen, die Aspekte der Funktionalität, Benutzbarkeit, Änderbarkeit und Sicherheit umfassen, zu belegen. Voraussetzung für die Erbringung eines solchen Nachweises von Korrektheit sowie Funktionalität und Benutzbarkeit des Softwareprodukts ist eine sorgfältige Planung und Dokumentation der Testprozesse und abzudeckenden Testfälle.

Das vorliegenden Dokument spezifiziert sowohl *Integrations-* und *Modultests*, die der Hersteller der Webanwendung Argos begleitend zum Entwicklungsprozess durchgeführt hat, um Korrektheit und spezifikationsgemäße Funktionalität der Implementierung zu überprüfen, als auch die *Abnahmetests*, die der Kunde bzw. Auftragsgeber mit dem Ziel durchführt, Funktionalitäten des gelieferten Produkts gegen seine Anforderungen zu validieren.

Da Auswahl von Testszenarien und Ausgestaltung der Testprozesse maßgeblich durch die dokumentierten Kundenanforderungen wie auch durch Architekturmerkmale der zu entwickelnden Anwendung determiniert wird, werden wesentliche Charakteristika des Projekts Argos nachstehend noch einmal zusammengefasst:

Argos ist eine Webanwendung, die die Versorgung geriatrischer Patienten durch Bereitstellung visuell aufbereiteter Informationen über allgemeine Verhaltensmuster eines Patienten und seine aktuelle Situierung unterstützen soll. Die Grundlage der Anwendung bilden Sensordaten, die aus passiv-assistierenden Gesundheitstechnologien (Erschütterungsdetektoren, Bewegungsmeldern, u. ä.) gewonnen werden. Argos besteht aus fünf interoperierenden Software-Komponenten, die im Wesentlichen in zwei Kategorien unterteilt werden können: einem *Backend* aus Web-Services und Datenhaltungskomponente, die die spezifizierten Produktfunktionen realisieren, und einem *Frontend* in Form einer responsiven Webschnittstelle.

Dem *Frontend* fallen die Aufgaben der Entgegennahme, Überprüfung, Aufbereitung und Umsetzung von Nutzereingaben sowie der Präsentation von durch *Backend*-Modulen realisierten Produktfunktionen zu. Dabei ist die Benutzerschnittstelle einfach und intuitiv bedienbar zu gestalten, um die Einarbeitungszeit für Endanwender gering zu halten.

Die Kapselung von Produktfunktionen in eigenständigen Web-Services führt zu einer modularen Systemarchitektur, die eine hohe Modifizier- und Erweiterbarkeit gewährleistet. Zuverlässigkeit

und Sicherheit der Anwendung sollten den Anforderungen genügen, die üblicherweise von konzeptionell vergleichbaren Produkten erwartet werden. D. h. unter anderem, dass die Bereitstellung von Produktfunktionen und Patientendaten nur unter der Prämisse erfolgter Authentifizierung und vorliegender Autorisierung erfolgt. Als Webanwendung ist Argos für einen Dauerbetrieb konzipiert. Somit ist ein zuverlässiger Betrieb über längere Zeit sicherzustellen.

Der weitere Aufbau dieses Dokuments orientiert sich am IEEE-Standard 829-2008 für Software-Testdokumentation. Zunächst werden in einem Testplan zu testende Komponenten, Funktionen und Merkmale identifiziert und anschließend adäquate Testfälle für die Teststufen *Abnahme-*, *Integrations-* und *Modultest* festgelegt.

2 Testplan

In diesem Kapitel wird der Testplan für die Durchführung der einzelnen Teststufen definiert. Zu diesem Zweck werden zunächst die zu testenden Komponenten und Produktfunktionen bestimmt und anschließend Vorgehen und Umfang der Maßnahmen während der einzelnen Teststufen sowie die Ablaufumgebung, in der die Tests durchgeführt werden, beschrieben.

2.1 Zu testende Komponenten

In direkter Bezugnahme auf die technische Entwurfsdokumentation wird in diesem Dokument zwischen folgenden Programmkomponenten unterschieden: der *grafischen Benutzerschnittstelle* (ArgosGUI) ⟨C010⟩, die als *Frontend*-Komponente der Entgegennahme von Nutzereingaben sowie der Präsentation von Realisierungen von Produktfunktionen dient, und den Web-Services *Identitätsmanagementdienst* ⟨C020⟩, *Visualisierungsdienst* ⟨C030⟩ und *Benachrichtigungsdienst* ⟨C040⟩, die im Verbund mit dem durch den Kunden betriebenen FHIR-Server ⟨C050⟩ das *Backend* der Anwendung bilden.

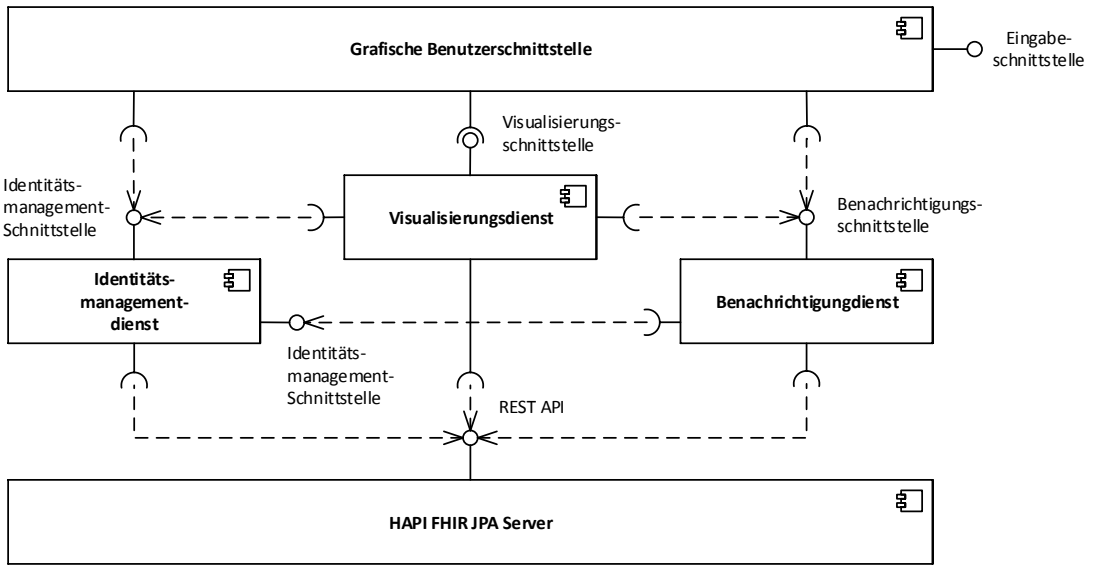

Abbildung 2.1: Komponentendiagramm der Webanwendung Argos

Mit Ausnahme der Komponente ⟨C050⟩ (vgl. hierzu auch Abschnitt 2.3), sind alle voranstehend aufgelisteten Programmbestandteile zum Gegenstand von Tests zu machen.

Die zu testenden Produktkomponenten werden als kompilierbare Java-Quelltexte (ausgeliefert mit den für einen automatisierten Erstellungsprozess erforderlichen Build-Dateien) sowie als präkompilierte *Web Application Archives* (WAR) verfügbar gemacht.

Java-Quelltexte und Erstellungsdateien werden im SVN-Repository des vom Kunden für die Projektorganisation betriebenen Redmine-Projektmanagementportals hinterlegt. Parallel dazu werden die Kompilate im Download-Bereich desselbigen Portals bereitgestellt.

Nr	Komponenten	Version	Auslieferungsform
1	⟨C010⟩ ArgosGUI	1.0	WAR, Quelltext
2	⟨C020⟩ Identitätsmanagementdienst	1.0	WAR, Quelltext
3	⟨C030⟩ Visualisierungsdienst	1.0	WAR, Quelltext
4	⟨C040⟩ Benachrichtigungsdienst	1.0	WAR, Quelltext

Tabelle 2.1: Zu testende Komponenten

2.2 Zu testende Funktionen/Merkmale

Die Erbringung bzw. Erfüllung folgender Funktionen und Qualitätsmerkmale ist durch Tests zu belegen:

- Zugriffskontrolle ⟨F010⟩
- Patient auswählen ⟨F020⟩
- Visualisierunsart wählen ⟨F030⟩
- Tagesverlauf anzeigen ⟨F040⟩
- Circle-Plot anzeigen ⟨F050⟩
- Live-Bild anzeigen ⟨F060⟩
- Heatmap anzeigen ⟨F070⟩
- Abmelden ⟨F080⟩

- Responsivität[1] der Anwendung ⟨Q10⟩
- Betriebssystemunabhängigkeit der Anwendung ⟨Q20⟩
- Intuivitäet der Anwendung ⟨Q30⟩
- Robustheit der Anwendung gegenüber fehlerhaften Benutzereingaben ⟨Q40⟩

[1] Unter der Prämisse des Bestehens einer bzgl. der Durchsatzrate mit den in Abschnitt 2.5 formulierten Mindestanforderungen konformen TCP/IP-Verbindung sollte bei Interaktion mit Nutzern eine Antwortzeit von 5 Sekunden nicht überschritten werden; im Hinblick auf berechnungsintensive Produktfunktionen, füer die eine Einhaltung dieses Zeitlimits nicht per se garantiert werden kann, bedeutet dies, das der Nutzer in geeigneter Weise üeber Fortschritt bzw. Andauern der Realisierung der Produktfunktion zu informieren ist.

2.3 Nicht zu testende Funktionen

Die spezifikationsgemäße Funktionsfähigkeit der Komponenten der Betriebsumgebung (d. h., der als *Host* der einzelnen Serverkomponenten dienende Computer und die als *Client* operierenden Endgeräte, inklusive der jeweils auf ihnen installierten Betriebssysteme) sowie der für die Vorbereitung und Durchführung des operativen Betriebs erforderlichen Anwendungsprogramme (z. B. Web-Container und Web-Browser), Programmierwerkzeuge und -bibliotheken von Drittanbietern wird grundsätzlich vorausgesetzt, und insofern nicht gesondert getestet.

Diese Ausschlusserklärung erfasst insbesondere:

- Oracle Java SE 8 Development Kit
- Apache Tomcat 8 Web-Container
- Vaadin 7.5 Webanwendungs-Framework
- Google Guava 18.0 EventBus
- JAX-RS 2.0 – Java API for RESTful Services
- HAPI FHIR JPA Server (Version 1.1) und Programmierschnittstelle (Version 1.0)
- JUnit 4-Testsuite und
- Maven 3.3-Erstellungswerkzeug

2.4 Vorgehen

Dieser Abschnitt beschreibt die grundsätzliche Vorgehensweise bei der Durchführung der Tests der einzelnen Funktionen des Produkts und ihrer Kombinationen und definiert, welche Aktivitäten, Techniken und Werkzeuge jeweils für die Tests eingesetzt werden.

Der Testablauf richtet sich nach den Phasen des V-Modells. Das heißt, dass zunächst die einzelne Programmkomponenten in *Modultests* separat getestet werden. Darauf folgend werden *Integrationstests* durchgeführt, die das spezifikationsgemäße Zusammenwirken der Komponenten im Verbund überprüfen. Anschließend wird üblicherweise im Rahmen eines Systemtests das Gesamtsystem gegen die Gesamtheit der im Pflichtenheft vereinbarten funktionalen und nichtfunktionalen Anforderungen getestet. Abgeschlossen wird der Testzyklus mit der Durchführung des Abnahmetests, bei dem das Produkt durch den Kunden bzw. Auftraggeber unter Vertragserfüllungs- und Akzeptanzgesichtspunkten bewertet wird.

Die einzelnen Tests im Rahmen des Projekts Argos werden wie folgt unterschieden:

- **Komponententest**
  Ziel dieser Teststufe ist der Nachweis der technischen Lauffähigkeit und Korrektheit der fachlichen Ergebnisse. Dabei werden die Komponenten, so weit dies möglich ist, voneinander isoliert betrachtet. Ein weiterer Testaspekt sind die nichtfunktionalen Anforderungen wie z.B. Richtigkeit, die ebenfalls getestet werden müssen. Durch Verwendung von Black-Box-Tests werden dabei Methoden, Klassen, Funktionen und Module auf funktionaler Ebene getestet. Mögliche Fehler müssen noch vor dem Integrationstest behoben werden.
  Sämtliche Komponenten (mit Ausnahme der ArgosGUI ⟨C010⟩) werden unter Verwendung von JUnit-Testfällen geprüft.

- **Integrationstest**
  Nach Fertigstellung und Test der einzelnen Komponenten des Systems, wird ihr Zusammenwirken getestet. Der Integrationstest überprüft die Vereinbarkeit der verschiedenen Komponenten untereinander. Die Komponenten werden dabei, unter Berücksichtigung ihrer Abhängigkeiten, in einer konkreten Reihenfolge getestet und integriert.
  Als Testwerkzeug kommt auch an dieser Stelle JUnit zum Einsatz.

- **Systemtest**
  Während des Systemtests wird die vollständige Erfüllung der funktionalen sowie der nichtfunktionalen Anforderungen getestet. Es wird ein realistischer Betrieb der Software simuliert.

- **Abnahme- und Funktionstests**
  Der Abnahmetest dient dazu, die vom Kunden an den Auftragsnehmer gestellten und im Pflichtenheft dokumentierten Anforderungen an das Endprodukt auf ihre Erfüllung zu testen. Hierzu sind dem Kunden alle Funktionen des vollständigen Produkts vorzustellen und alle Anforderungsfälle aus der Anforderungsspezifikation auf ihre Funktionalität hin zu überprüfen.
  Zu diesem Abnahmetests wird das Black-Box-Verfahren angewendet, d. h. der Test orientiert sich nicht am Code der Software, sondern nur an ihrem Verhalten bei spezifizierten Situationen oder Handlungen, kann also direkt auf der grafischen Benutzerschnittstelle der Anwendung auf Richtigkeit überprüft werden.

2.5 Testumgebung

Im Folgenden werden die zur Durchführung der in diesem Dokument definierten Tests eingesetzten Softwarewerkzeuge sowie die Charakteristika der Testumgebung näher beschrieben.

Für eine umfassende Testdurchführung werden mindestens drei Endgeräte unterschiedlicher Geräteklassen (Desktop-PC, Tablet-Computer und Smartphone) benötigt, die als *Clients* operieren.

Als *Host* der Serverkomponenten des Softwaresystems dient ein Computer mit Java 8 zertifizierter Systemkonfiguration[2], der mit einem *Oracle Java SE 8 Development Kit*, einem *Apache Tomcat 8*-Web-Container und einer *Maven 3.3*-Installation vorkonfiguriert ist.

Weitere für die Testdurchführung erforderliche Programmbibliotheken – dabei sei insbesondere die *JUnit 4*-Test-Suite erwähnt – werden über ein entsprechendes, der Distribution des Endprodukts beigefügtes Konfigurationsskript nachinstalliert.

Ferner wird vorausgesetzt, dass zwischen Host (Dienstrechner) und Clients (Endgeräten) eine TCP/IP-Netzwerkverbindung besteht, wobei speziell im Fall mobiler Endgeräte eine Mindestübertragungsrate von 14,4 MBit/s (gm. HSPA+[3]-Standard) vorausgesetzt wird.

Im Rahmen des Abnahmetests werden die Service-Komponenten der Webanwendung in einer Ablaufumgebung installiert, die der zukünftigen Produktivumgebung entspricht.

[2] vgl. `http://www.oracle.com/technetwork/java/javase/certconfig-2095354.html`
[3] vgl. `https://de.wikipedia.org/wiki/High_Speed_Packet_Access`

3 Abnahmetest

Der *Abnahmetest* bildet die letzte Teststufe des Softwaresystems und ist ein Test des Produkts aus Benutzersicht. Ziel ist es, dem Kunden bzw. Auftraggeber zu belegen, dass das gelieferte Endprodukt vereinbarte Funktionalitäten vollständing und in gewünschter Weise erbringt, so dass das Produkt freigegeben werden und die Abnahme erfolgen kann.

Bei diesem aufgabenorientierten Test stehen die Ergebnisse des Zusammenwirkens der Funktionen des Produkts im Vordergrund. Im Zuge des Abnahmetests wird die Erfüllung der im Pflichtenheft an das Endprodukt festgelegten Anforderungen überprüft, und zwar sowohl unter den Gesichtspunkten vertraglicher Akzeptanz, d. h. objektive Erbringung vereinbarter Leistungen, wie auch Benutzerakzeptanz, d. h. Erbringung der Leistungen in einer dem Kundenverständnis entsprechenden Weise.

Für die Zwecke dieser Teststufe ist das Produkt in einer praxisnahen Abnahmeumgebung des Kunden zu installieren und, in Vorbereitung der Begutachtung, ist der Kunde mit alle Funktionen des finalisierten Produkts vertraut zu machen.

3.1 Zu testende Anforderungen

Im Verlauf des Abnahmetests testet der Kunde bzw. Auftraggeber alle Funktionen, die Nutzer der Webanwendung im Produktivbetrieb verwenden. Dies sind im Einzelnen:

Nr	Anforderung	Testfälle	Kommentar
1	⟨F10⟩ Zugriffskontrolle	⟨T010⟩	Authentifizierung und Autorisierung des Nutzers.
2	⟨F10⟩ Patient auswählen	⟨T020⟩	Präsentation einer Patientenauswahlliste.
3	⟨F30⟩ Visualisierungsart wählen	⟨T030⟩	Präsentation einer Visualisierungsartenauswahl.
4	⟨F40⟩ Tagesverlauf anzeigen	⟨T040⟩	Nutzer wählt die Option *Balkendiagramm*.
5	⟨F50⟩ Circle-Plot-Diagramm anzeigen	⟨T050⟩	Nutzer wählt die Option *Circle-Plot*.

Fortsetzung umseitig

Nr	Anforderung	Testfälle	Kommentar
6	⟨F60⟩ Live-Bild anzeigen	⟨T060⟩	Nutzer wählt die Option *Live-Bild*.
7	⟨F70⟩ Heatmap-Diagramm anzeigen	⟨T070⟩	Nutzer wählt die Option *Heatmap*.
8	⟨F80⟩ Abmelden	⟨T080⟩	Nutzer wählt die Option *Abmelden*.

Tabelle 3.1: Zu testende Anforderungen

3.2 Testverfahren

Im Rahmen des Abnahmetests werden ausschließlich Black-Box-Tests durchgeführt.

Auf Eingaben über die vom Endnutzer ansprechbare Schnittstelle (ArgosGUI) oder einen über die Anwenderoberfläche gesteuerten Programmablauf muss die Webanwendung in entsprechender Weise reagieren und korrekte Ergebnisse erzielen.

Zu diesem Zweck werden die Produktfunktionen der Webanwendung durch die in Abschnitt definierten Testfälle in der Reihenfolge ihrer Auflistung systematisch getestet und das beobachtete Verhalten mit dem im Pflichtenheft in Form funktionaler Anforderungen dokumentierten erwarteten Verhalten verglichen.

Die dieser Teststufe zugeordneten Testfälle sind dabei für jede clientseitig verwendete Geräteklasse (Desktop-PC, Tablet-Computer, Smartphone) vollständig zu durchlaufen.

3.2.1 Testskripte

In dieser Teststufe werden keine automatisierten Testskripte verwendet.

3.3 Testfälle

Es werden die folgenden Abnahmetestfälle betrachtet:

3.3.1 Testfall ⟨T000⟩ - Inbetriebnahme der Webanwendung

Ziel

Vorbereitung der Durchführung des Abnahmetests.

Objekte/Methoden/Funktionen

Komponenten:

- Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩
- Identitätsmanagementdienst ⟨C020⟩
- Visualisierungsdienst ⟨C030⟩
- Benachrichtigungsdienst ⟨C040⟩

Pass/Fail-Kriterien

Pass: Start- und Funktionstestskript terminiert ohne Fehler, Webanwendung ist in Dienst gestellt

Fail: Start- und Funktionstestskript terminiert in Fehlerzustand

Vorbedingung

Bereitstellung einer mit den Anforderungen aus Abschnitt 2.5 konformen Testumgebung

Einzelschritte

Eingabe: Aufruf eines vorbereiteten Start- und Funktionstestskripts

Ausgabe: Benachrichtigung (Kommandozeilenausgabe), dass das Initialisierungs- und Startskript ohne Fehler beendet und die Webanwendung in Dienst gestellt wurde
Alternative Ausgabe: Benachrichtigung (Kommandozeilenausgabe), dass die Startprozedur oder ein Funktionstest fehlgeschlagen sind

Beobachtungen/Log/Umgebung

Die Ergebnisse der begleitend zur Indienststellung der Serverdienste durchgeführten Funktionstests werden in einer Logdatei protokolliert.

3.3.2 Testfall ⟨T010⟩ - Anmelden an der ArgosGUI

Ziel

Überprüfung des Normalfalls: Authentifizierung und Autorisierung eines registrierten Nutzers nach Bereitstellung gültiger Anmeldedaten.

Objekte/Methoden/Funktionen

Komponenten:
- Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩
- Identitätsmanagementdienst ⟨C020⟩

Funktion: Zugriffskontrolle ⟨F010⟩

Pass/Fail-Kriterien

Pass: Nutzer authentifiziert und autorisiert,
Patientenauswahldialog wird angezeigt

Fail: Anmeldung schlägt fehl,
Ausgabe einer Fehlermeldung

Vorbedingung

Nutzer ist registriert

Einzelschritte

Eingabe:
1. Benutzernamen eingeben
2. Passwort eingeben
3. Dialogoption *Anmelden* wählen

Ausgabe: Der Nutzer wird weitergeleitet zum Patientenauswahldialog.

Alternative Eingabe: wie oben, aber fehlerhafte Eingabe von Benutzernamen oder Passwort.

Alternative Ausgabe: Der Nutzer wird informiert, dass die eingegebenen Anmeldedaten fehlerhaft sind und der Anmeldevorgang wiederholt werden muss.

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

Der Testfall ist kausal von Testfall *Indienststellung der Webanwendung* ⟨T000⟩ abängig.

3.3.3 Testfall ⟨T020⟩ - Patient auswählen

Ziel

Überprüfung des Normalfalls: Auswahl eines Patienten aus einer Auswahlliste, Anzeige einer Zusammenstellung von Visualisierungsarten, die der angemeldete Nutzer für ausgewählten Patienten abzurufen autorisiert ist.

Objekte/Methoden/Funktionen

Komponente: Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩

Funktion: Patient auswählen ⟨F020⟩

Pass/Fail-Kriterien

Pass: Visualisierungsartenauswahldialog wird angezeigt

Fail: Dialogoption *Weiter* wird nicht freigeschaltet

Fail: Visualisierungsartenauswahldialog wird nicht angezeigt

Fail: Visualisierungsartenauswahldialog wird angezeigt, aber nicht alle Visualisierungsarten angeboten, die der angemeldete Nutzer für gewählten Patienten zu nutzen autorisiert wäre.

Vorbedingung

Nutzer ist registriert und angemeldet.

Einzelschritte

Eingabe:

1. Auswahl eines Patienten aus Auswahlliste
2. Dialogoption *Weiter* wählen

Ausgabe: Der Nutzer wird weitergeleitet zum Visualisierungsartenauswahldialog.

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

Der Testfall ist kausal von den Testfällen *Indienststellung der Webanwendung* ⟨T000⟩ und *Anmelden an der* ArgosGUI ⟨T010⟩ abängig.

3.3.4 Testfall ⟨T030⟩ - Visualisierungsart wählen

Ziel

Überprüfung des Normalfalls: Auswahl einer angebotenen Visualisierungsart, Aufruf der gewählten Produktfunktion

Objekte/Methoden/Funktionen

Komponente: Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩

Funktion: Visualisierungsart wählen ⟨F020⟩

Pass/Fail-Kriterien

Pass: Gewählte Produktfunktion wird aufgerufen

Fail: Gewählte Produktfunktion wird nicht aufgerufen, keine Veränderung

Vorbedingung

Nutzer ist registriert und angemeldet,
Nutzer hat einen Patienten ausgewählt.

Einzelschritte

Eingabe: Auswahl einer Visualisierungsart aus Visualisierungsartenauswahldialog.

Ausgabe: Gewählte Produktfunktion wird aufgerufen.

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

Der Testfall ist kausal von den Testfällen *Indienststellung der Webanwendung* ⟨T000⟩, *Anmelden an der* ArgosGUI ⟨T010⟩ und *Patient auswählen* ⟨T020⟩ abängig.

3.3.5 Testfall ⟨T040⟩ - Tagesverlauf anzeigen

Ziel

Überprüfung des Normalfalls: Aufgezeichnete Sensordaten für ggb. Patienten werden aufbereitet in Form eines Balkendiagramms angezeigt.

Objekte/Methoden/Funktionen

Komponenten:
- Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩
- Identitätsmanagementdienst ⟨C020⟩
- Visualisierungsdienst ⟨C030⟩

Funktion: Tagesverlauf anzeigen ⟨F040⟩

Pass/Fail-Kriterien

Pass: Daten füer gewählten Patienten werden korrekt im Form eines Balkendiagramms aufbereitet angezeigt

Fail: Balkendiagramm wird nicht angezeigt

Fail: Balkendiagramm wird angezeigt, aber Daten sind fehlerbehaftet

Vorbedingung

Nutzer ist registriert und angemeldet,
Nutzer hat einen Patienten ausgewählt.

Einzelschritte

Eingabe: Auswahl der Darstellungsoption *Balkendiagramm*.

Ausgabe: Balkendiagramm wird angezeigt.

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

Der Testfall ist kausal von den Testfällen *Indienststellung der Webanwendung* ⟨T000⟩, *Anmelden an der* ArgosGUI ⟨T010⟩, *Patient auswählen* ⟨T020⟩, und *Visualisierungsart wählen* ⟨T030⟩ abängig.

3.3.6 Testfall ⟨T050⟩ - Circle-Plot-Diagramm anzeigen

Ziel

Überprüfung des Normalfalls: Aufgezeichnete Sensordaten für ggb. Patienten werden aufbereitet in Form eines Circle-Plot-Diagramms angezeigt.

Objekte/Methoden/Funktionen

Komponenten:
- Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩
- Identitätsmanagementdienst ⟨C020⟩
- Visualisierungsdienst ⟨C030⟩

Funktion: Circle-Plot-Diagramm anzeigen ⟨F050⟩

Pass/Fail-Kriterien

Pass: Daten füer gewählten Patienten werden korrekt im Form eines Circle-Plot-Diagramms aufbereitet angezeigt

Fail: Circle-Plot-Diagramm wird nicht angezeigt

Fail: Circle-Plot-Diagramm wird angezeigt, aber Daten sind fehlerbehaftet

Vorbedingung

Nutzer ist registriert und angemeldet,
Nutzer hat einen Patienten ausgewählt.

Einzelschritte

Eingabe: Auswahl der Darstellungsoption *Circle-Plot*.

Ausgabe: Circle-Plot-Diagramm wird angezeigt.

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

Der Testfall ist kausal von den Testfällen *Indienststellung der Webanwendung* ⟨T000⟩, *Anmelden an der* ArgosGUI ⟨T010⟩, *Patient auswählen* ⟨T020⟩, und *Visualisierungsart wählen* ⟨T030⟩ abängig.

3.3.7 Testfall ⟨T060⟩ - Live-Bild anzeigen

Ziel

Überprüfung des Normalfalls: Aktuelle Sensordaten für ggb. Patienten werden auf den Grundriss einer Wohneinheit projiziert angezeigt.

Objekte/Methoden/Funktionen

Komponenten:

- Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩
- Identitätsmanagementdienst ⟨C020⟩
- Visualisierungsdienst ⟨C030⟩
- Benachrichtigungsdienst ⟨C040⟩

Funktion: Live-Bild anzeigen ⟨F060⟩

Pass/Fail-Kriterien

Pass: Aktuelle Sensorereignisse für gewählten Patienten werden korrekt aufbereitet angezeigt

Fail: Grundrissdarstellung ist fehlerhaft

Fail: Anzeige aktueller Sensorereignisse ist fehlerbehaftet

Vorbedingung

Nutzer ist registriert und angemeldet,
Nutzer hat einen Patienten ausgewählt.

Einzelschritte

Eingabe: Auswahl der Darstellungsoption *Live-Bild*.

Ausgabe: Live-Bild wird angezeigt.

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

Der Testfall ist kausal von den Testfällen *Indienststellung der Webanwendung* ⟨T000⟩, *Anmelden an der* ArgosGUI ⟨T010⟩, *Patient auswählen* ⟨T020⟩, und *Visualisierungsart wählen* ⟨T030⟩ abängig.

3.3.8 Testfall ⟨T070⟩ - Heatmap-Diagramm anzeigen

Ziel

Überprüfung des Normalfalls: Aufgezeichnete Sensordaten für ggb. Patienten werden aufbereitet in Form eines Heatmap-Diagramms (projiziert auf Grundriss einer Wohneinheit) angezeigt.

Objekte/Methoden/Funktionen

Komponenten:

- Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩
- Identitätsmanagementdienst ⟨C020⟩
- Visualisierungsdienst ⟨C030⟩

Funktion: Heatmap-Diagramm anzeigen ⟨F070⟩

Pass/Fail-Kriterien

Pass: Daten für gewählten Patienten werden korrekt im Form eines Heatmap-Diagramms aufbereitet angezeigt

Fail: Heatmap-Diagramm wird nicht angezeigt

Fail: Heatmap-Diagramm wird angezeigt, aber Grundrissdarstellung ist

Fail: Heatmap-Diagramm wird angezeigt, aber Daten sind fehlerbehaftet fehlerhaft

Vorbedingung

Nutzer ist registriert und angemeldet,
Nutzer hat einen Patienten ausgewählt.

Einzelschritte

Eingabe: Auswahl der Darstellungsoption *Circle-Plot*.

Ausgabe: Circle-Plot-Diagramm wird angezeigt.

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

Der Testfall ist kausal von den Testfällen *Indienststellung der Webanwendung* ⟨T000⟩, *Anmelden an der* ArgosGUI ⟨T010⟩, *Patient auswählen* ⟨T020⟩, und *Visualisierungsart wählen* ⟨T030⟩ abängig.

3.3.9 Testfall ⟨T080⟩ - Abmelden von der ArgosGUI

Ziel

Überprüfung des Normalfalls: Abmelden eines Nutzers, Entwerten aller erteilten Autorisierungen, Umleitung auf Anmeldeseite

Objekte/Methoden/Funktionen

Komponenten:
- Graphische Benutzerschnittstelle (ArgosGUI) ⟨C010⟩
- Identitätsmanagementdienst ⟨C020⟩

Funktion: Abmelden ⟨F080⟩

Pass/Fail-Kriterien

Pass: Nutzer wird abgemeldet, erteilten Autorisierungen werden entwertet

Fail: Nutzer wird abgemeldet, aber nicht auf Anmeldeseite navigiert

Fail: Nutzer wird abgemeldet, aber Autorisierungen werden nicht entwertet

Fail: Nutzer wird nicht abgemeldet

Vorbedingung

Nutzer ist registriert und angemeldet.

Einzelschritte

Eingabe: Betätigen der Dialogoption *Abmelden*.

Ausgabe: Nutzer wird abgemeldet und auf Anmeldeseite navigiert.

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

Der Testfall ist kausal von den Testfällen *Indienststellung der Webanwendung* ⟨T000⟩ und *Anmelden an der* ArgosGUI ⟨T010⟩ abängig.

4 Integrationstest

Dieses Kapitel dokumentiert die Vorgehensweise im Rahmen des *Integrationstests*.

Ziel der in dieser Teststufe durchgeführten Tests ist die Überprüfung des spezifikationsgemäßen Zusammenspiels der einzelnen Softwarekomponenten, die im Rahmen des Projekts Argos entwickelt wurden.

In den nachfolgenden Unterabschnitten werden zunächst die zu testenden Komponenten identifiziert und das verwendete Testverfahren beschrieben, anschließend die konkreten Testfälle spezifiziert.

4.1 Zu testende Komponenten

Nr	**Komponenten**	**Testfälle**
1	⟨C010⟩ ArgosGUI	⟨T100⟩
2	⟨C020⟩ Identitätsmanagementdienst	⟨T100⟩,⟨T110⟩, ⟨T140⟩, ⟨T150⟩
3	⟨C030⟩ Visualisierungsdienst	⟨T100⟩, ⟨T120⟩, ⟨T140⟩
4	⟨C040⟩ Benachrichtigungsdienst	⟨T100⟩, ⟨T130⟩, ⟨T150⟩

Tabelle 4.1: Im Rahmen des Integrationstests zu testende Komponenten

4.2 Testverfahren

Der Ablauf des Testverfahrens folgt nachstehendem Schema:

- Zunächst werden die Komponenten *Identitätsmanagementdienst* ⟨C020⟩, *Visualisierungsdienst* ⟨C030⟩ und *Benachrichtigungsdienst* ⟨C040⟩ jeweils voneinander isoliert im Verbund der Komponente *HAPI FHIR JPA Server* ⟨C050⟩ getestet.
- Danach wird das orchestrierte[1] Zusammenspiel der Komponenten ⟨C020⟩ – ⟨C040⟩ gemäß Verhaltenspezifikation getestet.
- Abchließend erfolgt ein Funktionstest der Komponente ArgosGUI ⟨C010⟩ in Verbund mit den Komponenten ⟨C020⟩ – ⟨C050⟩ in einem produktivbetriebsnahen Szenario getestet.

[1] https://de.wikipedia.org/wiki/Dienstekomposition#Orchestrierung

Aus diesem Schema ergeben sich die in Abschnitt 4.3 dargestellten Testfälle.

4.2.1 Testskripte

Die in Abschnitt 4.3 dargestellten Testfälle werden unter Einsatz des Unit-Testing-Frameworks JUnit automatisiert durchgeführt.

4.3 Testfälle

4.3.1 Testfall ⟨T100⟩ - Black-Box-Test der Komponente `ArgosGUI` ⟨C010⟩

Ziel

Sicherstellung von Zugriffskontrolle und Navigierbarkeit der Webanwendung

Objekte/Methoden/Funktionen

- `ArgosUI`
  - `login(Credentials) : Session`
  - `logout(Session) : void`
  - `getUserProfile(Session) : User`
  - `setUserProfile(User) : void`
  - `getVisualization(Properties, Session) : Component`
- `ArgosUINavigator`
  - `navigateTo(String) : Component`

Pass/Fail Kriterien

Pass: Erfolgreiche Nutzerauthentifizierung, Erbringung angeforderter Produktfunktionen über spezifizierte Navigationspfade

Fail: Nutzerauthentifizierung scheitert trotz Bereitstellung valider Anmeldedaten

Fail: Patientenauswahldialog ist fehlerbehaftet

Fail: Visualisierungsartenauswahldialog ist fehlerbehaftet

Fail: Erbringung angeforderter Produktfunktion scheitert

Vorbedingung

Indienststellung der Webanwendung (vgl. ⟨T100⟩).
Zwischen Client und Webanwendung besteht eine TCP/IP-Verbindung,
die eine Kommunikation über Port 8080 zulässt.

Einzelschritte

1. Aufruf der Testinstallation der Webanwendung Argos im Web-Browser (im Testbetrieb über http://<host-ip>:8080/)
2. Anmeldung über Anmeldedialog der Startseite unter Verwendung der Zugangsdaten eines speziellen Testnutzerkontos (mit Benutzererkennung *user* und Passwort *passwort*)
3. Angezeigte Patientenauswahl (Soll: *patient1*) überprüfen
4. Auswahl von *patient1*
5. Überprüfen, ob Option *Weiter* freigeschaltet ist
6. Auswahl der Option *Weiter*
7. Überprüfen, ob korrekte Visualisierungsartenauswahl (Soll: Balkendiagramm) angeboten wird
8. Auswahl der Option *Balkendiagramm*
9. Überprüfen der Korrektheit der Anzeige
10. Auswahl der Option *Zurück*
11. Auswahl von *Ausloggen*
12. Überprüfen, ob Nutzer ausgeloggt ist

Beobachtungen/Log/Umgebung

Eine dedizierte Protokollierung der Ergebnisse dieses Testfalls ist nicht vorgesehen; Erfolg bzw. Scheitern ist unmittelbar durch Beobachtung der Ausgabe der Benutzerschnittstelle zu interpretieren.

Abhängigkeiten

⟨T100⟩, ⟨T010⟩, ⟨T020⟩, ⟨T030⟩, ⟨T040⟩, ⟨T050⟩, ⟨T060⟩, ⟨T070⟩, ⟨T080⟩ ⟨T110⟩, ⟨T120⟩, ⟨T130⟩, ⟨T140⟩, ⟨T150⟩

4.3.2 Testfall ⟨T110⟩ - Identitätsmanagementdienst ⟨C020⟩ + FHIR-Server ⟨C050⟩

Ziel

Es wird getestet, ob das Zusammenspiel zwischen dem Identitätsmanagementdienst und FHIR-Server funktioniert. Nach Aufruf muss der Identitätsmanagementdienst die richtigen Daten aus dem FHIR-Server anfordern und diese dann zurückliefern.

Objekte/Methoden/Funktionen

getUser(), expire(), grant()

Pass/Fail Kriterien

Pass: Erwartete Daten sind korrekt

Fail: Es werden keine oder fehlerhafte Daten zurückgegeben

Vorbedingung

Der FHIR-Server ist gestartet. Der FHIR-Server und der Identitätsmanagementdienst sind über ein TCP/IP-Protokoll verbunden und können über einen Port 8080 kommunizieren.

Einzelschritte

1. Identitätsmanagement Daten beim FHIR-Server anfordern lassen
2. Zurückgegebene Daten aus dem FHIR-Server anschauen
3. Überprüfen, ob richtige Daten zurückgegeben wurden

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall.

Abhängigkeiten

⟨T010⟩, ⟨T020⟩, ⟨T030⟩

4.3.3 Testfall ⟨T120⟩ - Visualisierungsdienst ⟨C030⟩ + FHIR-Server ⟨C050⟩

Ziel

Es wird getestet, ob das Zusammenspiel zwischen dem Visualisierungsdienst und dem FHIR-Server funktioniert. Nach dem Aufruf muss der Visualisierungsdienst die richtigen Daten aus dem FHIR-Server anfordern und diese dann zurückliefern.

Objekte/Methoden/Funktionen

getBarChart(), getCirclePlot(), getHeatMap(), getLiveStream()

Pass/Fail Kriterien

Pass: Erwartete Daten sind korrekt

Fail: Es werden falsche Daten oder gar keine Daten zurückgegeben

Vorbedingung

Der FHIR-Server ist gestartet. Der FHIR-Server und der Visualisierungsdienst sind über ein TCP/IP-Protokoll verbunden und können über einen Port 8080 kommunizieren.

Einzelschritte

1. Visualisierungsdienst Daten beim FHIR-Server anfordern lassen
2. Zurückgegebene Daten aus dem FHIR-Server anschauen.
3. Überprüfen, ob richtige Daten zurückgegeben wurden.

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall.

Abhängigkeiten

⟨T040⟩, ⟨T050⟩, ⟨T070⟩

4.3.4 Testfall ⟨T130⟩ - Benachrichtigungsdienst ⟨C040⟩ + FHIR-Server ⟨C050⟩

Ziel

Es wird getestet, ob das Zusammenspiel zwischen dem Benachrichtigungsdienst und dem FHIR-Server funktioniert. Nach dem Aufruf muss der Benachrichtigungsdienst die richtigen Daten aus dem FHIR-Server anfordern und diese, wenn vorhanden, zurückliefern.

Pass/Fail Kriterien

Pass: Erwartete Daten sind korrekt

Fail: Es werden falsche Daten oder, obwohl richtige Daten vorhanden sind, keine Daten zurückgeliefert

Vorbedingung

Der FHIR-Server ist gestartet. Der FHIR-Server und der Benachrichtigungsdienst sind über ein TCP/IP-ÜProtokoll verbunden und können über einen Port 8080 kommunizieren.

Einzelschritte

1. Benachrichtigungsdienst Update beim FHIR-Server anfordern lassen.
2. Wenn Daten vorhanden, werden sie vom FHIR-Server zurückgeliefert
3. Angezeigte Daten überprüfen. Falls keine Daten angezeigt werden überprüfen, ob es Daten geben sollte

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall.

Abhängigkeiten

⟨T060⟩

4.3.5 Testfall ⟨T140⟩ - Identitätsmanagementdienst ⟨C020⟩ + Visualisierungsdienst ⟨C030⟩ + FHIR-Server ⟨C050⟩

Ziel

Es wird getestet, ob das Zusammenspiel zwischen dem Identitätsmanagementdienst, dem Visualisierungsdienst und dem FHIR-Server funktioniert.

Bevor der Visualisierungsdienst eine Visualisierung anzeigen kann fragt er beim Identitätsmanagement an, ob das Berechtigungstoken für den ausgewählten Patienten für die jeweilige Visualisierung gültig ist. Ist das Token gültig, werden die Daten beim FHIR-Server angefordert und von diesem zurückgeliefert.

Pass/Fail Kriterien

Pass: Die Komponenten arbeiten richtig zusammen

Fail: Es werden keine oder die falschen Daten zurückgeliefert

Fail: Es werden Daten angezeigt, obwohl kein Berechtigungstoken vorhanden

Vorbedingung

Testfall ⟨T110⟩, Testfall ⟨T120⟩ waren erfolgreich

Einzelschritte

1. Visualisierungsdienst Authorisierungsanfrage an den Identitätsmanagementdienst(mit Berechtigungstoken) stellen lassen
2. Identitätsmanagement bestätigt Anfrage
3. Visualisierungsdienst Daten aus FHIR-Server anfordern lassen und diese zurückgeliefert bekommen.
4. Überprüfen der Daten.

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall.

Abhängigkeiten

⟨T030⟩

4.3.6 Testfall ⟨T150⟩ - Identitätsmanagementdienst ⟨C020⟩ + Benachrichtigungsdienst ⟨C040⟩ + FHIR-Server ⟨C050⟩

Ziel

Es wird getestet, ob das Zusammenspiel zwischen dem Identitätsmanagementdienst, dem Benachrichtigungsdienst und dem FHIR-Server funktioniert.
Bevor der Benachrichtigungsdienst im FHIR-Server nach Daten fragen kann, fragt er beim Identitätsmanagementdienst an, ob für den ausgewählten Patienten das mitgegebene Berechtigungstoken für das Live-Bild gültig ist. Ist das Token gültig, werden die Daten vom Benachrichtigungsdienst beim FHIR-Server angefordert und von diesem zurückgeliefert.

Objekte/Methoden/Funktionen

Pass/Fail Kriterien

Pass: Token wurde überprüft und ist gültig, Daten werden vom Benachrichtigungsdienst angefordert und vom FHIR-Server zurückgeliefert.
Fail: Token ist ungültig.
Fail: Daten werden nicht zurückgeliefert.

Vorbedingung

Testfall ⟨T110⟩, Testfall ⟨T130⟩ waren erfolgreich

Einzelschritte

1. Benachrichtigungsdienst beim Identitätsmanagementdienst die Information anfordern lassen, ob Berechtigungstoken für das Live-Bild gültig ist.

2. Nach Bestätigung des Tokens Benachrichtigungsdienst Daten beim FHIR-Server anfordern lassen und diese, wenn vorhanden, zurückbekommen.

3. Zurückgegebene Daten überprüfen.

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall.

Abhängigkeiten

⟨T060⟩

5 Unit-Tests

Dieses Kapitel beschreibt die Vorgehensweise im Rahmen der Teststufe *Modultest*.

Die hier betrachtete Testfälle haben die Aufgabe, einzelne Komponenten auf Fehleranfälligkeit zu testen. Im Fokus der Betrachtung stehen dabei die Schnittstellen der Komponenten, die Eingaben von außen, d. h., durch Nutzereingaben oder Aufrufe durch andere Komponenten, erhalten.

5.1 Zu testende Komponenten

Folgende Komponenten werden in dieser Teststufe abgedeckt:

Nr	Komponenten	Testfälle
1	⟨C020⟩ Identitätsmanagementdienst	⟨T200⟩, ⟨T210⟩
2	⟨C030⟩ Visualisierungsdienst	⟨T220⟩
3	⟨C040⟩ Benachrichtigungsdienst	⟨T230⟩

Tabelle 5.1: Im Rahmen des Modultests zu testende Komponenten

Die Komponente ArgosGUI ⟨C010⟩ wird hingegen nicht isoliert, sondern nur durch Ausführung in einer produktivbetriebsnahen Testumgebung und Durchlauf des Integrationstestfalls ⟨T100⟩ getestet.

5.2 Testverfahren

Tests der Komponenten *Identitätsmanagementdienst* ⟨C020⟩, *Visualisierungsdienst* ⟨C030⟩ und *Benachrichtigungsdienst* ⟨C040⟩ werden sämtlich als Black-Box-Tests realisiert.

Hierbei werden, soweit erforderlich, Objekte der zu testenden Komponenten mit Testdaten befüllt, die es erlauben, die Korrektheit von Ausgaben zu überprüfen.

5.2.1 Testskripte

Die in Abschnitt 5.3 dargestellten Testfälle werden unter Einsatz des Unit-Testing-Frameworks JUnit automatisiert durchgeführt. Manuelle Testskripte finden in dieser Teststufe keine Verwendung.

5.3 Testfälle

5.3.1 Testfall ⟨T200⟩ - `IdentityServiceImpl` ⟨C210⟩

Ziel

Überprüfung des Sollverhaltens bei autorisiertem Zugriff

Objekte/Methoden/Funktionen

- `authenticate(Credentials) : Authorization`
- `expire(Authorization) : void`
- `getUser(Authorization) : User`
- `setUser(User, Authorization) : void`
- `grant(Collection<Request>, Authorization) : Authorization`

Pass/Fail Kriterien

Pass: Positiver Testbericht von JUnit (d. h., spezifikationsgemäße Absolvierung des Testfalls)

Fail: Negativer Testbericht von JUnit

Vorbedingung

Initiierung von `IdentityServiceImpl` mit geeigneten Testdaten

Einzelschritte

1. Anforderung einer Autorisierung über `authenticate(Credentials)` mit validen Anmeldedaten
2. Abfrage der Profildaten des authentifizierten Nutzers durch `getUser(Authorization)`
3. Modifikation der Profildaten, Aktualisierung des Nutzerprofils über `setUser(User, Authorization)` (unter Verwendung der in Schritt 1 gewonnenen Autorisierung)
4. Wiederholter Abruf der (aktualisierten) Profildaten, Vergleich von Ist gegen Soll
5. Entwerten der erlangten Autorisierung mittels `expire(Authorization)`
6. Wiederholter Versuch, Profildaten mit der in Schritt 1 gewonnenen Autorisierung abzurufen

Bemerkung: `getUser, setUser` verwenden intern `grant(Collection<Request>, Authorization)`, um Profildaten abzurufen respektive zu aktualisieren.

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall

5.3.2 Testfall ⟨T210⟩ - `IdentityServiceImpl` ⟨C210⟩

Ziel

Überprüfung des Sollverhaltens bei unautorisiertem Zugriff

Objekte/Methoden/Funktionen

- `grant(Collection<Request>, Authorization) : Authorization`

Pass/Fail Kriterien

Pass: Negativer Testbericht von JUnit

Fail: Postiver Testbericht von JUnit

Einzelschritte

Anforderung des Zugriffs auf eine Ressource ohne erforderliche Autorisierung.

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall

5.3.3 Testfall ⟨T220⟩ - VisualisationServiceImpl ⟨C310⟩

Ziel

Überprüfung des Sollverhaltens bei autorisierter Anfrage

Objekte/Methoden/Funktionen

- `getBarChart(Properties, Authorization) : VisualizationContext`
- `getCirclePlot(Properties, Authorization) : VisualizationContext`
- `getHeapMap(Properties, Authorization) : VisualizationContext`
- `getLiveStream(Properties, Authorization) : VisualizationContext`

Pass/Fail Kriterien

Pass: Positiver Testbericht von JUnit (d. h., Ist-Visualisierungskontext ist konform mit Soll-Kontext)

Fail: Negativer Testbericht von JUnit

Vorbedingung

Initiierung von `VisualizationServiceImpl` mit geeigneten Testdaten

Einzelschritte

1. Aufruf von `getBarChart(Properties, Authorization)`
2. Aufruf von `getCirclePlot(Properties, Authorization)`
3. Aufruf von `getHeatMap(Properties, Authorization)`
4. Aufruf von `getLiveStream(Properties, Authorization)`

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall

5.3.4 Testfall ⟨T230⟩ - NotificationServiceImpl ⟨C410⟩

Ziel

Überprüfung des Sollverhaltens bei autorisiertem Aufruf

Objekte/Methoden/Funktionen

- `register(register(Properties, Callable, Authorization) : void`

Pass/Fail Kriterien

Pass: Positiver Testbericht von JUnit

Fail: Negativer Testbericht von JUnit

Vorbedingung

Initiierung von `NotificationServiceImpl` mit geeigneten Testdaten

Einzelschritte

1. Aufruf von `register(register(Properties, Callable, Authorization)`

Beobachtungen/Log/Umgebung

Test-Bericht von JUnit für diesen Testfall

6 Glossar

(AAL): Ambient Assisted Living

(AGT):Assistierende Gesundheitstechnologien

Client:
Endgerät, das Dienste von einem Server abruft

Commit:
Ausdruck, der die bestätigende Freischaltung einer Änderung beschreibt

Dienstrechner (Host):
Endgerät, das in einem verteilten System seine Dienste für andere Endgeräte (siehe *Client*) anbietet

(FHIR): Fast Healthcare Interoperable Resources

Framework:
Programmiergerüst, das in der Softwaretechnik verwendet wird

Geriatrie:
Lehre von den Krankheiten des alternden Menschen

Graphical User Interface (GUI):
Grafische Benutzerschnittstelle

Heatmap:
Diagramm zur Visualisierung von Daten, deren abhängige Werte einer zweidimensionalen Definitionsmenge als Farben repräsentiert werden

Javadoc:
Erstellt aus Java-Quelltexten automatisch HTML-Dokumentationsdateien

JSON:
Kompaktes Datenformat für den Datenaustausch zwischen Anwendungen

JUnit:
Framework zum Testen von Java-Programmen

Redmine:
Webbasierte Projektmanagementsoftware

Repository:
Verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten

Representational State Transfer (REST):
Programmierparadigma für verteilte Systeme, das auf einer Abstraktion von Struktur und Verhalten des World Wide Web basiert, d. h., ein Softwareprodukt als System lose gekoppelter Komponenten, die über eine zustandslose zustandslose Architektur, die in der Regel über HTTP läuft.

Responsives Webdesign:
Gestalterisches und technisches Paradigma zur Erstellung von Websites, so dass diese auf Eigenschaften des jeweils benutzten Endgeräts, vor allem Smartphones und Tabletcomputer, reagieren können

Subversion (SVN):
Freie Software zur zentralen Versionsverwaltung von Dateien und Verzeichnissen

Vaadin:
Freies Webanwendungs-Framework zur Entwicklung von Rich Internet Applications (RIA)