BACHELORARBEIT

SECURITY EDUCATION TRAINING AWARENESS APPLIKATION
FÜR KLEINUNTERNEHMEN

Verfasser

Lukas Krickl

angestrebter akademischer Grad

Bachelor of Science (BSc)

Wien, 2021

Studienkennzahl lt. Studienblatt:	A 033 521
Fachrichtung:	Informatik - Informatik Allgemein
Betreuerin / Betreuer:	Univ.-Prof. Dipl.-Ing. Dr. Dr. Gerald Quirchmayr

Inhaltsverzeichnis

1 Einleitung 1
1.1 Zielgruppe Kleinunternehmen 1
1.2 Sicherheitsrisiko: „Human Factor“ 1
1.3 SETA – Security Education Training Awareness 2
1.4 Zielsetzung 3
1.5 Vorgehensweise 4
2 Literatur auf der diese Arbeit aufbaut 4
2.1 Training 4
2.2 E-Learning Systems und traditionelle Learning Management Systems 5
2.2.1 Nachteile von traditionellen SCORM, LOM und DCM 6
2.3 E-Learning Systems basierend auf Service Oriented Architecture (SOA) 6
2.4 E-Learning Systems basierend auf Cloud Computing 6
2.4.1 E-Learning Systeme in einer Cloud Architektur 7
3 Design 7
3.1 Didaktisches Modell 7
3.1.1 Assessments 8
3.1.2 Lernen der Inhalte 8
3.1.3 Empfehlung 8
3.1.4 Organisation der Aufgaben 8
3.1.5 Beurteilung des Modells 8
3.2 Benutzergruppen: 9
3.3 Übersicht der Benutzeranforderungen 9
3.4 Übersicht der Systemanforderungen 9
3.5 Architektur Design 9
3.5.1 Szenario View 10
3.5.2 Physical View 11
3.5.3 Development View 12
3.5.4 Logical View 13
3.5.5 Process View 14
3.6 Datenmodell 15
3.7 Design der REST API 17
4 Implementierung 17

4.1 Technology Stack 17
4.2 Systemkomponenten 18
4.2.1 Datenbank 19
4.2.2 Server 19
4.2.3 Client 20
4.3 User Interface 20
4.4 Deployment 27
5 Test und Evaluierung 28
5.1 Funktionale Tests 28
5.1.1 Komponententests 28
5.1.2 Integrationstest 29
5.2 Case Study 30
5.2.1 Beschreibung des Unternehmens 31
5.2.2 IT-Infrastruktur 31
5.2.3 Einrichten der Applikation 31
5.2.4 Ergebnisse der Case Study 32
6 Diskussion der Ergebnisse 32
6.1 Evaluierung der Implementierung 32
6.2 Diskussion der Ergebnisse der Case Study 32
7 Conclusio und Ausblick 33
8 Literaturverzeichnis 34
9 Anhang 35
9.1 Benutzeranforderungen (User-Stories): 35
9.1.1 Information bezüglich sicheren Umgangs mit Computern und Daten 35
9.1.2 Informationen über Passwörter 35
9.1.3 Informationen über Sicherheitsaspekte des Internets 36
9.1.4 Informationen über E-Mails und Spam 36
9.1.5 Informationen über Schadprogramme 36
9.1.6 WKO Inhalte verwenden 36
9.1.7 Monitoring 36
9.2 Systemanforderungen 37
9.2.1 Bereitstellen von IT-sicherheitsrelevanter Information 37
9.2.2 Multiple-Choice Szenarien 37

9.2.3 Zuordnungsszenarien ... 37
9.2.4 Trainings ... 38
9.2.5 Prüfung ... 38
9.2.6 Trainings- und Prüfungshistorie ... 39
9.2.7 Trainer Rolle ... 39
9.3 Szenario View ... 40
9.4 Logical Views ... 54
9.4.1 Logical View User Management ... 54
9.4.2 Logical View Training ... 55
9.4.3 Logical View Prüfung ... 56
9.4.4 Logical View Prüfungs- und Trainingsergebnisse ... 57
9.5 Process View ... 58
9.5.1 Process View Authentifizierung ... 58
9.5.2 Process View Autorisierung ... 59
9.5.3 Process View Training ... 60
9.6 Datenmodell ... 60
9.7 REST API ... 60
9.7.1 Meta Information ... 60
9.7.2 Szenarien ... 62
9.7.3 Authentification ... 63
9.7.4 Assessments ... 64
9.7.5 Assessments Abrufen (Ergebnisse) ... 66
9.7.6 Trainer Funktionen ... 69
9.8 Implementierung und Code-Basis ... 69
9.8.1 docker-compose.yml ... 69
9.8.2 Dockerfiles ... 70
9.9 Datenbank ... 71
9.9.1 Tabellen Statements (Database\setup\001_setup_tables.sql) ... 71
9.9.2 Nutzer Statements (Database\setup\001_setup_tables.sql) ... 72
9.9.3 View Statements (Database\setup\001_setup_tables.sql) ... 72
9.9.4 Kapitel Statements (Database\setup\001_setup_tables.sql) ... 73
9.9.5 Themen Statements (Database\setup\001_setup_tables.sql) ... 73
9.10 Server ... 74

9.10.1 application.properties 74
9.10.2 AuthenticationConfiguration.java 74
9.10.3 Controller 76
9.10.4 Services 84
9.10.5 Repositories 86
9.10.6 Entities 95
9.11 Client 103
9.11.1 „assessment“ Component 103
9.11.2 „pruefung-result“ Component 108
9.11.3 „training-result“ Component 111
9.11.4 „pruefung-page“ Component 113
9.11.5 „training-page“ Component 120
9.11.6 „fortschritt-page“ Component 131
9.11.7 „monitoring“ Component 135

Abbildungsverzeichnis

Abbildung 1: SETA Aufbau nach NIST [5] 2
Abbildung 2: Projekt Entwicklung nach V-Modell abgeleitet von [7] 4
Abbildung 3: Trainingsphasen Übersicht 7
Abbildung 4: 4+1 Views Model von Kruchten [8] 10
Abbildung 5: Use Case Training 11
Abbildung 6: Physical View 12
Abbildung 7: Development View 12
Abbildung 8: Logical View Server Überblick 13
Abbildung 9: Logical View Client Überblick 14
Abbildung 10: Process View Assessment 15
Abbildung 11: Datenmodell Übersicht 16
Abbildung 12: Implementierung Ordnerstruktur 18
Abbildung 13:UI Inhalte 21
Abbildung 14: UI Inhalte Mobil 22
Abbildung 15:UI Navigation Mobil 23
Abbildung 16: UI MC-Frage mit Bild 24
Abbildung 17: UI Training Inhalte 24
Abbildung 18: UI Übungsaufgabe 25
Abbildung 19: UI Trainingsergebnis 25
Abbildung 20: UI Zuordnung in einer Prüfung 26
Abbildung 21: UI Prüfungsergebnis 26
Abbildung 22: UI Monitoring 27
Abbildung 23: Applikation Deployment 28
Abbildung 24: Komponententest AssessmentController Setup 29
Abbildung 25: Komponententest AssessmentController Testfall 29
Abbildung 26: End-zu-End Test Login 30

Tabellenverzeichnis

Tabelle 1: Nutzergruppen 9
Tabelle 2: Technologie 18

Abstract

IT-Security Training wird von Kleinunternehmen aufgrund dessen Kosten und Aufwand oft vernachlässigt. In dieser Arbeit wird das Design einer webbasierten Security Education Training Awareness Plattform und ein zugehöriger Prototyp entwickelt, die primär in Kleinunternehmen einsetzbar ist. Zuerst werden Charakteristiken von effektiven Trainings, unterschiedliche Architekturen von Lernplattformen untersucht und anschließend das entwickelte didaktische Modell, das Datenmodell, die Server und die Client Architektur vorgestellt. Die Implementierung des Prototyps wird funktional getestet und zum Schluss mittels einer Case Study auf Anwendbarkeit geprüft.

1 Einleitung

In Unternehmen sind IT-sicherheitsspezifische Maßnahmen nicht mehr wegzudenken. Bei Sicherheitsvorfällen bestehen Risiken für erhebliche Schäden an den Unternehmen selbst, sowie deren Stakeholder. Ordentliches Risiko- und Sicherheitsmanagement zu betreiben ist eine sehr kosten- und zeitintensive Angelegenheit, die nicht direkt zum Gesamtumsatz des Unternehmens beiträgt, weshalb oft bei notwendigen Sicherheitsvorkehrungen gespart wird. Wie durch Studien belegt wurde, ist die Umsetzung von IT-Sicherheit vor allem in kleineren Unternehmen eine Herausforderung.

Die WIK - Wissenschaftliches Institut für Infrastruktur und Kommunikationsdienste GmbH veröffentlichte 2017 eine Studie bezüglich der Lage der IT-Sicherheit in KMU, in der 1508 deutsche Unternehmen befragt wurden:

> „Allerdings beabsichtigt auch in diesem Jahr jedes dritte Unternehmen keinerlei Investitionen in diesem Bereich (33%). Der prozentuale Anteil der Ausgaben für IT-Sicherheit am IT-Budget insgesamt beträgt ca. 11% und liegt damit fast auf dem Niveau der vorausgegangenen Befragung.
>
> […]
>
> Kleine Unternehmen investieren häufiger gar nicht in IT-Sicherheit als größere KMU, was aber angesichts ihrer geringeren finanziellen Ressourcen nicht überrascht.“ [1]

In [1] sind auch die Gründe für die Verbesserung der IT-Sicherheit untersucht worden, wobei 75% der befragten Kleinunternehmen Zeit und 68% den Kostenaufwand nannten.

1.1 Zielgruppe Kleinunternehmen

Mit KMU werden Unternehmen bezeichnet, die in ihrer Anzahl der Mitarbeiterinnen und Mitarbeitern und in ihrem Umsatz limitiert sind. Die Wirtschaftskammer Österreich (WKO) deklariert Betriebe mit 10 bis zu 49 Mitarbeiterinnen und Mitarbeitern und einem Umsatz von bis zu 10 Mio. Euro als Kleinunternehmen. Betriebe mit bis zu 249 Mitarbeiterinnen und Mitarbeitern und einem Umsatz von bis zu 50 Mio. Euro werden als Mittlere Unternehmen kategorisiert [2].

Laut einem Bericht des Bundesministeriums für Digitalisierung und Wirtschaftsstandort können in Österreich im Jahr 2017 ca. 99.6% der Unternehmen der marktorientierten Wirtschaft als KMU klassifiziert werden, die insgesamt 67% der Angestellten dieses Sektors umfassen [3]. Somit zählt ein Großteil der Unternehmen Österreichs zu jenen, die womöglich keine ausreichenden Vorkehrungen der IT-Sicherheit getroffen haben und für die ein erhöhtes Risiko für Verluste durch Cybersecurity Vorfälle bestünde.

1.2 Sicherheitsrisiko: „Human Factor“

Ein Großteil der IT-Sicherheitsvorfälle in Unternehmen ist auf menschliches Fehlverhalten zurückzuführen. Das hat auch eine Umfrage des Kaspersky Lab und B2B International in 2017 ergeben [4]. So nannten 46% der befragten Unternehmen unvorsichtige und unwissende Mitarbeiterinnen und Mitarbeiter als Hauptfaktor für Sicherheitsvorfälle. Dadurch, dass sich

Angestellte eines Unternehmens als effektiver Ausgangspunkt für Cyberangriffe eignen und Maßnahmen wie Sensibilisierung und Schulungen nur spärlich umgesetzt werden, nehmen Angriffe dieser Art zu. In [4] wird ein Anstieg von Vorfällen in Kleinunternehmen, die auf menschliches Versagen zurückzuführen sind, von 25% auf 32% innerhalb eines Jahres verzeichnet.

1.3 SETA – Security Education Training Awareness

Ein „Security Education Training Awareness“ (SETA) Programm dient zur Aufklärung und Bildung von Angestellten eines Unternehmens bezüglich IT-Security, um Risiken von Sicherheitsvorfällen zu minimieren. Das SETA Programm ist in ein mehrstufiges Modell gegliedert, wobei jede Stufe auf eine andere Zielgruppe mit unterschiedlicher Verantwortung gerichtet ist. Das „National Institute of Standards and Technology“ (NIST) veröffentlichte mit der „NIST Special Publication 800-50“ [5] Richtlinien für die Erstellung und Wartung eines SETA Programms. Das NIST beschreibt den Aufbau eines SETA Programms folgendermaßen:

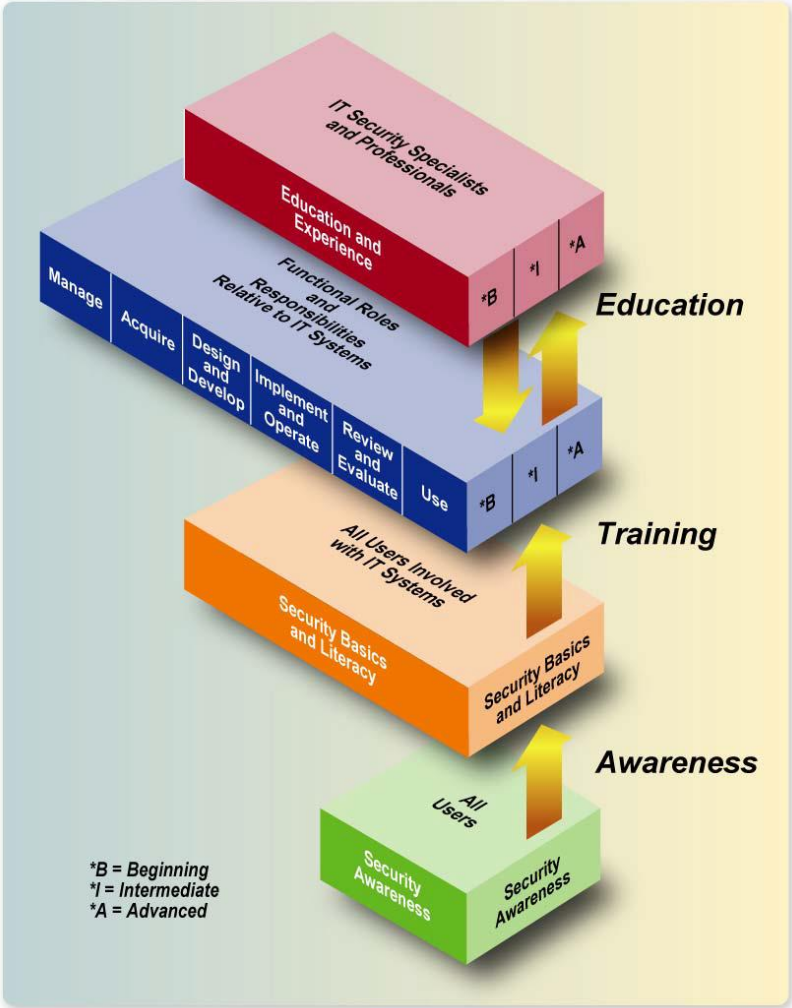


Abbildung 1: SETA Aufbau nach NIST [5]

Die Komplexität der IT-Security-Inhalte des Programms steigt mit jeder Stufe in einer „Bottom Up“ Herangehensweise, angefangen mit Sensibilisierung (Awareness) für alle Beteiligten bis zur Ausbildung (Education) von IT-Security-Spezialisten.

- Awareness: Im Unternehmen soll Sicherheits- und auch Verantwortungsbewusstsein verschafft werden. Das Ziel ist, Angestellten die mit IT-Systemen arbeiten, entsprechende Methoden und Vorgehensweisen der IT-Security beizubringen, damit in der Praxis durch deren Anwendung Sicherheitsvorfälle vermieden werden. Laut [5] erzielt diese Stufe eine Verhaltensänderung bei allen Beteiligten, sodass unter anderem IT-Security auch mehr Aufmerksamkeit geschenkt wird.

- Training: Beim Training geht es darum, Angestellte zu schulen, die eine funktionale Rolle in der IT-Sicherheit einnehmen. In der „NIST Special Publication 800-50“ [5] wird zwischen Awareness und Training so unterschieden, dass beispielsweise System Administratoren durch Training auch Aufgaben des Risikomanagements oder das Beheben von Sicherheitsvorfällen übernehmen können.

- Education: IT-Security Spezialisten, die eine sicherheitskritische Rolle innerhalb des Unternehmens einnehmen, benötigen eine fachgerechte Ausbildung. Herangehensweise kann laut [5] entweder das Einstellen von bereits ausgebildetem Personal sein, oder die explizite Schulung von einzelnen Angestellten.

Ein ordnungsgemäßes SETA Programm muss zuerst eingerichtet und anschließend immer auf neuestem Stand gehalten werden. Die damit verbundenen Aufgaben sind je nach Art des Unternehmens so umfangreich, dass sie nicht im bestehenden Alltag nebenbei erledigt werden können. Es müssen zusätzliche, nicht gewinnbringende Investitionen getätigt werden die, wie in vorherigen Kapiteln erwähnt, vor allem Kleinunternehmen vernachlässigen.

1.4 Zielsetzung

Diese Arbeit richtet sich an Betriebe, die von der WKO als Kleinunternehmen eingestuft werden und aus finanziellen Gründen nicht in der Lage sind, entsprechende Vorkehrungen der IT-Sicherheit zu treffen. Die Unternehmen haben zwischen 5 und 49 Angestellte, mindestens eine IT-Fachkraft und einen maximalen Umsatz von 10 Mio. Euro.

Es soll eine SETA Webapplikation entwickelt werden, die es Kleinunternehmen ermöglicht, ihre Angestellten kosteneffizient bezüglich IT-Sicherheit zu schulen, um Risiken von Sicherheitsvorfällen zu minimieren. Es werden Inhalte des Sicherheitshandbuches für Mitarbeiterinnen und Mitarbeiter [6] eingesetzt, die anhand von Szenario-basierten Übungen überprüft werden können.

Dabei sollen folgende Fragen beantwortet werden:

- Welche Systemarchitektur ist für eine SETA Webapplikation notwendig?
- Wie müssen Systemkomponenten einer SETA Webapplikation gestaltet werden:
    - Welche Anforderungen muss die Benutzeroberfläche erfüllen?
    - Welche Anforderungen muss die Serverarchitektur erfüllen?
    - Welche Anforderungen muss die Datenbank erfüllen?
- Welche Lernzyklen kommen bei SETA Applikationen zur Anwendung?

1.5 Vorgehensweise

Vorangehend wird eine Literaturanalyse bezüglich Trainings und Architekturen von E-Learning Systems durchgeführt. Die Webapplikation soll mithilfe des V-Modells von Boehm [7] in folgenden Schritten entwickelt werden:

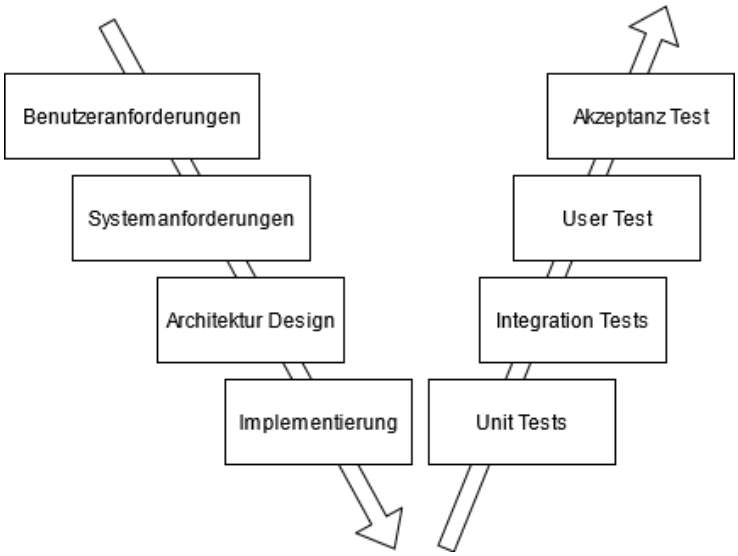


Abbildung 2: Projekt Entwicklung nach V-Modell abgeleitet von *[7]*

Zuerst werden Benutzeranforderungen in Form von „User Stories“ erhoben und anschließend funktionale Anforderungen des Systems formuliert. Die Architektur wird mittels des „4 + 1 Views Model“ [8], eines Datenmodells und der Definition der Schnittstellen modelliert und geplant. Nachdem die Implementierung abgeschlossen ist, können einzelne Komponenten mit zugehörigen „Unit Tests“ und das Zusammenspiel der Architektur mit „Integration Tests“ geprüft werden. Die Systemanforderungen werden durch eine Systeminspektion überprüft und die Umsetzung der Nutzeranforderungen durch die „User Stories“ validiert.

Die Anwendbarkeit der Webapplikation und die implementierte Lösung werden im Rahmen einer Case Study durch die beispielhafte Anwendung in einem Unternehmen evaluiert.

2 Literatur auf der diese Arbeit aufbaut

Zuerst werden die wichtigsten Aspekte eines effektiven Trainings und anschließend unterschiedliche Architekturen von E-Learning Systems untersucht.

2.1 Training

Training wird in [9] definiert als systematischer Aufwand, um Wissen, Fähigkeiten, Herangehensweisen oder andere Eigenschaften zu verbessern. Des Weiteren müssen die Absolventinnen und Absolventen auch in der Lage sein, gelerntes Wissen in der Praxis anzuwenden, wenn ein Training auch eine positive Veränderung innerhalb einer Organisation hervorrufen solle.

Die Wahl eines passenden didaktischen Modells ist daher ausschlaggebend für die Effektivität des Trainings. [9] beschreibt allerdings weitere Kernaspekte, die bei der Gestaltung von Trainings relevant sind:

- **Ziele des Trainings:** Die Ziele des Trainings müssen klar definiert und überprüfbar sein.
- **Individuelle Charakteristiken:** Eigenschaften der Trainingsabsolventinnen und Trainingsabsolventen, die zum Trainingserfolg beitragen:
    - Kognitive Fähigkeiten
    - Vertrauen in die eigenen Kompetenzen
    - Zielorientierung
    - Motivation
- **Organisationelle Charakteristiken:** Organisationskultur, Abläufe und Situationen innerhalb der Organisation wirken sich ebenfalls auf das Training aus.
- **Übungsmöglichkeiten:** Unterschiedliche Möglichkeiten, die nötigen Kompetenzen zu üben, erwies sich laut [9] als förderlich. Es wurden ebenfalls Stärken von webbasierten Trainings aufgezeigt, da Anwenderinnen und Anwender mehr Kontrolle über das Training und vor allem die Geschwindigkeit des Übens haben. Variationen der Übungsaufgaben, deren Reihenfolge oder Präsentation soll auch dazu beitragen, dass Absolventinnen und Absolventen allgemeine Kompetenz erlangen und diese weniger schnell verlernen.
- **Feedback:** Konstruktives und zeitgerechtes Feedback sei essentiell für den Erfolg des Trainings. Die Lernenden benötigen Rückmeldung über ihren aktuellen Wissenstand und ob dieser bereits ausreichend ist. Bei Feedback seien folgende Aspekte zu beachten:
    - Lernende sollen nicht kritisiert werden
    - Möglichkeiten der Verbesserung sollen aufgezeigt werden
    - Feedback soll umsetzbar sein
- **Lernstrategien:** Abhängig von den Lernzielen des Trainings gäbe es unterschiedliche Arten von Lernstrategien, wobei diese durch folgende grundsätzliche Eigenschaften gekennzeichnet seien:
    - Information und Konzepte, die erlernt werden sollen, müssen präsentiert werden
    - Fähigkeiten und Kompetenzen müssen vorgezeigt werden
    - Übungsmöglichkeiten sollten bereitgestellt werden
    - Lernende sollten während und nach dem Training Feedback erhalten

In [9] ist auch eine Liste mit möglichen Lernstrategien angegeben, in der auch Simulation-basiertes Training, Szenario-basiertes Training und E-Learning enthalten sind.

2.2 E-Learning Systems und traditionelle Learning Management Systems

E-Learning beschreibt laut [9] eine Lernstrategie, die elektronische Hilfsmittel für den Lernprozess einsetzt. Learning Management Systems (LMS) sind Systeme, die organisatorische Hilfestellung im E-Learning bieten soll. [10] beschreibt LMS als Systeme, bei denen Lernende registriert sind und Lernmaterialien abrufen können. Durch LMS sei es auch möglich, Abgaben für Dokumente einzurichten, Lösungen zu präsentieren und den Fortschritt der Lernenden zu überwachen.

Lernmaterialien und Elemente von Kursen der LMS seien definiert durch das „Shared Content Object Reference Model“ (SCORM), ein standardisiertes Format für Kurselemente, das von den meisten LMS unterstützt werde.

SCORM sei nach [10] in drei Module unterteilt: CAM für die Definition von Lerninhalten, SN für die Navigation zwischen den Inhalten und RTE beschreibt, wie Kurselemente gestartet werden, bzw. wie mit dem LMS kommuniziert wird. Eine Website verwendet eine API um SCORM Funktionen zu nutzen und mit dem LMS zu kommunizieren. LOM und DCM sind zwei Alternativen zu SCORM, wobei alle drei Spezifikationen auf dem XML-Format basieren [11].

2.2.1 Nachteile von traditionellen SCORM, LOM und DCM

Wie auch in [11] beschrieben, sind die großen Nachteile von standardisierten Formaten für Ressourcen von LMS der Mangel an Flexibilität und Integrationsmöglichkeiten von alternativen Lernmaterialien. So können Lernplattformen nicht ausreichend angepasst und speziellere Lernelemente, die nicht dem Format des verwendeten Standards entsprechen, auch nicht eingerichtet werden. Des Weiteren wird in [12] der Fokus dieser LMS auf administrative Aufgaben, wie beispielsweise die Erstellung von Kursen und Materialien anstatt auf die Lernenden kritisiert.

2.3 E-Learning Systems basierend auf Service Oriented Architecture (SOA)

In einer Serviceorientierten Architektur werden Geschäftsprozesse oder Funktionalitäten in eigenständige Softwarekomponenten als „Service Provider“ gekapselt, die als Teil eines verteilten Systems von „Service Consumer“ angesprochen und genutzt werden können [13] [14].

Eine gängige Gliederung von Services eines E-Learning Systems ist laut [12] die Einteilung in Applikationen, Applikationen mit Nutzerinteraktion, Services des Lernprozesses, Allgemeine Services wie beispielsweise Authentifizierung und Infrastruktur Services. [11] nutzt wiederum eine „Layered Architecture“ und klassifiziert Komponenten in folgende Schichten:

- Daten Schicht: stellt verschiedene Ressourcen bereit
- Service Schicht: stellt Systemfunktionalität bereit
- „Business Logic“ Schicht: organisiert und nutzt Systemfunktionalität der Service Schicht, um Geschäftsprozesse umzusetzen
- Benutzeroberfläche: Darstellung und Interaktion mit den Anwenderinnen und Anwendern

2.4 E-Learning Systems basierend auf Cloud Computing

Cloud Computing ermöglicht die Verwendung von Ressourcen wie Netzwerke, Server, Speicher, Services oder Applikationen eines Cloud Providers auf eine dynamische Art und Weise und mit minimalem Managementaufwand [15]. Cloud Ressourcen werden als drei Servicemodelle angeboten:

- Software as a Service: Anwendungen werden direkt von dem Cloud Provider zur Verfügung gestellt und können beispielsweise über den Webbrowser verwendet werden.
- Plattform as a Service: Laufzeitumgebungen von Applikationen und Systemen können direkt vom Cloud Provider genutzt werden.
- Infrastructure as a Service: Der Cloud Provider stellt Server Infrastruktur bereit, die genutzt werden kann.

2.4.1 E-Learning Systeme in einer Cloud Architektur

Der große Unterschied zwischen traditionellen und Cloud-basierten Architekturen ist, dass in einer Cloud alle infrastrukturspezifischen und die meisten administrativen Aufgaben von dem Cloud Provider übernommen werden. Ein Vorteil bei der Verwendung von E-Learning Systemen in einer Cloud, der auch in [16] aufgezeigt wird ist, dass sich Institutionen und Organisationen auf den Lehr- und Lernprozess konzentrieren können und sich nicht mit der Einrichtung, Administration und Wartung von Systemen herumschlagen müssen.

[16] stellt auch eine mögliche Architektur eines Cloud-basierten E-Learning Systems vor. Das System sei in fünf Schichten eingeteilt: Die Infrastruktur Schicht bietet dynamisch skalierbare Ressourcen, die Software Schicht agiert als Zugang für Erstellerinnen und Ersteller von Inhalten, die Resource Management Schicht ist für das Verwalten von Software und Hardware Ressourcen zuständig, während die Service Schicht alle drei Service Modellen des Cloud Computing und die Applikationsschicht E-Learning Anwendungen anbietet. Vorteile dieser Architektur sind laut [16] eine hohe Rechenleistung und Speichermöglichkeit, Verfügbarkeit, Security, Virtualisierung, leichte Zugänglichkeit zu Softwarekomponenten und verbesserte Interaktion zwischen Lehrenden und Lernenden.

Eine ähnliche Struktur wird auch in [17] präsentiert. Anders als in [16] wird in [17] die Software und die Ressource Management Schicht parallel zueinander auf dieselbe Ebene gesetzt und die Applikationsschicht mit einem Lernportal basierend auf „Artificial Intelligence“ versehen.

3 Design

3.1 Didaktisches Modell

Trainings der Plattform werden in folgende vier Phasen gegliedert:

1. Assessment
2. Lernen der Inhalte
3. Assessment
4. Empfehlung

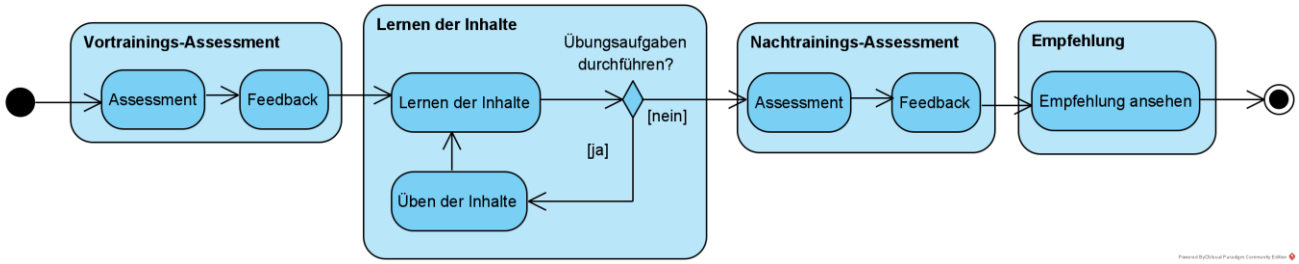


Abbildung 3: Trainingsphasen Übersicht

3.1.1 Assessments

In Assessments werden den Anwenderinnen und Anwendern eine Reihe von Aufgaben gestellt, die einem Szenario des Alltages nachempfunden sind. Die Aufgaben haben entweder die Form von Multiple-Choice Fragen oder Zuordnungsaufgaben. Multiple-Choice Fragen haben mehrere Antwortmöglichkeiten, von denen keine, eine, mehrere oder alle korrekt sein können. Zuordnungen sind Aufgaben, bei denen mehrere Begriffe und deren Beschreibungen gegenübergestellt sind. Begriffe müssen Beschreibungen zugeordnet werden, wobei ein Begriff mehreren, aber mindestens einer Beschreibung zugeordnet werden kann. Multiple-Choice Fragen können auch Bilder enthalten, von denen die Fragen handeln.

Nachdem das Assessment abgeschlossen wurde, wird Anwenderinnen und Anwendern Feedback bezüglich der Aufgaben gegeben, wobei auf korrekte und inkorrekte Antworten hingewiesen wird.

In dieser Arbeit werden Multiple-Choice Fragen oder Zuordnungsaufgaben als Aufgaben oder Szenarien bezeichnet.

3.1.2 Lernen der Inhalte

Informationen bezüglich IT-Sicherheit, die für die Durchführung der Aufgaben notwendig sind, werden in dieser Phase bereitgestellt. Des Weiteren können anhand von Übungsaufgaben die bereitgestellten Inhalte geübt und gelernt werden. Übungsaufgaben sind zufällig ausgewählte Aufgaben, die während des Lernens der Inhalte beliebig oft angefragt werden können.

3.1.3 Empfehlung

In der Schlussphase des Trainings wird den Absolventinnen und Absolventen ein Vergleich der Ergebnisse des Vortrainings-Assessments (Assessment bei Trainingsbeginn) und Nachtrainings-Assessments (Assessment am Ende des Trainings) angezeigt und eine etwaige Verbesserung oder Verschlechterung wird evaluiert. Abhängig von dem Trainingsergebnis wird auch eine Empfehlung bezüglich nächster Schritte bereitgestellt.

3.1.4 Organisation der Aufgaben

Aufgaben von Vortrainings- und Nachtrainings-Assessment werden vor Trainingsbeginn zufällig ausgewählt. Des Weiteren werden Antworten bei Aufgaben bei jedem Training und bei jeder Übungsaufgabe zufällig angeordnet. Die Auswahl der Aufgaben von Vortrainings- und Nachtrainings-Assessment überschneidet sich nicht, sodass beim Vergleich der Resultate der Assessments keine Verfälschung stattfindet, falls Aufgaben des Vortrainings-Assessments ebenfalls im Nachtrainings-Assessment vorkommen sollten und deren korrekte Antwort im ersten Feedback aufgelöst werden.

3.1.5 Beurteilung des Modells

Das Didaktische Modell ist so gestaltet, um in [9] genannte Kriterien für ein effektives Training zu erfüllen. Ziele von Trainings sind klar durch die Aufgaben und Inhalte definiert und organisatorische Abläufe werden durch die Orientierung der Aufgaben an Alltagsszenarien integriert.

Übungsmöglichkeiten, die durch die Art und Weise der Organisation der Aufgaben variieren, werden in der zugehörigen Trainingsphase geboten. Feedback wird während des Trainings, nach den Assessments und zum Schluss in der Empfehlungsphase bereitgestellt.

3.2 Benutzergruppen:

[5] definiert eine Vielzahl von Rollen und deren Verantwortlichkeiten, die ein SETA Programm benötigt. Für kleinere Unternehmen wäre jedoch eine vereinfachte Definition eher umsetzbar:

Nutzergruppen:	**Anwendungsfälle:**
Mitarbeiterinnen und Mitarbeiter (Anwenderinnen und Anwender)	• benutzen die SETA Applikation • wenden Sicherheitsmaßnahmen an
SETA Applikation Administratorinnen und Administratoren	• Einrichten der Applikation • Warten der Applikation
Abteilungsleiterinnen und Abteilungsleiter (Trainerinnen und Trainer)	• verantwortlich dafür, dass zugehöriges Personal die notwendige Schulung erhält. • Monitoring der Angestellten

Tabelle 1: Nutzergruppen

3.3 Übersicht der Benutzeranforderungen

Ausgehend von der Bedingung, dass die SETA Applikation in Kleinunternehmen eingesetzt werden kann, entstehen im Allgemeinen folgende Benutzeranforderungen:

- Informationen der IT-Sicherheit können von Anwenderinnen und Anwendern abgefragt werden.
- Informationen der IT-Sicherheit müssen nicht von Administratorinnen und Administratoren selbst erzeugt werden, um Personalkosten zu minimieren.
- Aufgaben von Trainings müssen nicht von Administratorinnen und Administratoren selbst erzeugt werden, um Personalkosten zu minimieren.

Die vollständigen Benutzeranforderungen sind als User Stories Anhang 9.1 dokumentiert.

3.4 Übersicht der Systemanforderungen

Systemanforderungen ergeben sich einerseits aus den Benutzeranforderungen, andererseits auch durch das didaktische Modell. Zusätzlich wurde die Funktionalität einer Prüfung hinzugefügt. Der Wissensstand und die Kompetenzen der Anwenderinnen und Anwender sollen mit Prüfungen auch ohne vorangehendes Training abgeprüft werden können. Systemanforderungen sind in Anhang 9.2 ausführlich dokumentiert.

3.5 Architektur Design

Bei der Modellierung der Architektur kommt das „4+1 Views Model“ [8] zur Anwendung. Systeme werden bei dieser Methode aus vier unterschiedlichen Perspektiven, Logical View, Process View, Development View und Physical View beschrieben. Die fünfte Sicht, die das Plus Eins des Modellnamen ausmacht, ist die Scenario View, welche alle Sichten miteinander vereint. Die unterschiedlichen Sichten sollen folgende Aspekte der Architektur abbilden:

- Logical View: Objekte, Artefakte und Abhängigkeiten
- Process View: Prozesse und möglicherweise parallele Abläufe
- Development View: Organisation von Komponenten aus Sicht von Entwicklern
- Physical View: Organisation von Komponenten in Bezug auf die Infrastruktur
- Szenario View: Anwendungsfälle des Systems

In folgender **Abbildung *4*** ist eine Übersicht der fünf Perspekiven und deren Relationen zueinander dargestellt:

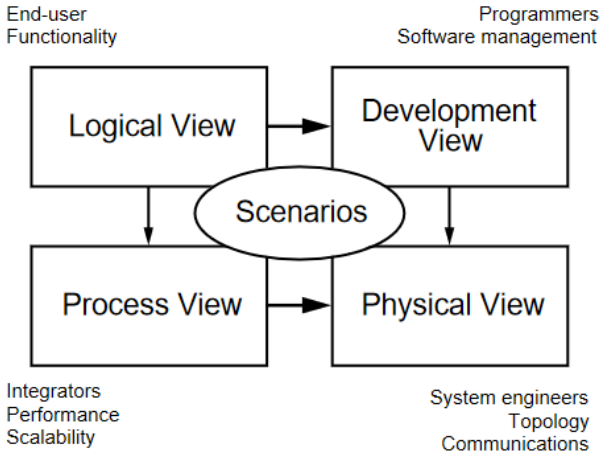


Abbildung 4: 4+1 Views Model von Kruchten *[8]*

3.5.1 Szenario View

Die Vollständigen Use-Cases und deren Beschreibung sind in Anhang 9.3 angegeben. Die wichtigsten Anwendungsfälle des Systems sind das Anmelden, Training, Monitoring und die Prüfung.

Die Authentifizierung und Autorisierung von Benutzerinnen und Benutzern des Systems ist der Einstiegspunkt für fast alle anderen Funktionen. Der einzige Anwendungsfall der keine Anmeldung benötigt, ist das Ansehen von Inhalten der IT-Sicherheit, bezeichnet mit USC004. Das soll Benutzerinnen und Benutzern ermöglichen, Informationen der IT-Sicherheit auf eine zügige Art und Weise nachzuschlagen.

Der Anwendungsfall USC005 des Trainings (siehe **Abbildung *5***) bildet das didaktische Modell ab und ist in Anwendungsfälle USC006 bis USC012 unterteilt. Prüfungen werden ebenfalls unterteilt und beinhalten das Durchführen von Szenarien und Ansehen des Feedbacks.

Das Monitoring kann nur von Trainerinnen und Trainern durchgeführt werden. Beim Monitoring können Ergebnisse von Prüfungen der Benutzerinnen und Benutzer abgerufen und die Kompetenzen der Mitarbeiterinnen und Mitarbeiter überprüft, bzw. mangelnde Kompetenz aufgedeckt werden.

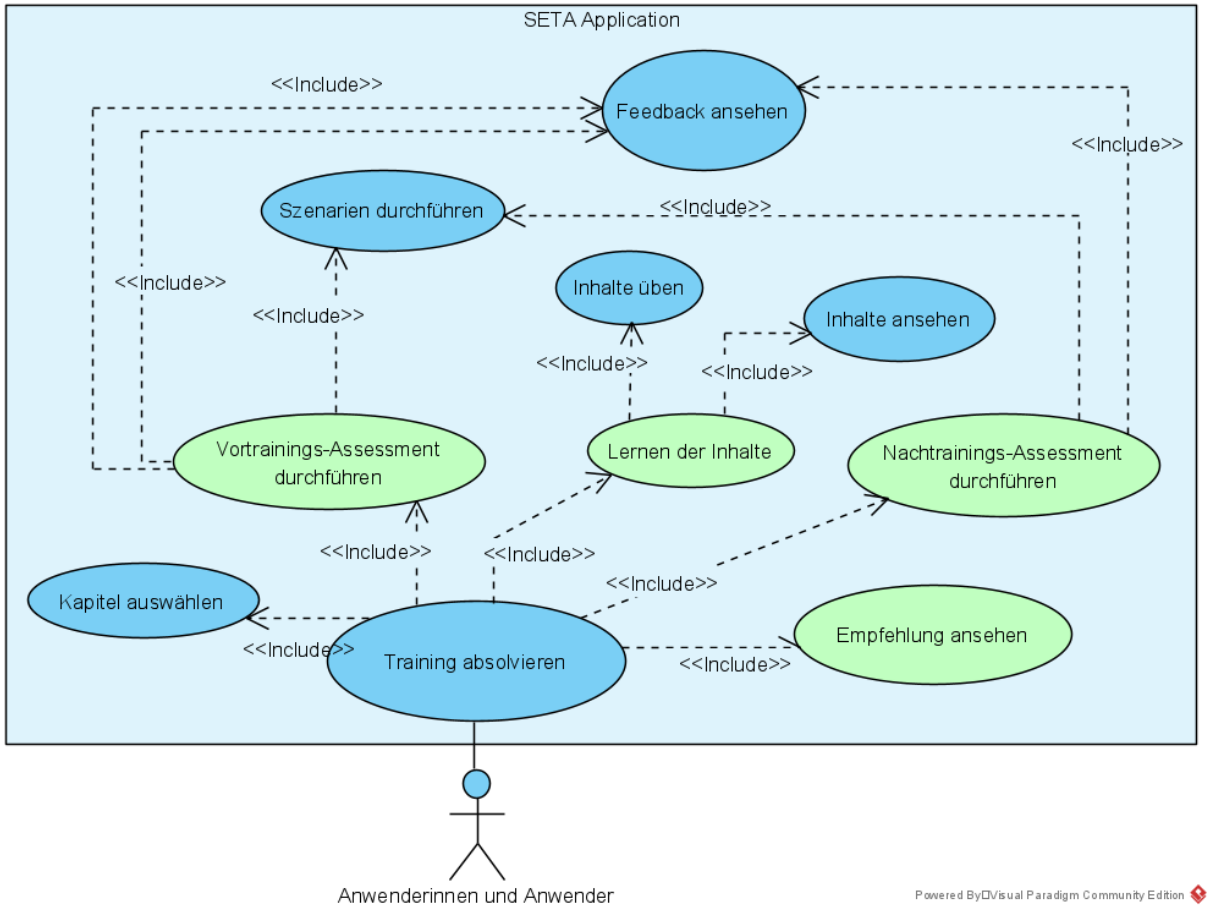


Abbildung 5: Use Case Training

3.5.2 Physical View

Die Infrastruktur der Webapplikation ist wie eine klassische Client-Server Architektur aufgebaut. Der Webbrowser kommuniziert als Client über das HTTPS Protokoll mit dem Server. Im Vergleich zu traditionellen monolithischen Architekturen ist der Server als „containerized application“ realisiert. Komponenten des Servers, wie zum Beispiel die Datenbank oder die Webserver Komponente, sind in Container, also unabhängige virtuelle Umgebungen, gekapselt. Container Umgebungen werden häufig in Cloud Computing Architekturen verwendet, von denen diese Designentscheidung inspiriert ist. Die Vorteile dieser Container Appliaktion sind Modularität, Plattformunabhängigkeit und Wartbarkeit. Ein weiterer wichtiger Vorteil ist die automatisierte Konfiguration der Komponenten und das automatisierte Deployment.

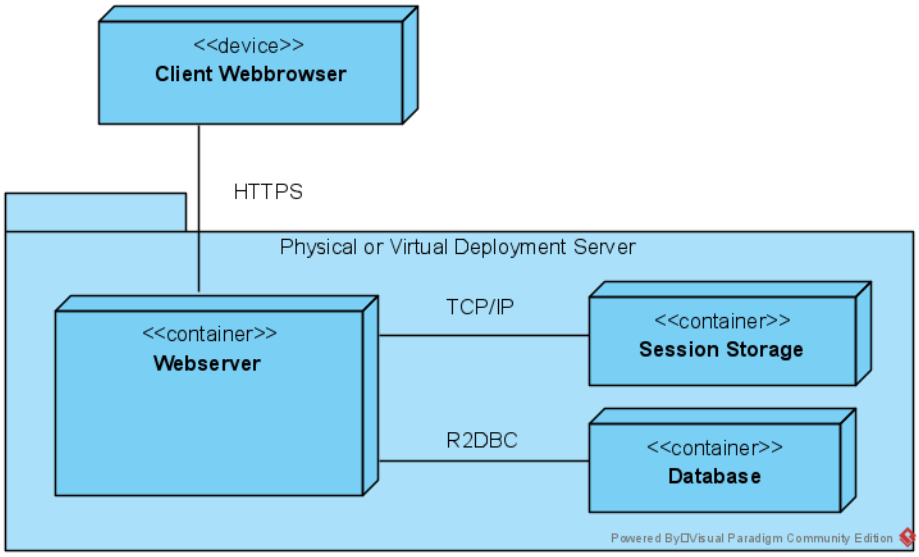


Abbildung 6: Physical View

3.5.3 Development View

Im Client wurde der Model View Controller Design Pattern angewendet. Die Webseite ist logisch in die Visuelle (HTML & CSS) und die kontrollierende Komponente (JavaScript) unterteilt. Die visuelle Komponente nimmt die Rolle der View ein, die den aktuellen Applikationszustand anzeigt. Die View wird durch den Controller, der kontrollierenden JavaScript Komponente, verändert und angepasst. Der Webserver symbolisiert das Model: applikationsspezifische Daten und Zustände, die über eine REST API vom Controller abgerufen und bearbeitet werden können. Die Webseite wird von dem Webserver bereitgestellt, wenn sie durch den Webbrowser angefragt wird. Weitere Interaktionen zwischen Client und Server finden über asynchrone HTTPS anfragen statt, visualisiert durch die AJAX Verbindung.

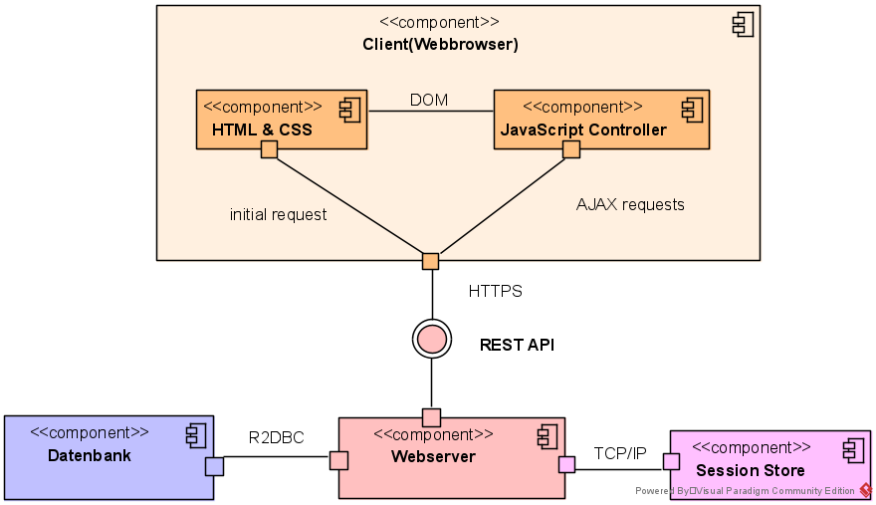


Abbildung 7: Development View

3.5.4 Logical View

In diesem Kapitel wird separat ein Überblick des logischen Aufbaus von Client und Server gegeben. Eine ausführlichere Übersicht befindet sich im Anhang 9.4.

3.5.4.1 Server

Der Server ist logisch in mehrere Pakete eingeteilt die als „Layered Architecture“ organisiert sind, ähnlich zu E-Learning Systems basierend auf SOA welche in 2.3 untersucht wurden:

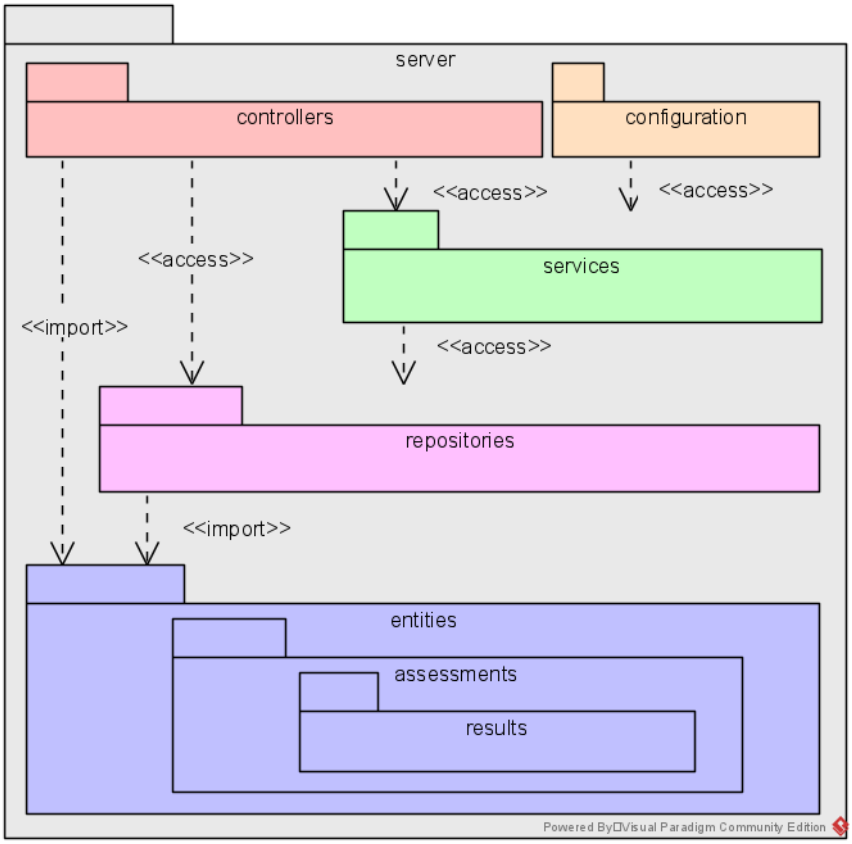


Abbildung 8: Logical View Server Überblick

Pakete sind logisch in Schichten eingeteilt, wobei Klassen nur auf die der darunterliegenden Pakete zugreifen. Diese Struktur bringt die Vorteile der Modifizierbarkeit von Schichten und Separation von Verantwortlichkeit mit sich.

Auf der obersten Ebene ist neben dem „configuration“ auch das „controllers“ Paket, das die „Endpoints“ der REST API enthält, wobei beide auf das darunterliegende „services“ Paket zugreifen. In „services“ sind wiederverwendbare Klassen mit der Funktionalität der Applikation, die von Klassen der darüberliegenden Schicht benötigt wird. Klassen des „repositories“ Paketes agieren als Instanzen des Datenzugriffes und sprechen die Datenbank an. Diese Instanzen werden einerseits von „services“ Klassen andererseits aber auch direkt von „controllers“ Klassen angesprochen, sollte keine weitere Datenverarbeitung durch ein Service notwendig sein. Das „entities“ Paket enthält Klassen, die Entitäten der Datenbank repräsentieren und innerhalb der gesamten Applikation verwendet werden.

3.5.4.2 Client

Der Client ist semantisch in die Pakete „components“, „services“ und „model“ eingeteilt, wobei „components“ als „controllers“ und „views“ spezialisiert sind. Das „model“ Paket enthält Klassen die Daten und Zustand der Applikation repräsentieren. Sie spiegeln Objekte des „entities“ Paketes des Servers wider, jedoch in abgeänderter Form, sodass sie von dem Client optimal genutzt werden können. Die Kommunikation mit der REST Serverschnittstelle geschieht über Klassen des „services“ Pakets und wird von denen des „controllers“ Paketes verwendet. Wie schon in 3.5.3 erwähnt, kommt der Model View Controller Design Pattern zum Einsatz, der auch in **Abbildung *9*** erkennbar ist. Klassen von „controllers“ greifen über Klassen von „services“ auf das Model zu und kontrollieren Instanzen des „views“ Paketes.

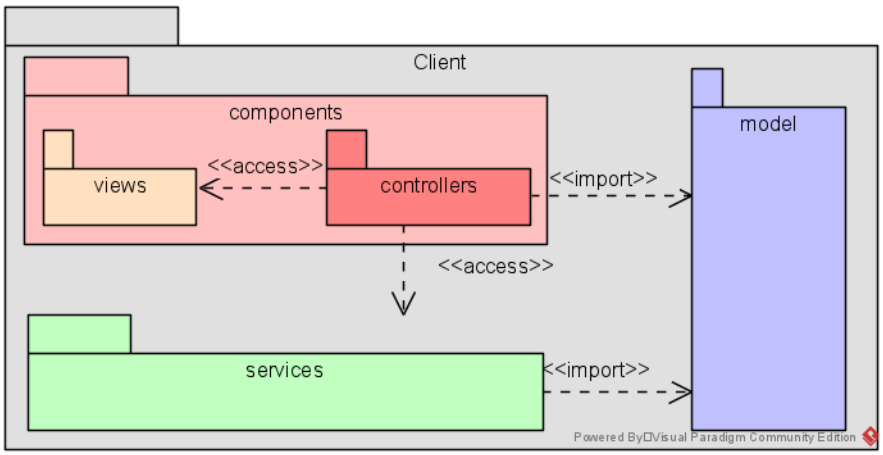


Abbildung 9: Logical View Client Überblick

3.5.5 Process View

Einer der häufigsten Abläufe der Applikation ist das Assessment. Assessments werden in Prüfungen und in Trainings als Vortrainings- und Nachtrainings-Assessment durchgeführt. Noch bevor das Assessment gestartet wird, werden dem Client die Identifier aller Szenarien, die in diesem Assessment abgeprüft werden sollen, überliefert. Die eigentlichen Daten eines Szenarios werden dynamisch, erst wenn es angezeigt wird, geladen. Dieser Prozess soll verhindern, dass beim Start des Assessments längere Ladezeiten entstehen, vor allem wenn Multiple-Choice Fragen mit Bildern angefragt werden müssen. Die Ladezeit der Szenarien wird durch diese Strategie über das gesamte Assessment verteilt, um so eine besser User Experience zu schaffen. Zusätzlich werden auch keine Netzwerkressourcen verschwendet, sollte ein Assessment abgebrochen werden. In **Abbildung *3*** ist der Prozess eines Assessments abstrahiert dargestellt. Visualisierungen der Prozesse der Authentifizierung, Autorisierung und des Trainings sind dem Anhang unter 9.5 beigelegt.

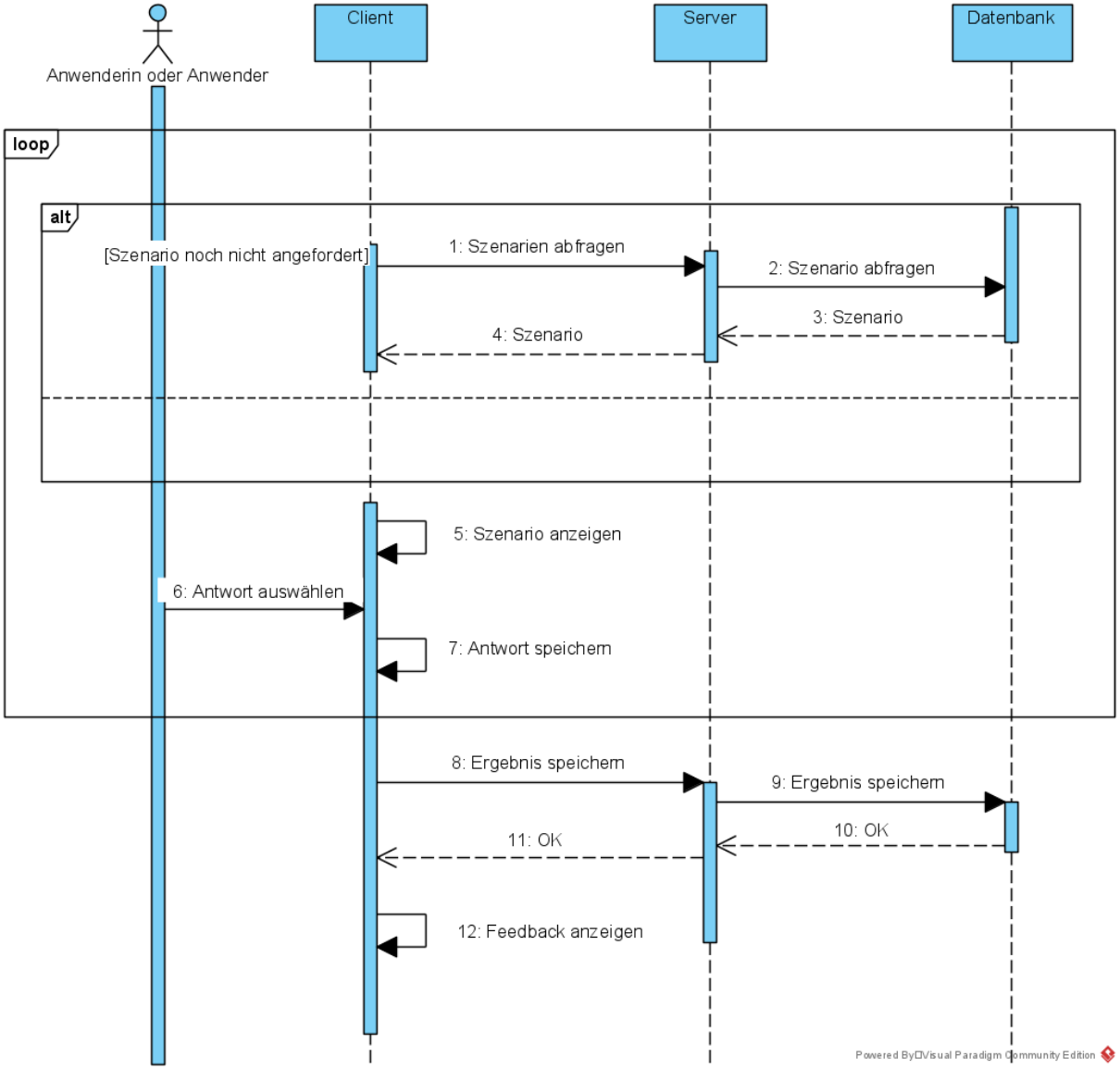


Abbildung 10: Process View Assessment

3.6 Datenmodell

Die Organisation der Daten und die Erstellung eines Modells für die Repräsentation in einer Datenbank ist vor allem für Assessments und Szenarien keine triviale Aufgabe. Wie schon in 2.2 untersucht, macht eine einheitliche Definition der Lerninhalte durchaus Sinn, um diese in einer Datenbank speichern, bzw. Ergebnisse von Übungsaufgaben vergleichen zu können. Ziel des Datenbankdesigns war ein möglichst generisches Modell zu erstellen, in dem Szenarien jeglicher Art verglichen und evaluiert, aber auch die Datenbank um neue Arten von Aufgaben erweitert werden kann, ohne bestehende Lerninhalte zu beeinflussen. Für die Realisierung des Modells sollte eine relationale Datenbank verwendet werden, um dessen Eigenschaften der Konsistenz zu nutzen.

Für die Darstellung des Modells gäbe es mehrere Möglichkeiten, wie beispielsweise UML Klassen Diagramme. Hier wurde jedoch ein Entity-Relationship-Modell (ER) Diagramm gewählt, da das Relationale Modell besser abgebildet werden kann. **Abbildung *11*** zeigt das ER-Diagramm einer vereinfachten Form des Models der Datenbank, wobei das vollständige Modell in Anhang 9.6 spezifiziert ist.

Information der IT-Security ist in Kapitel und weiter in Themen organisiert, um Funktionale Anforderungen FA101 bis FA105 der Systemanforderungen in 9.2.1 zu erfüllen. Kapitel und Themen ergeben sich aus denen des Sicherheitshandbuches für Mitarbeiterinnen und Mitarbeiter der WKO [6], sodass auch Benutzeranforderung US001 bis US006 aus 9.1 erfüllt sind.

Die „Szenario“ Entität ist eine Abstraktion aller möglichen Szenarien. Aufgabentypen jeglicher Art können problemlos neben den bereits existierenden Typen „MC_Frage“ und „Zuordnung“ von der „Szenario“ Entität abgeleitet und eingefügt werden. Assessments werden als schwache Entität modelliert, da sie von einem Benutzer, der „User“ Entität, abhängig sind. Assessments werden daher eindeutig mit dem Identifier des Benutzers, des Assessments und dem Typ des Assessments identifiziert. Typen von Assessments sind „VORTRAINING“, „NACHTRAINING“ oder „PRUEFUNG“. Auf diese Weise können die beiden Assessments eines Trainings miteinander verglichen und alle Assessments dennoch einheitlich verarbeitet werden, um den Aufwand einer getrennten Verarbeitung zu umgehen.

Ergebnisse der Szenarien werden mit dem einem booleschen Wert, ob dieses korrekt beantwortet wurde oder nicht, in der „beantwortet“ Relation in Verbindung mit einem Assessment gespeichert.

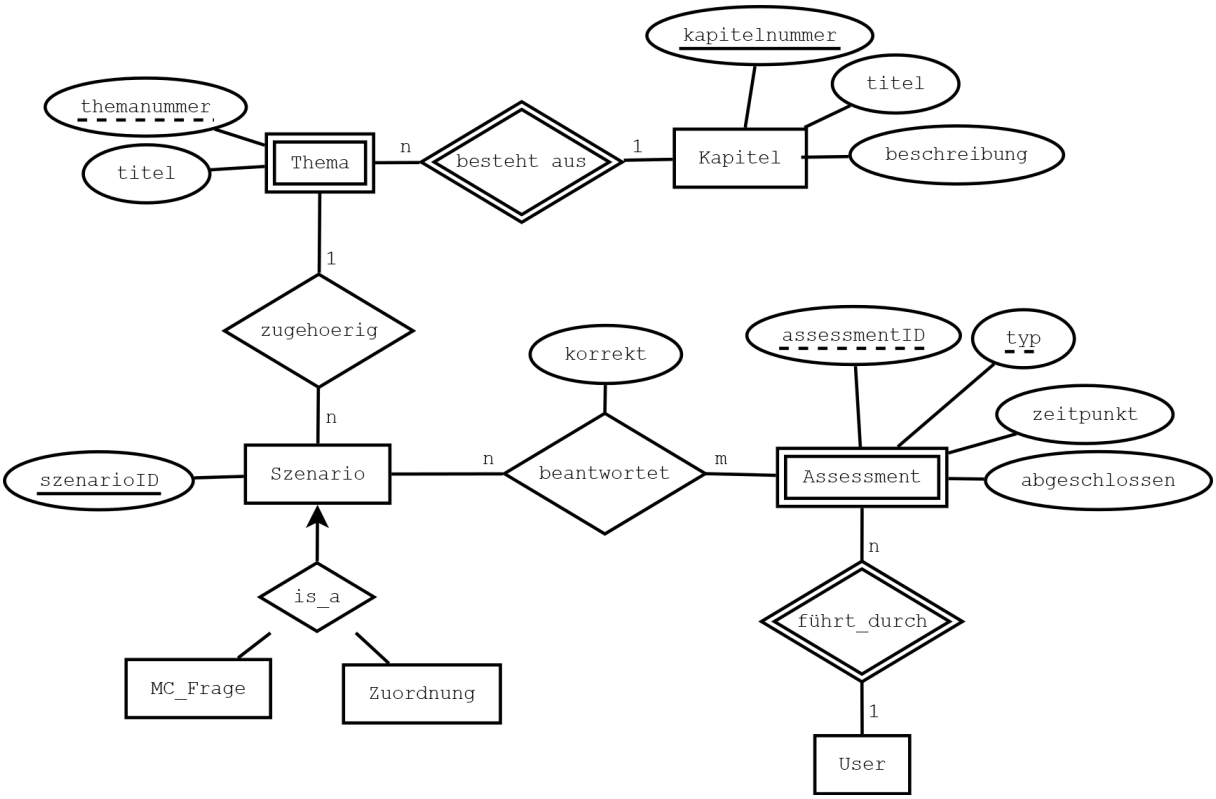


Abbildung 11: Datenmodell Übersicht

3.7 Design der REST API

Der Server stellt seine Funktionalität über eine REST API zur Verfügung, die von Clients angesteuert werden kann. Diese REST Schnittstelle ermöglicht einerseits eine geringe Kopplung zwischen Client und Server, andererseits auch die Integration von unterschiedlichen Clients, wie beispielsweise mobile App auf den Plattformen iOS oder Android. Die Spezifikation der API mit Beispieldaten kann in Anhang 9.7 gefunden werden.

4 Implementierung

In diese Kapitel werden die wichtigsten Aspekte der Implementierung beschrieben und dargestellt. Zuerst wird die verwendete Technologie präsentiert, anschließend Systemkomponenten beschrieben, die Benutzeroberfläche dargestellt und zum Schluss die Einrichtung des Systems in einer realen Umgebung erläutert. Relevanter Programmcode wurde unter 9.8 angehängt.

Die Implementierung des Prototyps enthält aus Entwicklungsgründen Konfiguration von Passwörtern und Zugangsdaten, die in einer Produktionsversion durch eine „state of the art“ Lösung für die Verwaltung von Zugangsdaten in Servern ersetzt werden muss.

4.1 Technology Stack

Wie schon in 3.5.2 erwähnt ist die Grundlage des Systems eine Containerumgebung, durch die das System auf beliebigen Betriebssystemen eingerichtet werden kann. Eine Übersicht der verwendeten Technologien ist in **Tabelle *2*:** Technologie**Tabelle *2*** zu finden. Für die Containerumgebung wird Docker verwendet. Der Server ist in der Programmiersprache Java in der Version 11 geschrieben und mithilfe des Spring Frameworks realisiert. Die Webseite für den Client wird durch den Server gehostet und ist mittels des Angular JavaScript Frameworks umgesetzt. Für die Datenbank wird das Relationale Open Source DBMS MariaDB und für den Session Storage die populäre Redis Applikation verwendet.

Anwendungszweck	**Technologie**
Container Umgebung	• Docker 20.10.15 • docker-compose 1.29.0 • docker-compose file 3.9
Server-Backend	• Openjdk 11 Docker image • Gradle 6.8.3 • Spring Boot Starter Web 2.5.0 • Spring Boot Starter Security 2.5.0 • Spring Session Data Redis 2.5.0 • Spring Boot Starter Data Redis Reactive 2.5.0 • v 2.5.0 • R2DBC Driver For MariaDB 1.0.1

	• Reactive Relational Database Connectivity SPI 0.8.4.RELEASE • Spring Boot Starter Tests 2.5.0 • Reactor Test Support 3.4.6 • Spring Security Test 5.5.0
Server-Frontend (Client)	• Angular 11.2.14 • Angular CLI 11.2.13 • HTML 5 • Sass • TypeScript 4.1.5
Datenbank	• MariaDB 10.5 Docker image
Session Storage	• Redis 6.2.4 Docker image

Tabelle 2: Technologie

4.2 Systemkomponenten

Docker Container, in denen die Systemkomponenten implementiert sind, werden mittels Docker-Compose (siehe Anhang 9.8.1) verwaltet, wobei die Konfiguration der Container in zugehörigen Dockerfiles (siehe Anhang 9.8.1) spezifiziert ist. Folgende **Abbildung *3*Abbildung *12*** zeigt die Ordnerstruktur der Implementierung.

```
├───Database\
│       ├───setup\
│       ├───szenarien\
│       ├───config.cnf
│       └───Dockerfile
│
├───SessionStore\
│       ├───redis.conf
│       └───Dockerfile
│
├───SETAServer\
│       ├───seta-frontend\
│       ├───SETAServer\
│       │       ├───java\
│       │       └───resources\
│       │               ├───keystore\
│       │               ├───static\
│       │               └───application.properties
│       ├───szenario_resources\
│       └───Dockerfile
│
├───docker-compose.yml
└───docker-compose-test-environment.yml
```

Abbildung 12: Implementierung Ordnerstruktur

4.2.1 Datenbank

Die Datenbank wird bei Initialisierung des DBMS durch die Konfigurationsdatei „Database\config.cnf“ konfiguriert. Der Ordner „Database\setup“ enthält SQL-Daten für die Errichtung von Tabellen, Nutzern, Views und das Einfügen von Kapiteln und Themen. Die SQL-Daten, die die Szenarien in die Datenbank einfügen, sind in „Database\szenarien“ gespeichert und nach Kapitel, Thema und Typ benannt. Alle SQL-Daten werden durch das „Dockerfile“ lokal in den Container kopiert und nach Initialisierung der Datenbank in alphanumerischer Reihenfolge ausgeführt.

Bilder von Multiple Choice Szenarien werden nicht direkt in der Datenbank, sondern als Ressource auf dem Webserver gespeichert. Die Datenbankeinträge enthalten lediglich den Server-Endpunkt, unter welchem das zugehörige Bild erreichbar ist.

Alle SQL-Daten für das Einrichten der Datenbank können im Anhang unter 9.9 gefunden werden.

4.2.2 Server

Der Server, gespeichert unter „SETAServer\SETAServer“, ist eine Java Applikation die mithilfe des Spring Framework implementiert ist. Wenn das Dockerfile ausgeführt wird, werden Artefakte der Webseite aus „SETAServer\seta-frontend“ und Bilder von Multiple Choice Szenarien aus „SETAServer\szenario_resources“ in den Ordner „SETAServer\SETAServer\resources\static“ des Servers kopiert. Die Webseite ist unter dem Endpunkt „/“ über das HTTPS Protokoll mit TLS Verschlüsselung erreichbar.

Die Implementierung ist, wie in **Abbildung *8*** dargestellt, in mehrere Pakete strukturiert. Der „SETAServer\SETAServer\resources“ Ordner enthält notwendige Ressourcen des Servers, darunter der Ordner „keystore“ mit notwendigen Daten für die TLS Verschlüsselung, „static“, der bei laufender Applikation öffentlich zugänglich ist und die Konfigurationsdatei „application.properties“ (siehe Anhang 9.10.1).

4.2.2.1 Authentifizierung und Autorisierung

Die Authentifizierung und Autorisierung wird durch Spring Security durchgeführt. Konfiguriert ist dieser Prozess in der „AuthenticationConfiguration“ Klasse im „configuration“ Paket. Passwörter von Benutzerkonten werden in der Datenbank in Form eines Hash Wertes gespeichert, erzeugt durch den Hash Algorithmus „BCrypt“ der von Spring Security unterstützt wird. In „AuthenticationConfiguration“ ist auch der Session Storage Container als Speicherort für Session Cookies eingerichtet, was unter der Verwendung der „Spring Session Data Redis“ Abhängigkeit mit sehr geringem Aufwand verbunden ist.

4.2.2.2 Controller Klassen

Wie schon in 3.5.4.1 erwähnt implementieren Klassen des „controllers“ Paketes die Endpunkte der REST API des Servers. Dadurch, dass die Hauptfunktionalität der Applikation das Abfragen und Einfügen von Daten in die Datenbank ist, enthält das System nur wenig Business-Logic. Aus diesem Grund sind Funktionalitäten von Endpoints meist direkt in den Controller Klassen durch einfache Aufrufe von Klassen des „repositories“ Paketes implementiert.

4.2.2.3 Repository Klassen

Klassen des „repositories“ Paketes abstrahieren Datenbankzugriffe. Verwendet wird die „R2dbcEntityTemplate“ Klasse der „Spring Boot Starter Data R2DBC“ Abhängigkeit, die das „Reactive“ Programmiermodell verwendet. Komplexe Datenbankabfragen, die nicht über Methoden der „R2dbcEntityTemplate“ umgesetzt werden können, werden als SQL Statement formuliert und über den Datenbanktreiber von „R2dbcEntityTemplate“ ausgeführt.

4.2.2.4 Service Klassen

Es gibt zwei Klassen des „services“ Paketes, die Business-Logic implementieren. Einerseits „DatabaseUserDetailsService“ um in der Datenbank gespeicherte Benutzerkonten für das Spring Security Framework zugänglich zu machen, andererseits auch die „AssessmentServices“ Klasse, die Methoden für das Abschließen und Speichern von Trainings und Prüfungen implementiert.

Daten der Implementierung des Servers sind Anhang 9.110 beigelegt.

4.2.3 Client

Der Client, implementiert mit Hilfe des Angular Frameworks, ist logisch wie in **Abbildung *9*** aufgebaut, aber Implementierung und Daten sind organisiert, wie es bei einer Angular Applikation üblich ist. Klassen klassifiziert als „model“ und „services“ sind organisiert in gleichnamigen Ordnern. Angular Components, in **Abbildung *9*** als „components“ kategorisiert, sind in Ordnern mit Namen der Komponenten angelegt mit jeweils einer HTML und Sass Datei agierend als View und einer TypeScript Date die als Controller die View manipuliert.

Komponenten „assessment“, „training-result“ und „pruefung-result“ sind so implementiert, dass sie möglichst wiederverwendbar sind und auf mehreren Seiten verwendet werden können. Der „assessment“ Angular Component enthält die Zentrale Funktionalität der Assessments, nämlich das Anzeigen und Beantworten der Szenarien und Navigationsmöglichkeiten zwischen Szenarien eines Assessments. Des Weiteren kann diese Komponente in einem Feedback-Modus verwendet werden, in dem Nutzerantworten angezeigt und die korrekten Antworten aufgelöst werden. Komponenten „training-result“ und „pruefung-result“ zeigen jeweils das Ergebnis eines Trainings und einer Prüfung. Sie werden am Ende eines Trainings oder einer Prüfung und auf den Seiten des Monitorings und der Trainings- bzw. Prüfungs-Historie eingesetzt.

Clientspezifische Daten der Implementierung können im Anhang unter 9.10 gefunden werden.

4.3 User Interface

Das User interface wird durch die Views der Angular Components erzeugt. Die Webseite ist „Responsive“, was die Nutzung sowohl auf PCs, als auch auf Mobilgeräten ermöglicht. Navigiert kann über die Navigationsleiste werden, die bei Desktop-Ansicht (**Abbildung 13**) immer auf der linken Seite angezeigt wird und in Mobil-Ansicht mit einer Schaltfläche am Ende der Seite aufklappbar ist (**Abbildung 14**, **Abbildung 15**). Sie enthält ebenfalls das Login-Formular, über das sich Benutzer an- und abmelden können.

Abbildung 16 bis **Abbildung 19** zeigen die einzelnen Phasen eines Trainings. In der Trainingsphase kann zwischen der Anzeige der Inhalte des Kapitels (**Abbildung 17**) und den Übungsaufgaben (**Abbildung 18**) mittels untenliegender Schaltflächen gewechselt werden. **Abbildung 16** zeigt eine

Multiple-Choice Frage mit einem Bild, das mit einem Klick in einem neuen Browserfenster separat geöffnet und vergrößert werden kann.

Über nummerierte Schaltflächen kann zu dem Szenario mit der ausgewählten Nummer gesprungen werden. Das ist auch bei der Anzeige von Trainings- und Prüfungsergebnissen (**Abbildung *19***, **Abbildung *21***) möglich.

Wenn Szenarien aufgelöst werden, werden korrekte Antworten grün und inkorrekte Antworten rot umrandet. Von Anwenderinnen und Anwendern ausgewählte Antworten sind hellblau eingefärbt.

Abbildung *22* zeigt die Seite des Monitorings, die nur von Benutzern mit der Trainer Rolle aufgerufen werden kann. Zuerst wird nur eine Liste des Personals mit der zuletzt abgeschlossenen Prüfung, inklusive Prüfungsergebnis angezeigt. Bei Auswahl eines Eintrags wird eine Liste der Prüfungen der ausgewählten Person angezeigt. Wenn anschließend eine Prüfung ausgewählt wird, werden dessen Ergebnis geladen.

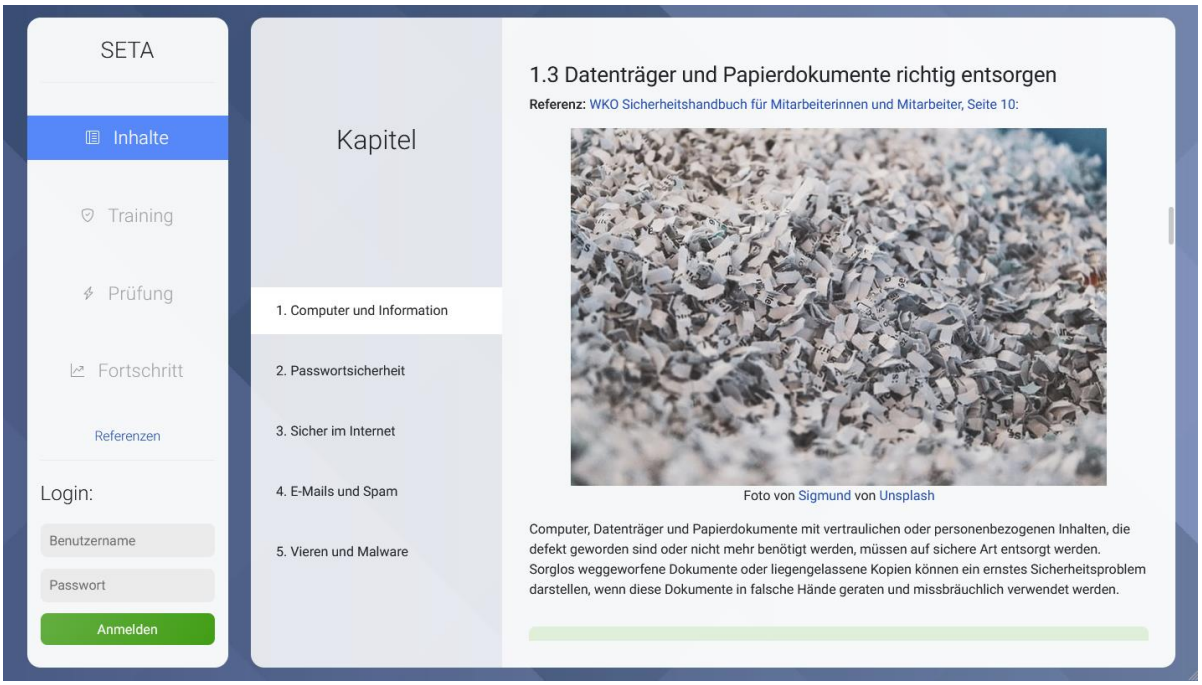


Abbildung 13:UI Inhalte

Abbildung 14: UI Inhalte Mobil

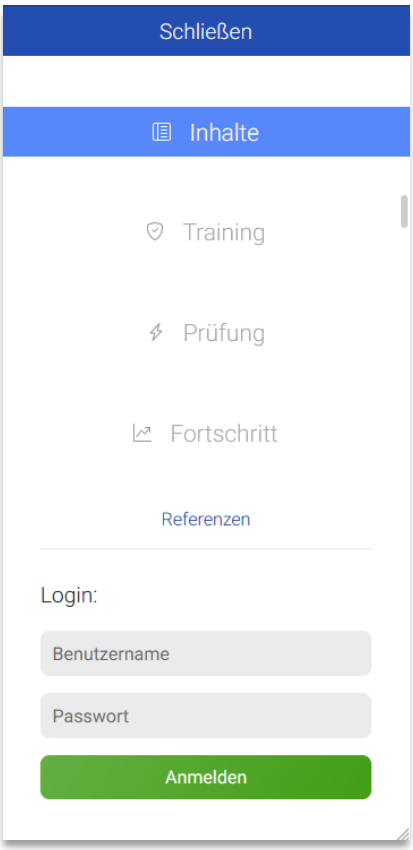


Abbildung 15:UI Navigation Mobil

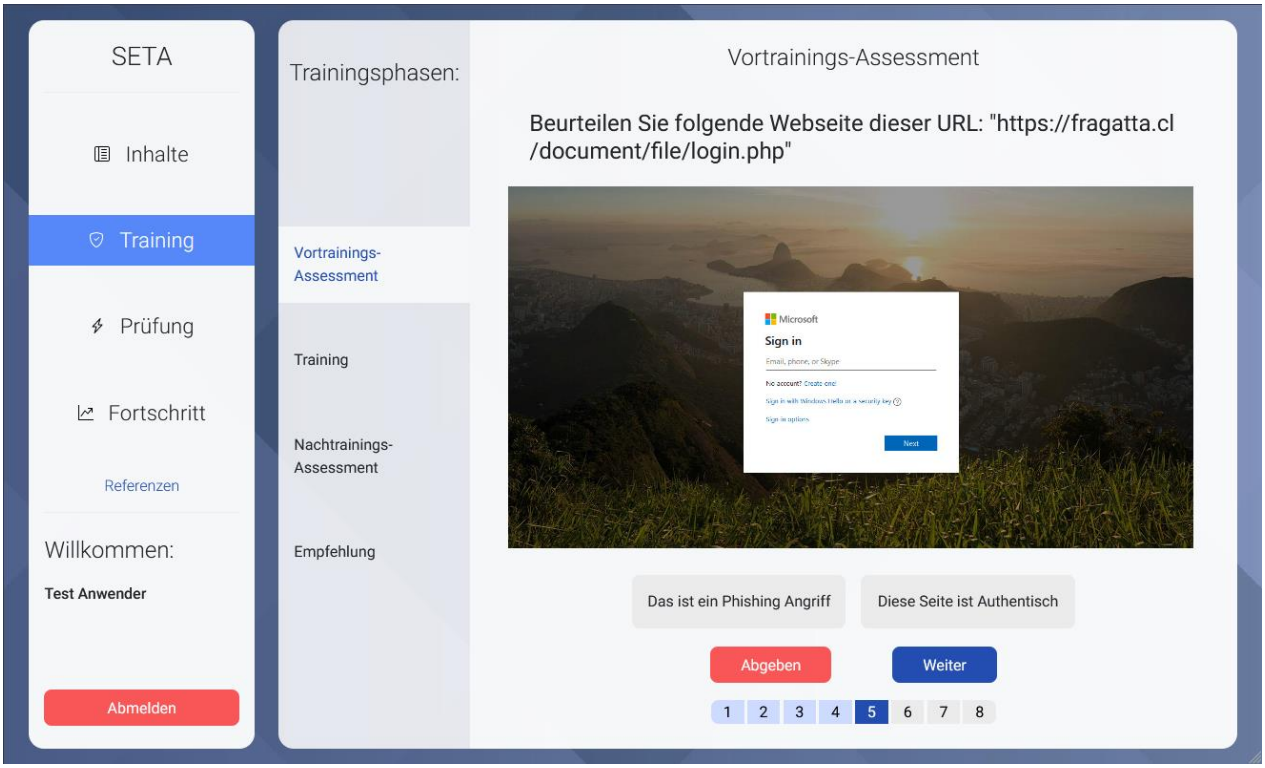


Abbildung 16: UI MC-Frage mit Bild

Abbildung 17: UI Training Inhalte

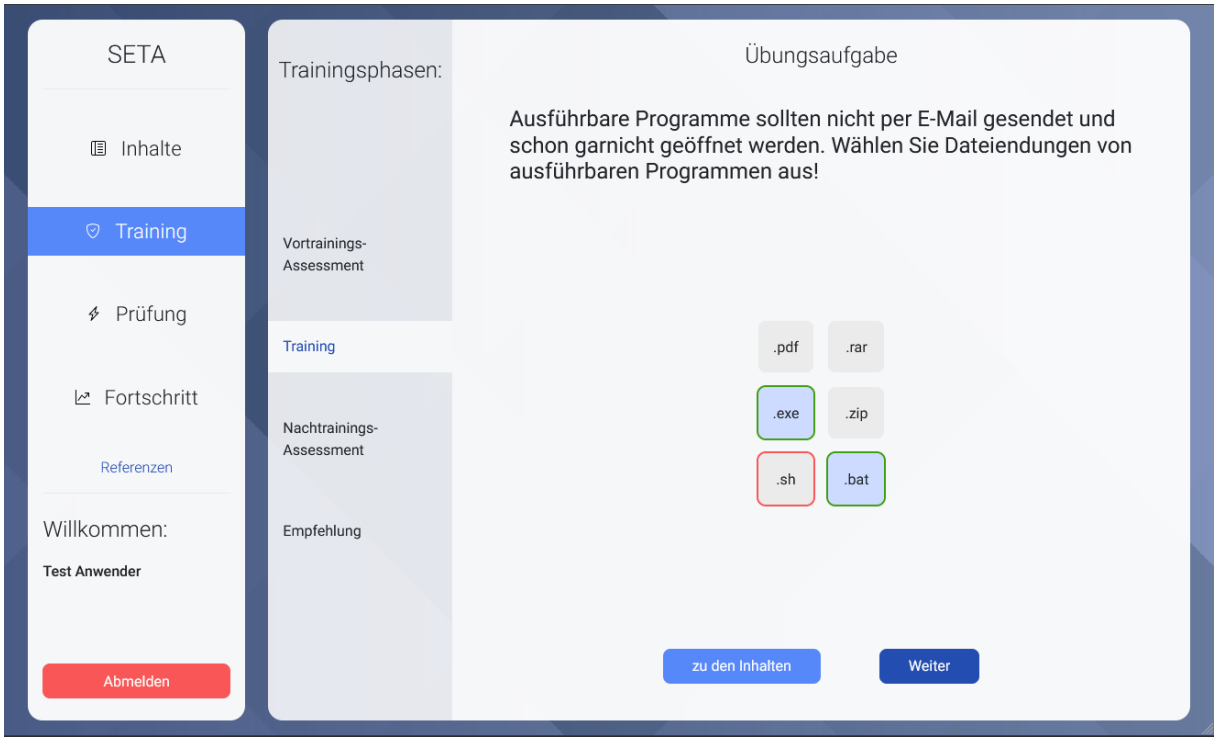


Abbildung 18: UI Übungsaufgabe

Abbildung 19: UI Trainingsergebnis

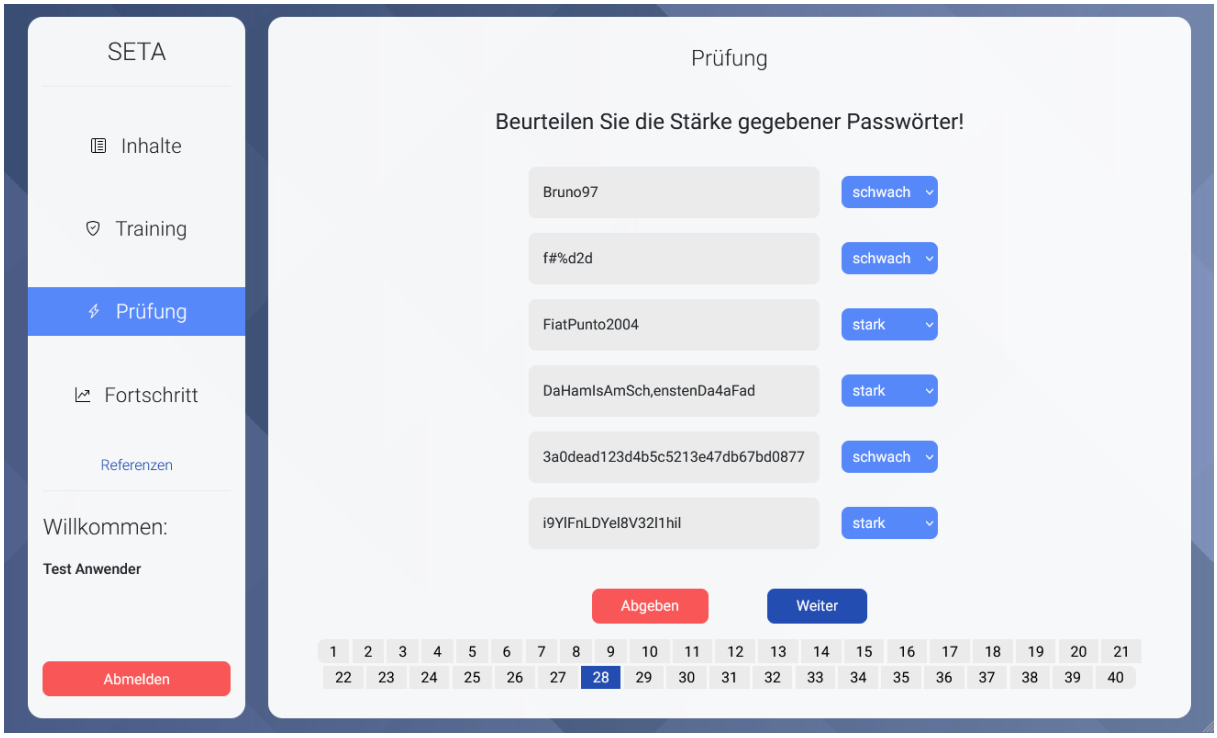


Abbildung 20: UI Zuordnung in einer Prüfung

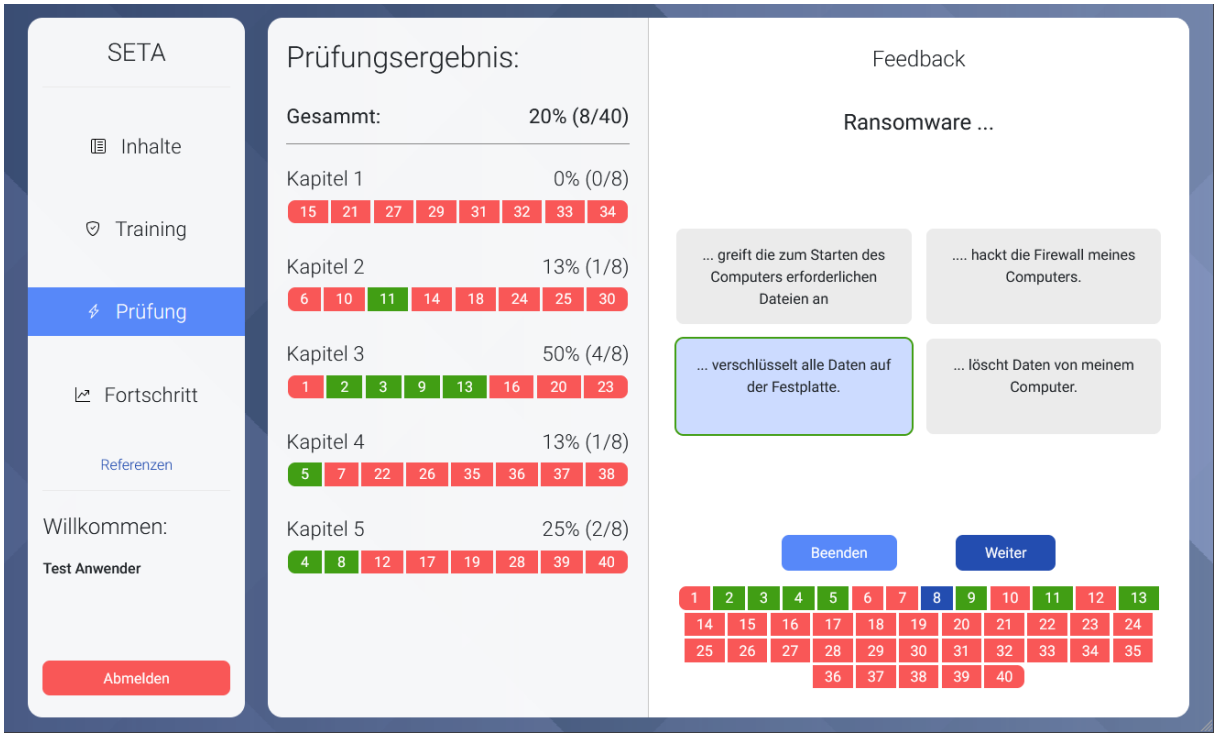


Abbildung 21: UI Prüfungsergebnis

Abbildung 22: UI Monitoring

4.4 Deployment

Um Integration bestehender Benutzerkonten weitestgehend zu ermöglichen, müssen Administratorinnen und Administratoren die der SETA Applikation selbst anlegen. Alternativ können auch neue Nutzerkonten speziell für dieses System erzeugt werden.

Für die erstmalige Installation des Systems ist eine bestehende Internetverbindung notwendig, die anschließend getrennt werden kann, da eine Lokale Netzwerkverbindung zu dem Server ausreichend ist. Aufgrund der Containerumgebung ist die Applikation von anderen Systemen strikt getrennt und kann daher unabhäng neben bestehenden Systemen, mit Beachtung der Belegung der Netzwerk-Ports eingerichtet werden (siehe **Abbildung *23***). Die SETA Webapplikation belegt den Netzwerk-Port 443, der jedoch bei der Überschneidung mit einem anderen System über die Containerumgebung geändert werden kann.

Nachdem alle Artefakte der Implementierung der Applikation auf einen Server mit installierter Container Umgebung (spezifiziert in **Tabelle *2***) kopiert wurden, kann die Applikation mit der Ausführung des Befehls „docker-compose up --build“ eingerichtet werden. Anschließend ist die Weboberfläche des Systems mit HTTPS unter dem Hostnamen des Servers erreichbar.

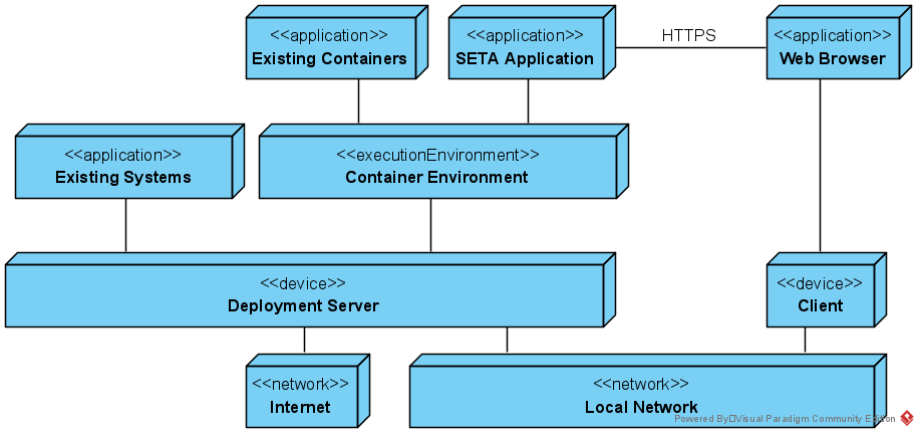


Abbildung 23: Applikation Deployment

5 Test und Evaluierung

Nachdem die Implementierung des Systems abgeschlossen war, wurde eine Reihe von Tests durchgeführt um sie auf Charakteristiken der Softwarequalität, darunter auch die korrekte Funktionsweise, zu prüfen. Dabei kamen Tests unterschiedlicher Art und Granularität zur Anwendung. Zum Schluss wurde eine Case Study erstellt, um die Applikation auf Anwendbarkeit in einem Kleinunternehmen zu prüfen.

5.1 Funktionale Tests

Wie durch das Vorgehensmodell vorgegeben und auch in **Abbildung *2*** veranschaulicht, wurde die Implementierung durch automatisierte Komponenten- und Integrationstests getestet, bevor sie durch User Tests überprüft wurde.

5.1.1 Komponententests

Komponententests überprüften die korrekte Funktionsweise der Systemkomponenten in einer sehr feinen Granularität. Abhängigkeiten und Module, die während der Tests nicht zu Verfügung standen, wurden simuliert, sodass auch nur die Teile getestet wurden, auf die sich die Tests beziehen.

Ein Beispiel für einen solchen Test ist in **Abbildung *24*** und **Abbildung *25*** dargestellt. Es soll die „AssessmentController“ Klasse des Servers und den Testfall in **Abbildung *25***, speziell die Erstellung eines neuen Trainings testen. In **Abbildung *24*** werden Komponenten, die nicht Teil des Tests sind, simuliert. In **Abbildung *25*** wird zuerst das Verhalten der simulierten Komponenten definiert, in der blau hervorgehobenen Zeile die Funktion aufgerufen, die getestet wird und zum Schluss das zurückgelieferte Ergebnis auf Korrektheit überprüft.

```java
@BeforeAll
static void beforeAll() {
    szenarioRepositoryMock = Mockito.mock(SzenarioRepository.class);
    assessmentRepositoryMock = Mockito.mock(AssessmentRepository.class);
    assessmentServicesMock = Mockito.mock(AssessmentServices.class);
    kapitelRepositoryMock = Mockito.mock(KapitelRepository.class);
    controller = new AssessmentController(szenarioRepositoryMock,
            assessmentRepositoryMock, assessmentServicesMock, kapitelRepositoryMock);
    ReflectionTestUtils.setField(controller, name: "szenarienPerAssessment", szenarienPerAssessment);
    ReflectionTestUtils.setField(controller, name: "szenarienPerKapitel", szenarienPerKapitel);
}
```

Abbildung 24: Komponententest AssessmentController Setup

```java
@Test
void startTraining_returnsValidTraining() {
    long assessmentID = 123;
    List<Long> szenarioIDs = generateSzenarioIDS( amount: szenarienPerAssessment*2);
    Mockito.when(szenarioRepositoryMock.getSzenarienOfKapitel( kapitelnummer: 1, limit: szenarienPerAssessment*2))
            .thenReturn(szenarioIDs);
    Mockito.when(assessmentRepositoryMock.insertTraining( benutzername: "testUser")).thenReturn(assessmentID);

    TrainingInitialisierung training = controller.startTraining( benutzername: "testUser", kapitelnummer: 1);
    assertNotNull(training);
    assertEquals(assessmentID, training.getAssessmentID());
    assertTrue(szenarioIDs.containsAll(training.getVortraining()));
    assertTrue(szenarioIDs.containsAll(training.getNachtraining()));
    int totalSzenarienReturned = training.getVortraining().size() +
            training.getNachtraining().size();
    assertEquals(szenarioIDs.size(), totalSzenarienReturned);
}
```

Abbildung 25: Komponententest AssessmentController Testfall

5.1.2 Integrationstest

Integration und das Zusammenspiel der Komponenten werden mittels End-zu-End Tests überprüft. Im Vergleich zu Komponententests zielt diese Art auf ein höheres Abstraktionslevel ab. Bevor diese Tests ausgeführt werden können, muss eine Testumgebung mit Testdaten eingerichtet sein.

Mithilfe der Containerumgebung kann die Testumgebung über den Befehl „docker-compose -f docker-compose-test-environment.yml up --build“ im Ordner der Implementierung gestartet werden. Anschließend kann das System durch End-zu-End Tests geprüft werden.

Ein Beispiel für einen End-zu-End Test ist in **Abbildung *26*** zu sehen. Hierbei soll die Funktionalität des Login-Mechanismus getestet werden. Es werden HTTPS Anfragen an die REST API des Servers der Testumgebung gesendet und Antworten überprüft. Hierbei wird die Integration der Komponenten des Servers, der Datenbank und des Session Storage geprüft.

```java
@Test
void loginWithInvalidCreds_status403Returned() {
    HttpRequest request = HttpRequest
            .newBuilder()
            .uri(URI.create(baseUrl + "/login"))
            .POST(HttpRequest.BodyPublishers.ofString( body: ""))
            .header(HttpHeaders.CONTENT_TYPE,  value: "application/x-www-form-urlencoded")
            .build();
    try {
        HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
        assertEquals( expected: 403, response.statusCode());
    } catch (IOException | InterruptedException e) {
        e.printStackTrace();
        fail();
    }
}

@Test
void loginWithValidCreds_statusOKReturned() {
    HttpRequest request = HttpRequest
            .newBuilder()
            .uri(URI.create(baseUrl + "/login"))
            .POST(HttpRequest.BodyPublishers.ofString( body: "benutzername=anwender&passwort=test"))
            .header(HttpHeaders.CONTENT_TYPE,  value: "application/x-www-form-urlencoded")
            .build();
    try {
        HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
        assertEquals( expected: 200, response.statusCode());
    } catch (IOException | InterruptedException e) {
        e.printStackTrace();
        fail();
    }
}
```

Abbildung 26: End-zu-End Test Login

5.2 Case Study

In dieser Case Study wird ein fiktives Unternehmen beschrieben und die SETA Applikation simuliert eingerichtet. In diesem anwendungsorientierten Prozess sollen Probleme und Komplikationen erkannt werden, die in der bisherigen Evaluierung nicht beobachtet werden konnten. Eine bessere Alternative wäre, die Case Study in einem realen Unternehmen durchzuführen, da die Anwendung des Systems tatsächlich stattfindet und eine genauere Beurteilung möglich wäre, was leider aufgrund der COVID-19 Pandemie nicht möglich war. Im Zuge einer Case Study in einem realen Unternehmen wären auch die Durchführung von ausführlicheren Usability-Tests möglich gewesen, die sich aufgrund der momentanen Situation auf eine einfache Systeminspektion beschränken. Das fiktive Unternehmen wurde auf Basis eines Unternehmens der Unternehmensberatung erstellt, das für diese Case Study annonymisiert wurde. Namen des Unternehmens und der erwähnten Angestellten sind frei erfunden.

5.2.1 Beschreibung des Unternehmens

Das Unternehmen „AB Consulting“ wurde im Jahr 2016 von Hans P. als GmbH gegründet, ist ansässig in Wien und hat 30 Angestellte. Die Kernaufgabe des Unternehmens ist die Beratung von Führungskräften in strategischen Fragestellungen im Energiesektor und öffentlichen Bereich. Die Mehrheit der Mitarbeiterinnen und Mitarbeiter hat ihren fachlichen Hintergrund überwiegend in den Wirtschaftswissenschaften. Ebenfalls werden Supportprozesse wie Buchhaltung, Rechtsberatung, Assistenzen, Personalmanagement von weiteren Mitarbeiterinnen und Mitarbeitern durchgeführt. Die Hauptaufgaben der Angestellten werden am PC erledigt und umfassen beispielsweise das Lesen und Schreiben von E-Mails oder Arbeiten mit Microsoft Office Produkten, die sehr typisch für Managementberatungen sind. Administration der IT wird von den Angestellten Petra Mayer und Herbert Lang übernommen.

5.2.2 IT-Infrastruktur

Die Hardware des Unternehmens umfasst im Allgemeinen einen PC für jeden Angestellten und einen zentralen Server. Auf den Arbeitsrechnern ist das Betriebssystem Windows 10 und auf dem Server Windows Server 2019 installiert. Das Active Directory Domain Service von Microsoft ist auf dem Server eingerichtet, der als zentraler Speicherort für Daten des Unternehmens dient. Außerdem verwaltet er als Domain Controller mittels LDAP die Benutzerkonten des Unternehmens. Ebenfalls ist Microsoft Exchange Server für den E-Mail-Verkehr installiert. Auf dem Arbeitsrechner jeder Mitarbeiterin und jedes Mitarbeiters ist Microsoft Office 365 und die für den Arbeitsbetrieb notwendige Software installiert.

5.2.3 Einrichten der Applikation

Da der Geschäftsführer viel Wert auf das Vertrauen der Kunden legt, beschließt er, dass alle Angestellten ein regelmäßiges Training der IT-Security Kenntnisse absolvieren müssen, um Sicherheitsvorfällen und dem Verlust des Vertrauens der Kunden vorzubeugen. Beide IT-Administratoren Petra Mayer und Herbert Lang werden beauftragt, hierfür eine kosten- und aufwandeffiziente Lösung zu finden und diese einzurichten.

Planung und Analyse ergaben, dass die SETA Applikation dieser Arbeit eingesetzt und auf dem Zentralen Server eingerichtet wird. Herbert Lang sei für die Installation des Systems und Petra Mayer für das Anlegen der Benutzerkonten verantwortlich. Nachdem diese Aufgaben erledigt sind, übernehmen beide Administratoren die Rolle einer Trainerin und eines Trainers, die den Trainingsfortschritt der Mitarbeiterinnen und Mitarbeitern überwachen und bei Fragen Hilfestellung leisten. Es wird auch ein Sicherheitskonzept ausgearbeitet, das allen Mitarbeiterinnen und Mitarbeitern den jährlichen Abschluss einer Überprüfung ihrer Kompetenzen der IT-Sicherheit vorschreibt, wobei mindestens 70% der Fragen korrekt beantwortet sein müssen.

Da es sich bei der SETA Applikation um einen Prototyp handelt, wird sie von Herbert Lang in einer Virtuellen Umgebung auf dem Server eingerichtet, sodass die Applikation auf der Ebene des Betriebssystems von dem bestehenden System getrennt ist. Hier kommt das Windows Subsystem Linux zum Einsatz, mithilfe dessen das Linux-basierte Betriebsystem Ubuntu aufgesetzt, die Containerumgebung laut **Tabelle *2*** installiert und die SETA Applikation wie in 4.4 beschrieben eingerichtet wird.

Aufgrund der Funktionsweise der Authentifizierung des Prototyps der SETA Applikation und aus Sicherheitsgründen werden von Petra Mayer für jeden Angestellten neue Benutzerkonten speziell für dieses System erstellt und mit der Microsoft Access Software, die als Teil von Microsoft Office 365 verfügbar ist, in die Datenbank der Applikation eingefügt.

5.2.4 Ergebnisse der Case Study

Die SETA-Applikation konnte in dem Unternehmen ohne hohe Kosten und Aufwand verwendet werden. Sie wurde innerhalb einer Virtuellen Umgebung in das bestehende System des Unternehmens integriert. Da die SETA-Applikation für die Authentifizierung und Autorisierung von Benutzern dessen interne Datenbank als Quelle für Benutzerkonten heranzieht, war es nicht möglich bestehende Konten des Domain Controllers des Unternehmens einzubinden. Benutzerkonten müssen in die Datenbank importiert werden, wobei das System des Unternehmens keinen passenden Export bereitstellt. Mitarbeiterinnen und Mitarbeiter konnten Trainings in IT-Security erfolgreich absolvieren.

6 Diskussion der Ergebnisse

6.1 Evaluierung der Implementierung

Die SETA Applikation konnte mithilfe des V-Modells, wie in 1.5 beschrieben, umgesetzt werden. Es wurden Benutzeranforderungen und anschließend Systemanforderungen formuliert, ein Design der Architektur entwickelt und das System Implementiert. Die Funktionalen Komponententests und Integrationstests wurden ohne Fehler ausgeführt. Systemanforderungen konnten durch einen User Tests manuell überprüft und Userstories der Benutzeranforderungen mit deren zur Überprüfung formulierten Fragen validiert werden.

Durch Verwendung einer Architektur die in Schichten strukturiert ist, kann das System einfach erweitert und gewartet werden. Auch unter Einsatz von Design Patterns wie der Model View Controller Pattern, wurde vor allem in der Implementierung der Benutzeroberfläche komplexe Funktionalität geteilt und getrennt behandelt, was für eine höhere Modularität und weniger Komplexität in Teilkomponenten sorgt.

Die Datenbank und das Datenmodell sind so organisiert, dass unterschiedliche Arten von Aufgaben und Lernmaterialien des didaktischen Modells gespeichert werden können und deren Ergebnisse vergleichbar sind. Mit diesem Modell kann das System um neue Arten von Aufgaben und Lernmaterialien erweitert werden.

6.2 Diskussion der Ergebnisse der Case Study

Die Case Study hat gezeigt, dass ohne kostenintensive Anschaffungen oder unverhältnismäßig hohen Aufwand, die SETA Applikation problemlos in die bestehende IT-Infrastruktur eines Kleinunternehemens eingepflegt werden kann. Die Applikation baut auf Open Source und kostenfreien Technologien auf, sodass bei Verwendung keine Lizenzkosten entstehen. Kosten des Einrichtens und des Betreibens entstehen nur in Bezug auf die IT-Infrastruktur, die in Unternehmen dieser Größe meist schon vorhanden ist. Durch Virtualisierung und Container-Technologien sind einfaches Management, Integration und Deployment auf bestehender Server-Hardware möglich, was

von diesem System vorteilhaft genutzt wird. Informationen und Übungen zur IT-Security werden von der Applikation nativ Angeboten und müssen durch die Systemadministratoren nicht selbst erstellt werden, wodurch der Aufwand des Einrichtens und Betreibens des Trainingsystems drastisch reduziert wird. Außerdem wird ein Trainingsmodell geboten, welches den Wissensstand der Mitarbeiterinnen und Mitarbeiter transparent macht und Personen, die die Rolle einer Trainerin oder eines Trainers einnehmen ermöglicht, die anderen Anwenderinnen und Anwender des Systems zu fördern und zu unterstützen.

Im Rahmen dieser Case Study wurde jedoch ein nachteiliger Aspekt erkennbar. Die momentane Methode der Authentifizierung und Autorisierung verwendet Informationen von Benutzern, die in der Datenbank gespeichert sind. Diese Vorgehensweise restriktiert die Möglichkeit der Integration mit bestehenden Authentifizierungslösungen, deren Benutzerkonten nicht exportiert und in die Datenbank importiert werden können.

Ein Aspekt, der zwar durch User Tests in der Evaluierung der Implementierung geprüft wurde, aber nicht ausreichend evaluiert werden konnte, ist die Usability der Benutzeroberfläche. Wie oben schon erwähnt, konnte dieser Aspekt in der Case Study nicht geprüft werden, da es sich um eine simulierte Anwendung handelte.

7 Conclusio und Ausblick

Das Trainingsmodell, die Datenbank, Server- und Clientarchitektur einer SETA Webapplikation zu entwerfen, ist eine nicht-triviale Aufgabe und erfordert ausführliche Planung und Evaluierung. In dieser Arbeit wurden unterschiedliche Systemarchitekturen von E-Learning Systemen betrachtet und darauf aufbauend das Design einer möglichen Architektur und ein zugehöriger Prototyp entwickelt, der von Kleinunternehmen zu Trainingsszwecken der Angestellten bezüglich IT-Security eingesetzt werden kann.

Zuerst wurden Kerncharakteristiken von Trainings untersucht, um daraus ein effektives Trainingsmodell ableiten zu können. Anschließend sind unterschiedliche Konzepte von Architekturen von E-Learning Systemen recherchiert worden, die vorteilhaft eingesetzt werden können. Durch Anwendung des V-Modells von Boehm wurden Vorgehensweisen der Anforderungsanalyse, Implementierung und Verifizierung definiert, die zum Erfolg der Entwicklung beitrugen.

Es wurden Benutzergruppen klassifiziert und Anforderungen des Systems formuliert, die nach Prüfung durch im Vorfeld definierten Tests von der Implementierung des Prototyps erfüllt wurden. Fokus bei der Entwicklung der präsentierten Lösung waren geringe Kosten und Aufwand um die Einsetzbarkeit in Kleinunternehmen zu Steigern, was mittels einer Case Study evaluiert wurde.

Zukünftig könnte das System dahingehend erweitert werden, dass in Kleinunternehmen bestehende Systeme der Benutzerverwaltung und Autorisierung in die SETA Applikation integriert, bzw. Authentifizierungs und Autorisierungsfunktionalität der bestehenden Systeme von der SETA Applikation verwendet werden kann. Des Weiteren wäre die Erweiterung um Interaktive Übungsaufgaben oder die Artificial Intelligence unterstützte Analyse der Trainings von Interesse.

8 Literaturverzeichnis

[1] WIK - Wissenschaftliches Institut für Infrastruktur und Kommunikationsdienste GmbH, „IT Sicherheit in KMU," [Online]. Available: https://www.wik.org/regulierung/post-und-logistik/it-sicherheit-in-kmu. [Zugriff am 28 Mai 2021].

[2] Wirtschaftskammer Österreich, „Klein- und Mittelbetriebe (KMU): Definition - WKO.at," 28 Mai 2021. [Online]. Available: https://www.wko.at/service/zahlen-daten-fakten/KMU-definition.html. [Zugriff am 28 Mai 2021].

[3] Bundesministerium für Digitalisierung und Wirtschaftsstandort, „KMU im Fokus 2019," Bundesministerium für Digitalisierung und Wirtschaftsstandort, Wien, 2020.

[4] Kaspersky Labs GmbH, „media.kaspersky.com," [Online]. Available: https://media.kaspersky.com/de/enterprise-security/KL_Human%20factor_main%20threats_datasheet.pdf. [Zugriff am 29 Mai 2021].

[5] M. Wilson und . J. Hash, „Building an Information Technology Security Awareness and Training Program, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD," 2003. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=151287. [Zugriff am 29 Mai 2021].

[6] Wirtschaftskammer Österreich, „IT-Sicherheitshandbuch für Mitarbeiter - WKO.at," 12 November 2019. [Online]. Available: https://www.wko.at/site/it-safe/mitarbeiter-handbuch.html. [Zugriff am 2021 Mai 30].

[7] M. Grande, „Vorgehensmodelle," in *100 Minuten für Anforderungsmanagement*, Vieweg+Teubner, 2011, pp. 109-112.

[8] P. Kruchten, „Architectural Blueprints—The "4+1" View Model of Software Architecture," *IEEE Software 12,* pp. 42-50, November 1995.

[9] C. W. Coultas, R. Grossman und E. Salas, „Design, Delivery, Evaluation, and Transfer of Training Systems," in *Handbook of Human Factors and Ergonomics*, Hoboken, NJ, USA, John Wiley & Sons, 2012, pp. 490-533.

[10] I. Ruano, P. Cano, J. Gamez und J. Gomez, „Advanced LMS Integration of SCORM Web Laboratories," *IEEE Access,* p. 6352–6363, 7 July 2016.

[11] L. Xiaofeng und W. Chanle, „Building a New-generation Learning Resources Management System Based on SLO and SOA Technology," in *2010 2nd International Conference on Computer Engineering and Technology*, Chengdu, China, 2010.

[12] D. Dagger, A. O'Connor, S. Lawless, E. Walsh und V. P. Wade, „Service-Oriented E-Learning Platforms: From Monolithic Systems to Flexible Services.," *IEEE Internet Computing 11.3,* pp. 28-35, May 2007.

[13] M. P. Papazoglou, „Service -Oriented Computing: Concepts, Characteristics and Directions," in *International Conference on Web Information Systems Engineering* , Rome, Italy, 2003.

[14] D. Booth, H. Haas, F. McCabe und E. Newcomer , „Web Services Architecture," W3C, 11 Februar 2004. [Online]. Available: https://www.w3.org/TR/ws-arch/#service_oriented_architecture. [Zugriff am 12 Juni 2021].

[15] P. Mell und T. Grance, „The NIST Definition of Cloud, Special Publication 800-145," Sptember 2011. [Online]. Available: https://dl.acm.org/doi/pdf/10.5555/2206223. [Zugriff am 12 Juni 2021].

[16] A. Masud und X. Huang, „An E-learning System Architecture based on Cloud Computing.," *International Journal of Social and Human Sciences. 6.,* pp. 736 - 740, 21 Februar 2012.

[17] P. Hendradi, M. Khanapi und S. N. Mahfuzah, „Cloud Computing-Based E-Learning System Architecture in Education 4.0," *Journal of Physics. Conference Series vol. 1196.1 ,* p. 12038, 2019.

9 Anhang

9.1 Benutzeranforderungen (User-Stories):

9.1.1 Information bezüglich sicheren Umgangs mit Computern und Daten

Identifier: US001
Beschreibung: Als Angestellte oder Angestellter möchte ich Informationen bezüglich sicheren Umgangs mit Computern abfragen, um Risiken des Missbrauchs und Verlust von Daten zu vermeiden.
Überprüfung:

- Kann ich Informationen bezüglich sicheren Umgangs mit Computern abfragen?

9.1.2 Informationen über Passwörter

Identifier: US002
Beschreibung: Als Angestellte oder Angestellter möchte ich Informationen über das Wählen sicherer Passwörter und deren Umgang abfragen, um Missbrauch und unberechtigte Zugriffe zu verhindern.
Überprüfung:

- Kann ich erfahren, wie sichere Passwörter gewählt werden?
- Kann ich erfahren, wie ich mit Passwörtern umgehe?

9.1.3 Informationen über Sicherheitsaspekte des Internets

Identifier: US003
Beschreibung: Als Angestellte oder Angestellter möchte ich Informationen über Sicherheitsmechanismen des Internets abfragen, um Bedrohungen zu erkennen.
Überprüfung:

- Kann ich Informationen zu Sicherheitsmechanismen des Internets abfragen?
- Kann ich Informationen zu Bedrohungen des Internets abfragen?

9.1.4 Informationen über E-Mails und Spam

Identifier: US004
Beschreibung: Als Angestellte oder Angestellter möchte ich Informationen über Sicherheitsrisiken von E-Mails und Spam abrufen, um Sicherheitsbedrohungen erkennen zu können.
Überprüfung:

- Kann ich Informationen zu Sicherheitsrisiken in E-Mails abfragen?

9.1.5 Informationen über Schadprogramme

Identifier: US005
Beschreibung: Als Angestellte oder Angestellter möchte ich Informationen zu Schadprogrammen und Verhaltensweisen bei deren Auftreten abfragen, um Risiken und etwaige Verluste zu minimieren.
Überprüfung:

- Kann ich Informationen zu Schadsoftware ansehen?
- Kann ich Verhaltensregeln bei einer Infektion mit Schadsoftware erfahren?

9.1.6 WKO Inhalte verwenden

Identifier: US006
Beschreibung: Als Administrator oder Administratorin der SETA Applikation möchte ich die Inhalte des IT-Sicherheitshandbuches für Mitarbeiterinnen und Mitarbeiter der WKO [6] verwenden, um Personalkosten für die eigenständige Erstellung zu minimieren.
Überprüfung:

- Müssen Inhalte von Administratoren selbst erzeugt werden?

9.1.7 Monitoring

Identifier: US007
Beschreibung: Als Trainer oder Trainerin möchte ich die sicherheitsspezifische Kompetenzen der Angestellten überwachen, um etwaige Mängel erkennen zu können.
Überprüfung:

- Kann ich den Wissensstand der Angestellten überprüfen?

9.2 Systemanforderungen

9.2.1 Bereitstellen von IT-sicherheitsrelevanter Information

Beschreibung: IT-sicherheitsrelevante Informationen sind inhaltlich in Kapitel eingeteilt. IT-sicherheitsrelevante Informationen sind innerhalb eines Kapitels in Themen organisiert.
Voraussetzungen: Keine
Funktionale Anforderungen:

- **FA101:** Die Webapplikation bietet eine Übersicht aller vorhandenen Kapitel.
- **FA102:** Wenn ein Kapitel ausgewählt wird, werden zugehörige Themen mit deren IT-sicherheitsrelevanten Inhalten angezeigt.
- **FA103:** Anwenderinnen oder Anwender können innerhalb eines Kapitels über Themen navigieren.
- **FA104:** Anwenderinnen oder Anwender können bei ausgewähltem Kapitel ein Training starten.
- **FA105:** Anwenderinnen oder Anwender können bei ausgewähltem Kapitel eine Überprüfung starten.

9.2.2 Multiple-Choice Szenarien

Beschreibung: Multiple Choice Szenarien bestehen aus einer Frage, die optional ein Bild behandeln kann und Antwortmöglichkeiten, von denen eine oder mehrere richtig sein können.
Voraussetzungen:

1. Anwenderinnen oder Anwender haben sich mit ihrem Benutzerkonto angemeldet.
2. Anwenderinnen oder Anwender haben ein Training oder eine Überprüfung gestartet.

Funktionale Anforderungen:

- **FA201:** Bei einem Multiple Choice Szenario werden den Anwenderinnen oder Anwendern eine Frage und mehrere Antwortmöglichkeiten angezeigt.
- **FA202:** Die Anwenderinnen oder Anwender können eine oder mehrere Antwortmöglichkeiten auswählen.
- **FA203:** Die Anwenderinnen oder Anwender können ihre Auswahl bestätigen und die Frage beantworten.

9.2.3 Zuordnungsszenarien

Beschreibung: Zuordnungen sind Szenarien, in denen Begriffe ihrer Beschreibung gegenübergestellt sind. Die Anwenderinnen und Anwender müssen jeden Begriff seiner Beschreibung zuordnen.
Voraussetzungen:

1. Anwenderinnen oder Anwender haben sich mit ihrem Benutzerkonto angemeldet.
2. Anwenderinnen oder Anwender haben ein Training oder eine Überprüfung gestartet.

Funktionale Anforderungen:

- **FA301:** Bei einem Zuordnungsszenario wird den Anwenderinnen oder Anwendern eine Liste von Beschreibungen angezeigt.
- **FA302:** Jede Beschreibung hat ein Drop Down Menü mit einer Liste von Begriffen.
- **FA303:** Anwenderinnen oder Anwender können für jede Beschreibung einen Begriff auswählen.

- **FA304:** Ein Begriff kann mehrfach ausgewählt werden.

9.2.4 Trainings

Beschreibung: Anwenderinnen und Anwender können Trainings zu einem ausgewählten Kapitel absolvieren. Ein Training ist ein geführter Ablauf von vier Phasen, in denen mithilfe von didaktischen Methoden IT-sicherheitsrelevantes Wissen vermittelt und Kompetenzen gefördert werden sollen.
Voraussetzungen:

1. Anwenderinnen oder Anwender haben sich mit ihrem Benutzerkonto angemeldet.
2. Anwenderinnen oder Anwender haben ein Kapitel ausgewählt.

Funktionale Anforderungen:

- **FA401:** Anwenderinnen oder Anwender können ein Training zu dem ausgewählten Kapitel starten.
- **FA402:** Während des Trainings werden dem Anwender alle Phasen und der Fortschritt des Trainings angezeigt.
- **FA403:** Zu Beginn des Trainings wird der Trainingsablauf und dessen Phasen erklärt.
- **FA404:** In der ersten Phase des Trainings müssen die Anwenderinnen oder Anwender mehrere, zufällig ausgewählte Szenarien durchführen.
- **FA405:** Nachdem die Szenarien der ersten Phase abgeschlossen wurden, wird eine Feedback-Übersicht angezeigt, inkorrekt beantwortete Szenarien und deren Themen hervorgehoben.
- **FA406:** In der zweiten Phase werden die bereitgestellten IT-sicherheitsrelevanten Informationen und Inhalte zu jedem Thema des Kapitels angezeigt.
- **FA407:** In der zweiten Trainingsphase wird eine Reihe von Übungsszenarien bereitgestellt, in denen die Anwenderinnen oder Anwender die vermittelten Informationen praktisch anwenden und üben können.
- **FA408:** Jedes Übungsszenario wird nach Beantwortung korrekt aufgelöst.
- **FA409:** Die Übungsszenarien werden wiederholt angezeigt, bis die Anwenderinnen oder Anwender das Üben über die zugehörige Schaltfläche beenden.
- **FA410:** In der dritten Phase des Trainings müssen die Anwenderinnen oder Anwender erneut Szenarien absolvieren.
- **FA411:** Nachdem die Szenarien der dritten Phase abgeschlossen wurden, wird eine Feedback-Übersicht angezeigt, inkorrekt beantwortete Szenarien und deren Themen hervorgehoben.
- **FA412:** In der vierten Phase des Trainings wird eine Übersicht der beantworteten Szenarien, ein Vergleich des Erfolges der ersten und dritten Phase und Empfehlungen bezüglich Verbesserungsmöglichkeiten angezeigt.
- **FA413:** Die bereitgestellten Szenarien der ersten und dritten Phase überschneiden sich nicht.
- **FA414:** Für jedes Training werden die Szenarien in den unterschiedlichen Phasen jeweils zufällig ausgewählt.

9.2.5 Prüfung

Beschreibung: Anwender können ihre IT-sicherheitsrelevanten Kompetenzen anhand von Prüfungen überprüfen.
Voraussetzungen:

1. Anwenderinnen oder Anwender haben sich mit ihrem Benutzerkonto angemeldet.

Funktionale Anforderungen:

- **FA501:** Anwenderinnen oder Anwender können eine Prüfung starten.
- **FA502:** Eine Prüfung kann jederzeit abgebrochen werden.
- **FA503:** Bei einer Prüfung müssen die Anwenderinnen oder Anwender mehrere Szenarien durchführen.
- **FA504:** Bei der Prüfung werden der Fortschritt und die Anzahl der Szenarien angezeigt.
- **FA505:** Nachdem alle Szenarien der Prüfung beantwortet wurden, wird eine Erfolgsstatistik mit richtigen und falschen Antworten angezeigt.

9.2.6 Trainings- und Prüfungshistorie

Beschreibung: Anwender können Ergebnisse von vergangenen Prüfungen und Trainings einsehen.
Voraussetzungen:

1. Anwenderinnen oder Anwender haben sich mit ihrem Benutzerkonto angemeldet.

Funktionale Anforderungen:

- **FA601:** Anwenderinnen oder Anwendern wird eine Übersicht über abgeschlossene Trainings und Prüfungen angezeigt.
- **FA602:** Ein abgeschlossenes Training kann ausgewählt, dessen Ergebnis und Empfehlung angezeigt werden.
- **FA603:** Eine abgeschlossene Prüfung kann ausgewählt und dessen Ergebnis angezeigt werden.

9.2.7 Trainer Rolle

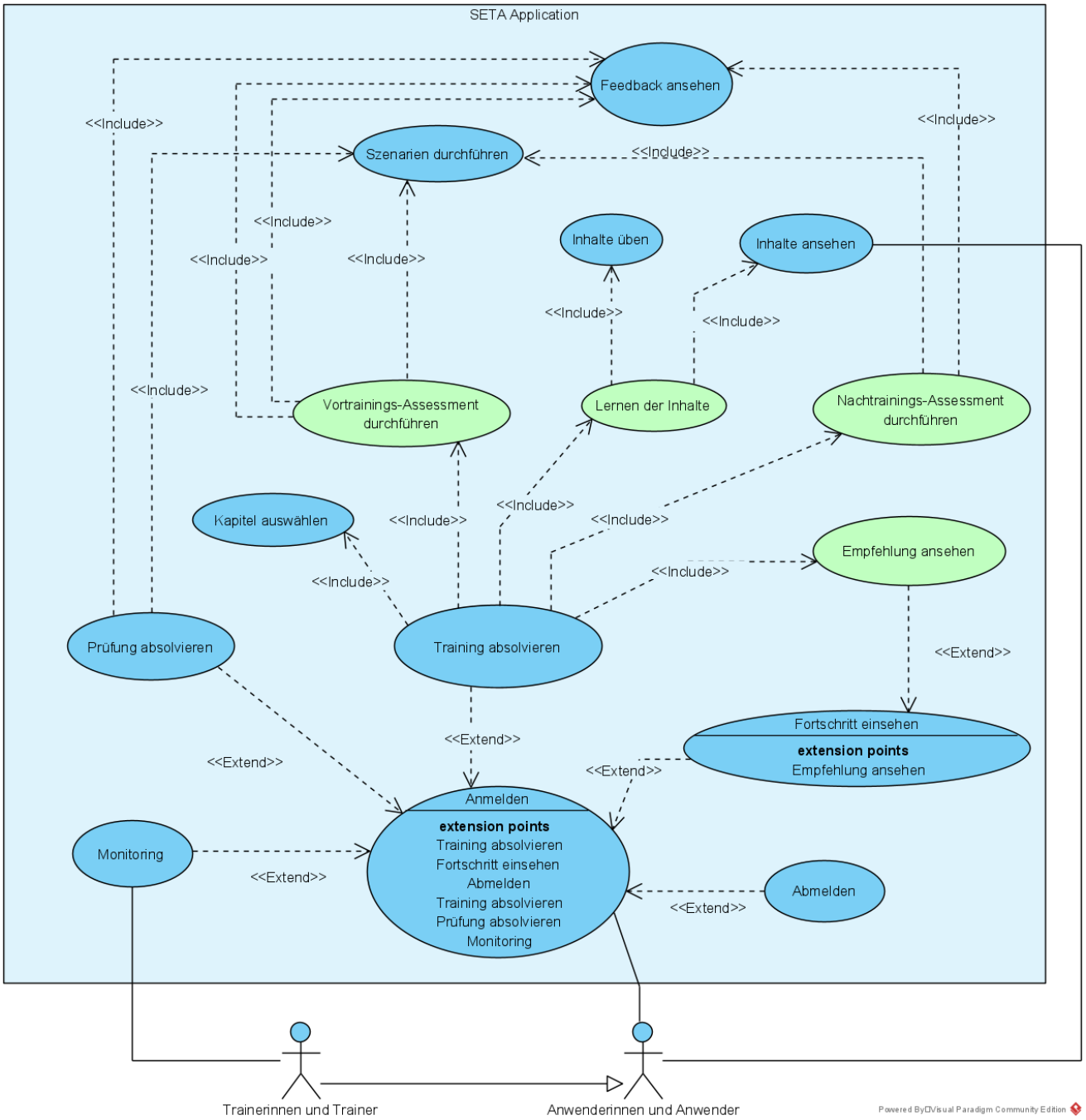
Beschreibung: Trainerinnen und Trainer können Resultate von Prüfungen der Angestellten einsehen.
Voraussetzungen:

1. Trainerinnen und Trainer haben sich mit ihrem Benutzerkonto angemeldet.

Funktionale Anforderungen:

- **FA701:** Trainerinnen und Trainer wird eine Übersicht über Angestellte und deren aktuellste Prüfungen angezeigt.
- **FA702:** Eine abgeschlossene Prüfung kann ausgewählt und dessen Ergebnis angezeigt werden.

9.3 Szenario View



Use Case:	Anmelden
Use Case ID:	USC001
Rolle:	Anwenderinnen und Anwender

Beschreibung:	Anwenderinnen oder Anwender können sich mit ihrem Benutzerkonto anmelden.
Voraussetzung:	Keine
Auslöser:	Anmeldebildschirm wird angezeigt
Ergebnis:	Der Anwender oder die Anwenderin ist angemeldet.
Erfolgsszenario:	1. Anwenderinnen oder Anwender geben ihren Benutzernamen in ein Formular ein. 2. Anwenderinnen oder Anwender geben ihr Passwort in ein Formular ein. 3. Bestätigen die Eingabe
Extensions:	Keine
Priorität:	Hoch
Risiken:	Transfer von Benutzernamen und Passwörtern

Use Case:	**Abmelden**
Use Case ID:	USC002
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen oder Anwender können sich Abmelden.
Voraussetzung:	USC001
Auslöser:	Schaltfläche zum Abmelden wird ausgewählt.
Ergebnis:	Anwenderinnen oder Anwender sind abgemeldet.
Erfolgsszenario:	Schaltfläche zum Abmelden wird ausgewählt.
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	**Kapitel auswählen**

Use Case ID:	USC003
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen oder Anwender können Kapitel auswählen.
Voraussetzung:	Keine
Auslöser:	Die Schaltfläche eines Kapitels wird angeklickt.
Ergebnis:	Das gewünschte Kapitel ist ausgewählt.
Erfolgsszenario:	Die Schaltfläche eines Kapitels wird angeklickt.
Extensions:	Keine
Priorität:	Mittel
Risiken:	Keine

Use Case:	**Inhalte ansehen**
Use Case ID:	USC004
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Nutzer können IT-sicherheitsspezifische Inhalte ansehen.
Voraussetzung:	USC003
Auslöser:	Ein Kapitel wurde ausgewählt.
Ergebnis:	IT-sicherheitsspezifische Inhalte werden angezeigt.
Erfolgsszenario:	Ein Kapitel wurde ausgewählt.
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	Training absolvieren
Use Case ID:	USC005
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen und Anwender können ein Kapitel auswählen, ein Training darin starten und absolvieren. Wenn kein Kapitel ausgewählt ist, kann es bei Trainingsstart ausgesucht werden.
Voraussetzung:	USC001 und optional USC003
Auslöser:	Ein Training wird gestartet.
Ergebnis:	Ein Training wurde durchgeführt.
Erfolgsszenario:	1. Ein Vortrainings-Assessment wird durchgeführt. USC006 2. Inhalte werden gelernt. USC007 3. Nachtrainings-Assessment wird durchgeführt. USC008 4. Empfehlung wird angesehen. USC009
Extensions:	USC003 wenn kein Kapitel ausgewählt wurde.
Priorität:	Hoch
Risiken:	Keine

Use Case:	Vortrainings-Assessment durchführen
Use Case ID:	USC006
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Im Rahmen eines Trainings wird ein Vortrainings-Assessment durchgeführt. Die Anwenderinnen und Anwender müssen dabei Szenarien beantworten und können sich anschließend ein Feedback darüber ansehen.
Voraussetzung:	USC005
Auslöser:	Training gestartet
Ergebnis:	Das Vortrainings- Assessment ist abgeschlossen.
Erfolgsszenario:	1. Alle Szenarien werden beantwortet. USC010 2. Feedback wird angesehen. USC011
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	Lernen der Inhalte
Use Case ID:	USC007
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen und Anwender können IT-Security relevante Inhalte aktiv lernen. Inhalte werden präsentiert und aktiv anhand von Szenarien geübt.
Voraussetzung:	USC006
Auslöser:	Vortrainings-Assessment abgeschlossen
Ergebnis:	Lernen der Inhalte abgeschlossen
Erfolgsszenario:	1. Anwenderinnen und Anwender lesen die angezeigten Inhalte 2. Anwenderinnen und Anwender gehen Übungsszenarien durch. USC012
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	Nachtrainings-Assessment durchführen
Use Case ID:	USC008
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Die Anwenderinnen und Anwender müssen Szenarien beantworten und können sich anschließend ein Feedback darüber ansehen.
Voraussetzung:	USC007
Auslöser:	Lernen der Inhalte abgeschlossen
Ergebnis:	Nachtrainings-Assessment ist abgeschlossen
Erfolgsszenario:	1. Alle Szenarien werden beantwortet. USC010 2. Feedback wird angesehen. USC011
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	Empfehlung ansehen
Use Case ID:	USC009
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen und Anwender können sich Empfehlungen ansehen. Ergebnisse des Vor und Nachtrainings-Assessments werden verglichen und die Entwicklung dargestellt. Empfehlungen bezüglich Verbesserungsmöglichkeiten werden angezeigt.
Voraussetzung:	USC008 oder USC015
Auslöser:	USC008 oder USC015
Ergebnis:	Empfehlung wird angezeigt
Erfolgsszenario:	Anwenderinnen und Anwender sehen sich die Empfehlungen des Trainings an.
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	Szenarien durchführen
Use Case ID:	USC010
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen und Anwender können Szenarien durchführen. Dabei wird das Szenario angezeigt und kann beantwortet werden.
Voraussetzung:	USC006 oder USC008 oder USC013
Auslöser:	Szenario wird gestartet
Ergebnis:	Das Szenario wurde beantwortet.
Erfolgsszenario:	1. Anwenderinnen und Anwender lesen die Aufgabenstellung 2. Anwenderinnen und Anwender wählen eine Antwort aus 3. Anwenderinnen und Anwender bestätigen ihre Auswahl.
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	Feedback ansehen
Use Case ID:	USC011
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen und Anwender können sich das Feedback zu Szenarien ansehen, die im Rahmen eines Vor-Nachtrainings-Assessment oder einer Kompetenzüberprüfung beantwortet wurden.
Voraussetzung:	USC006 oder USC008 oder USC013 oder USC015
Auslöser:	Beantwortung aller geplanten Szenarien.
Ergebnis:	Feedback zu beantworteten Szenarien wird angezeigt.
Erfolgsszenario:	Anwenderinnen und Anwender sehen die ihre und die korrekten antworten der Szenarien.
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	Inhalte üben
Use Case ID:	USC012
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Inhalte können anhand von Übungsszenarien geübt werden. Übungsszenarien werden nach Beantwortung richtig aufgelöst.
Voraussetzung:	USC007
Auslöser:	Inhalte werden im Rahmen eines Trainings angesehen und ein Übungsszenario gestartet
Ergebnis:	Übungsszenario wird beendet.
Erfolgsszenario:	1. Anwenderinnen und Anwender lesen die Aufgabenstellung. 2. Anwenderinnen und Anwender wählen eine Antwort aus. 3. Anwenderinnen und Anwender bestätigen ihre Antwort 4. Anwenderinnen und Anwender sehen die korrekte Antwort.
Extensions:	Keine
Priorität:	Hoch
Risiken:	Keine

Use Case:	Kompetenz abprüfen
Use Case ID:	USC013
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen und Anwender können mit einer Überprüfung ihre Kompetenzen evaluieren. Überprüfungen beinhalten Szenarien des ausgewählten Kapitels oder zufällige Szenarien aller Kapitel, wenn keines ausgewählt wurde. Nachdem alle Szenarien abgeschlossen wurden, wird ein Feedback angezeigt.
Voraussetzung:	USC001 oder USC003
Auslöser:	Überprüfung wird gestartet.
Ergebnis:	Feedback wird angezeigt.
Erfolgsszenario:	1. Szenarien werden beantwortet. USC010 2. Feedback wird angesehen. USC011
Extensions:	Keine
Priorität:	Mittel
Risiken:	Keine

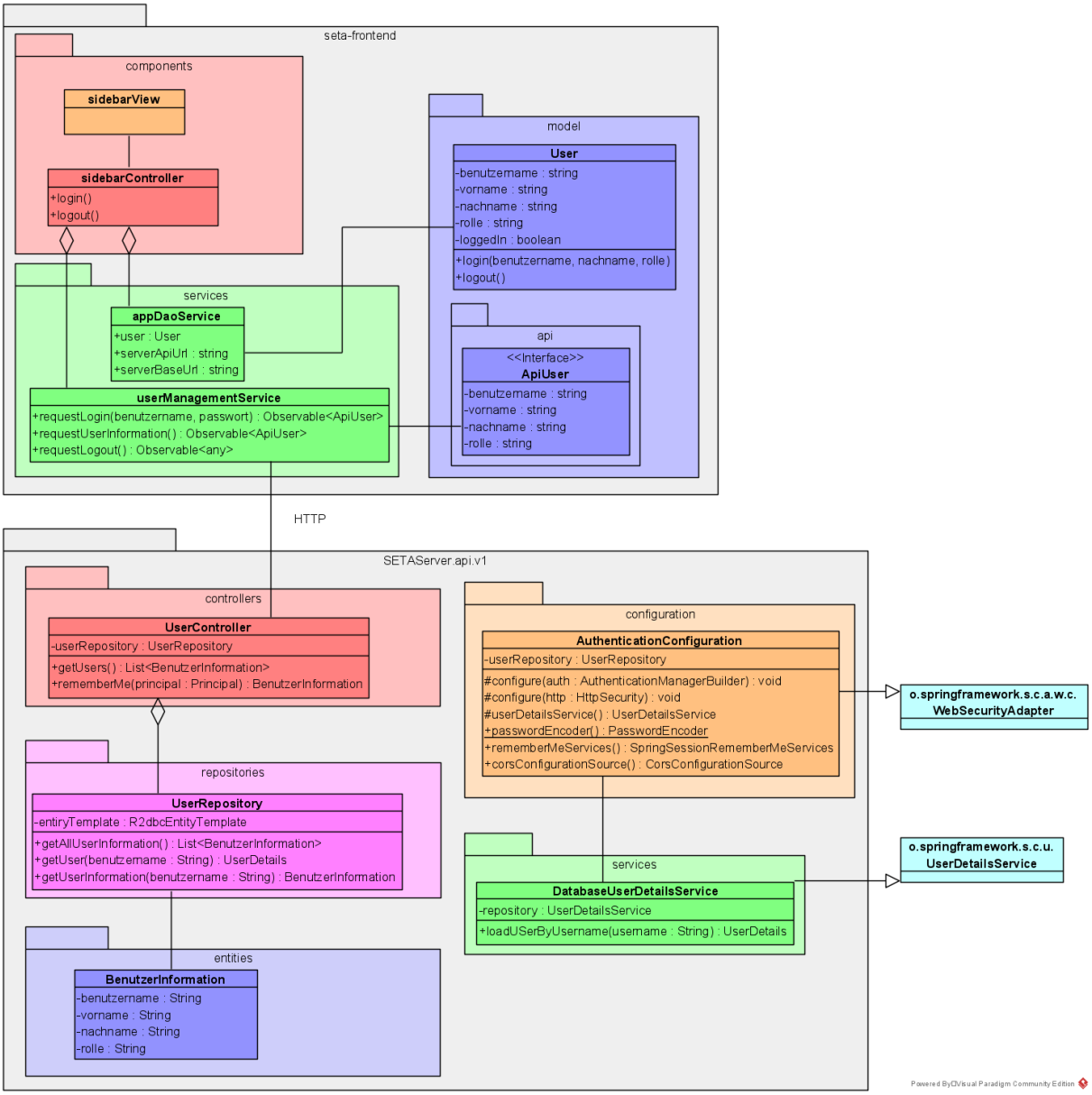
Use Case:	Erfolgsstatistik einsehen
Use Case ID:	USC014
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Erfolgsstatistiken abgeschlossener Trainings und Überprüfungen kann angezeigt werden
Voraussetzung:	USC001
Auslöser:	Erfolgsstatistik wird ausgewählt
Ergebnis:	Erfolgsstatistik wird angezeigt
Erfolgsszenario:	Erfolgsstatistik wird ausgewählt
Extensions:	USC015
Priorität:	Mittel
Risiken:	Keine

Use Case:	Ergebnis auswählen
Use Case ID:	USC015
Rolle:	Anwenderinnen und Anwender
Beschreibung:	Anwenderinnen und Anwender können in der Erfolgsstatistik das Ergebnis eines Trainings oder einer Überprüfung auswählen und anzeigen lassen.
Voraussetzung:	USC014, USC013, USC05
Auslöser:	Schaltfläche eines Trainings oder einer Überprüfung wird angeklickt
Ergebnis:	Empfehlung oder Feedback wird aufgerufen
Erfolgsszenario:	Schaltfläche eines Trainings oder einer Überprüfung wird angeklickt
Extensions:	Keine

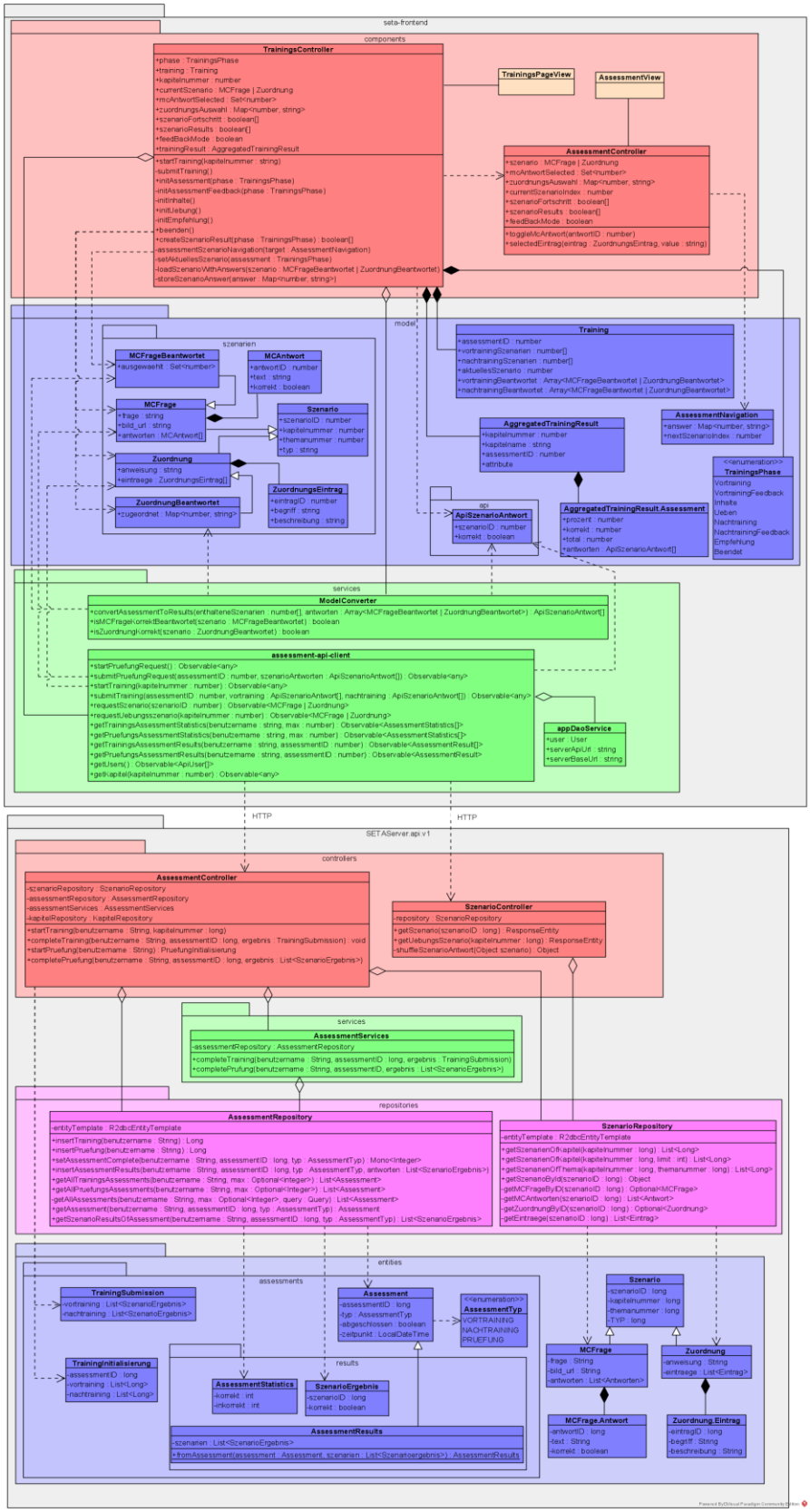
Priorität:	Mittel
Risiken:	Keine
Use Case:	**Monitoring**
Use Case ID:	USC016
Rolle:	Trainerinnen oder Trainer
Beschreibung:	Trainerinnen oder Trainer können aktuelle Ergebnisse von Überprüfungen der Angestellten einsehen.
Voraussetzung:	USC001
Auslöser:	Schaltfläche des Angestellten Monitoring ausgewählt.
Ergebnis:	Statistiken der Angestellten wird angezeigt.
Erfolgsszenario:	Schaltfläche des Angestellten Monitoring ausgewählt.
Extensions:	Keine
Priorität:	Mittel
Risiken:	Keine

9.4 Logical Views

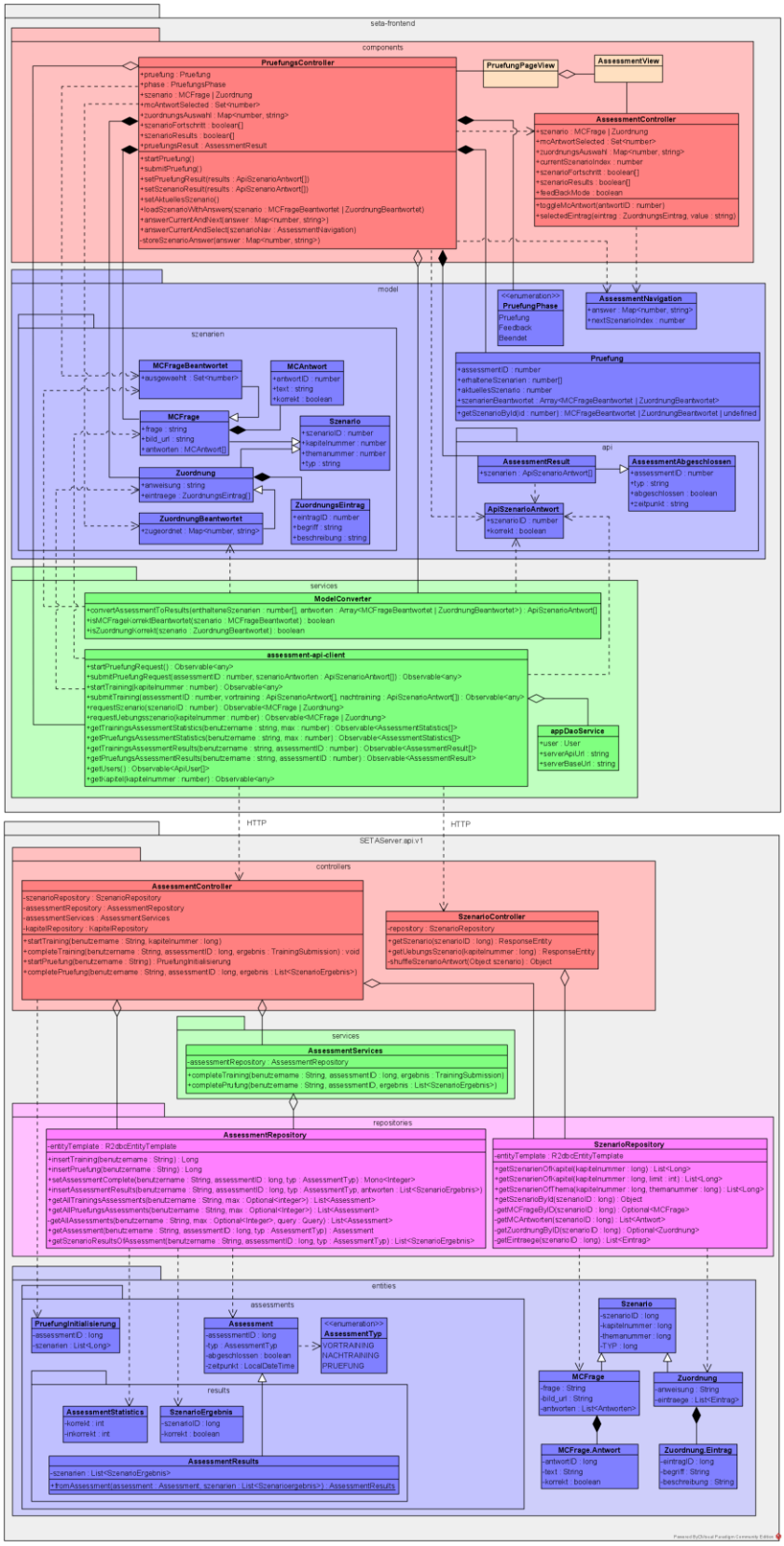
9.4.1 Logical View User Management



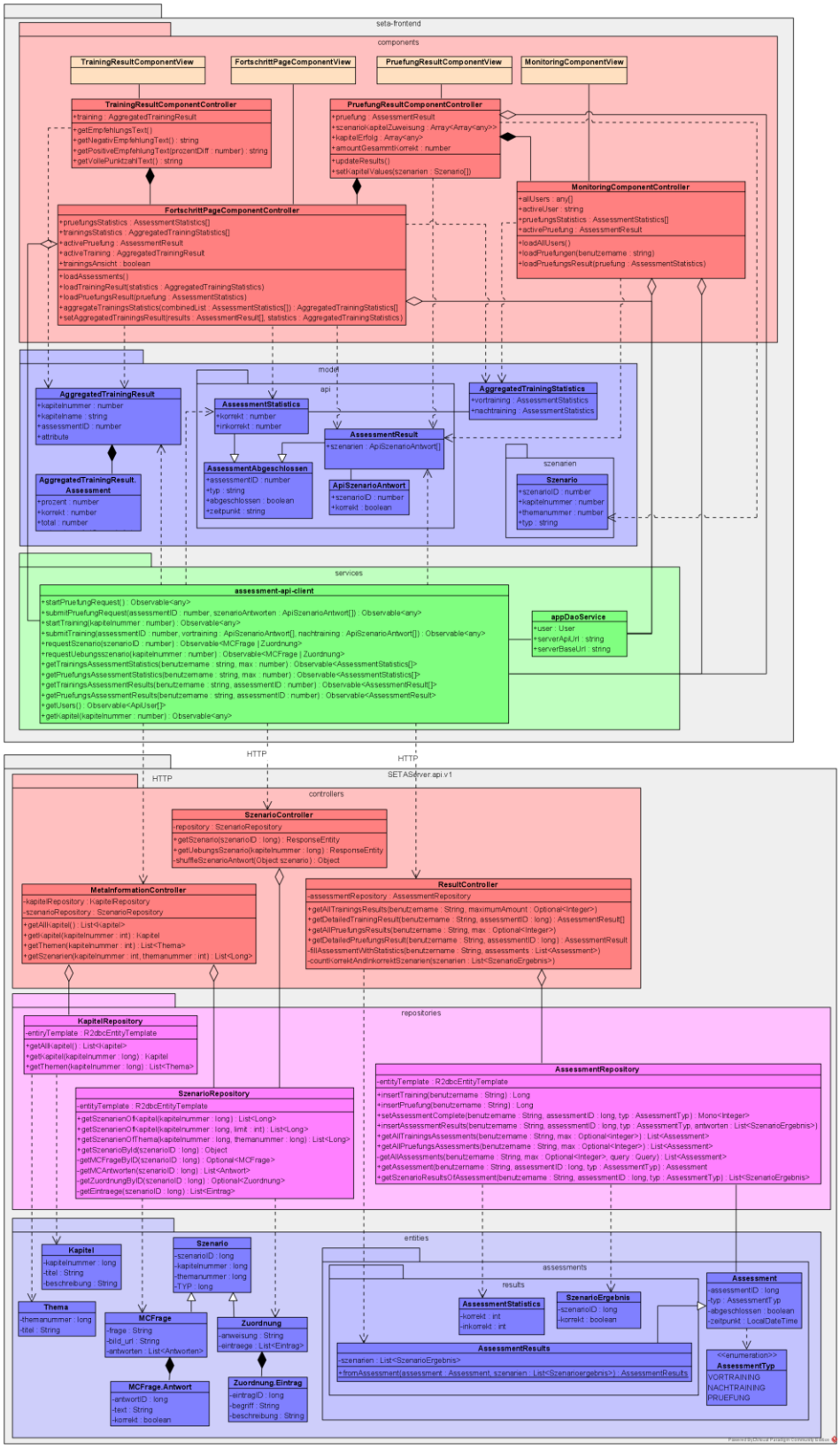
9.4.2 Logical View Training



9.4.3 Logical View Prüfung

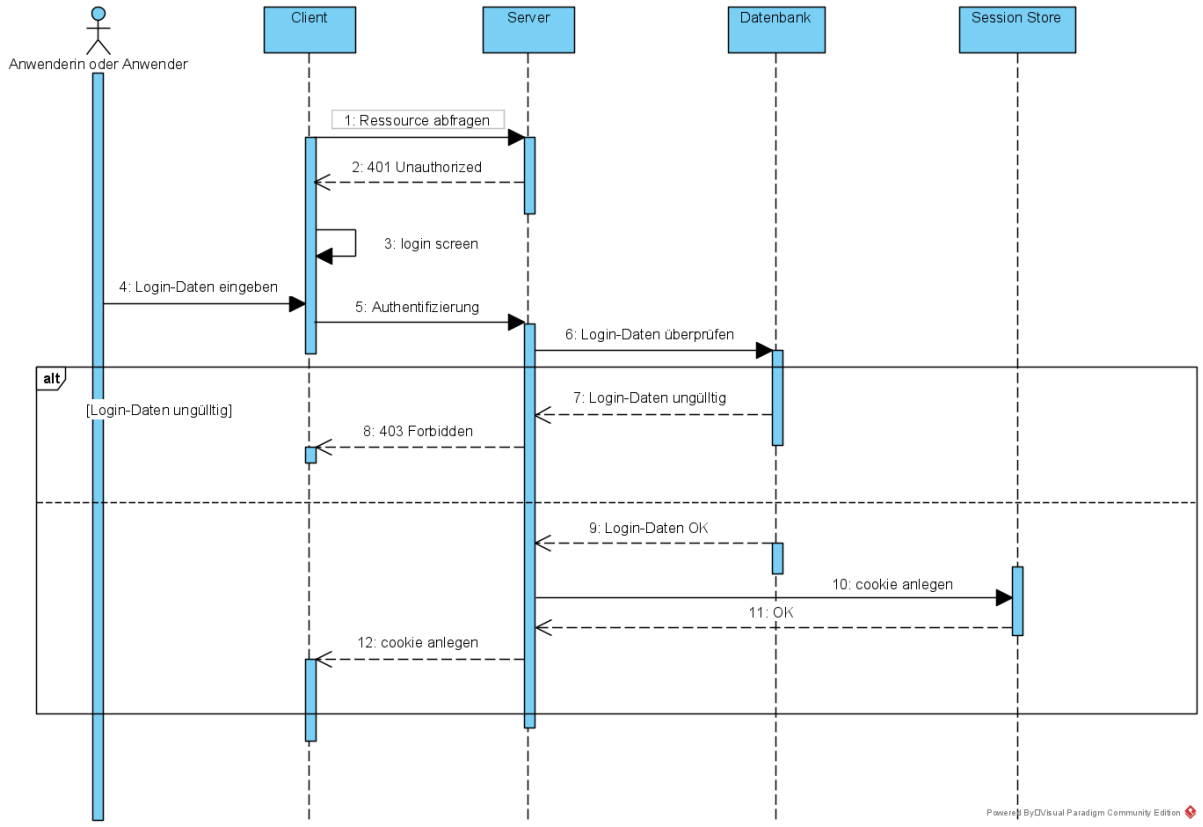


9.4.4 Logical View Prüfungs- und Trainingsergebnisse

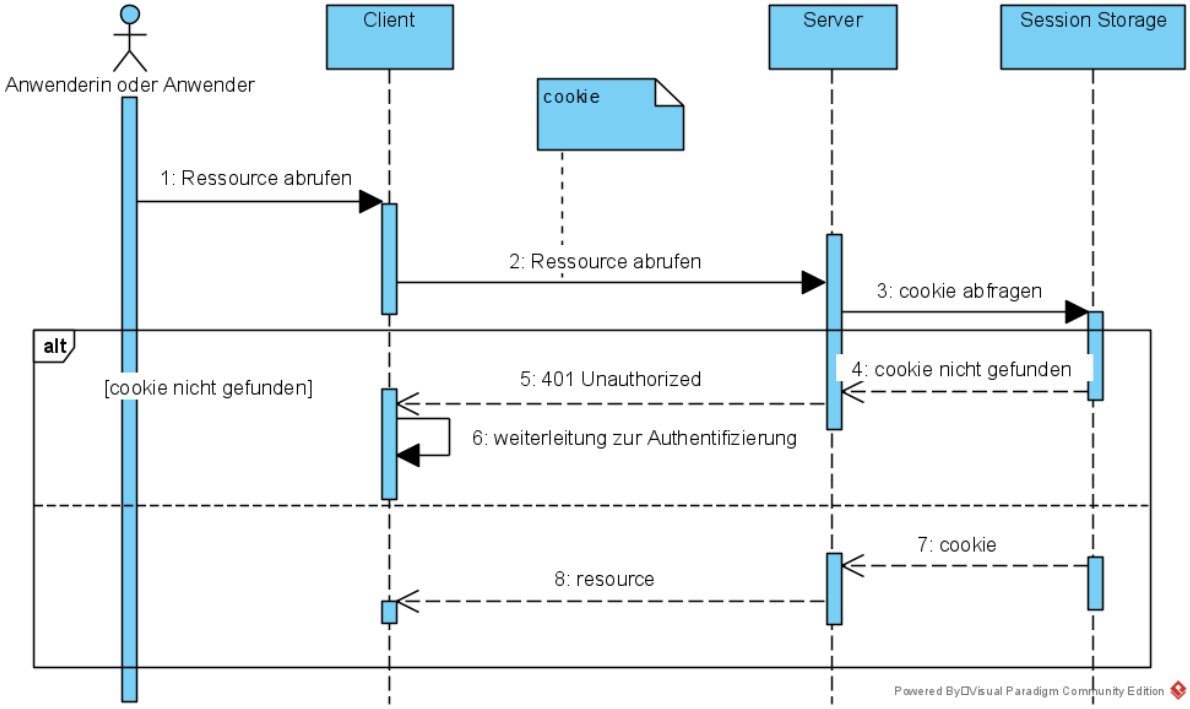


9.5 Process View

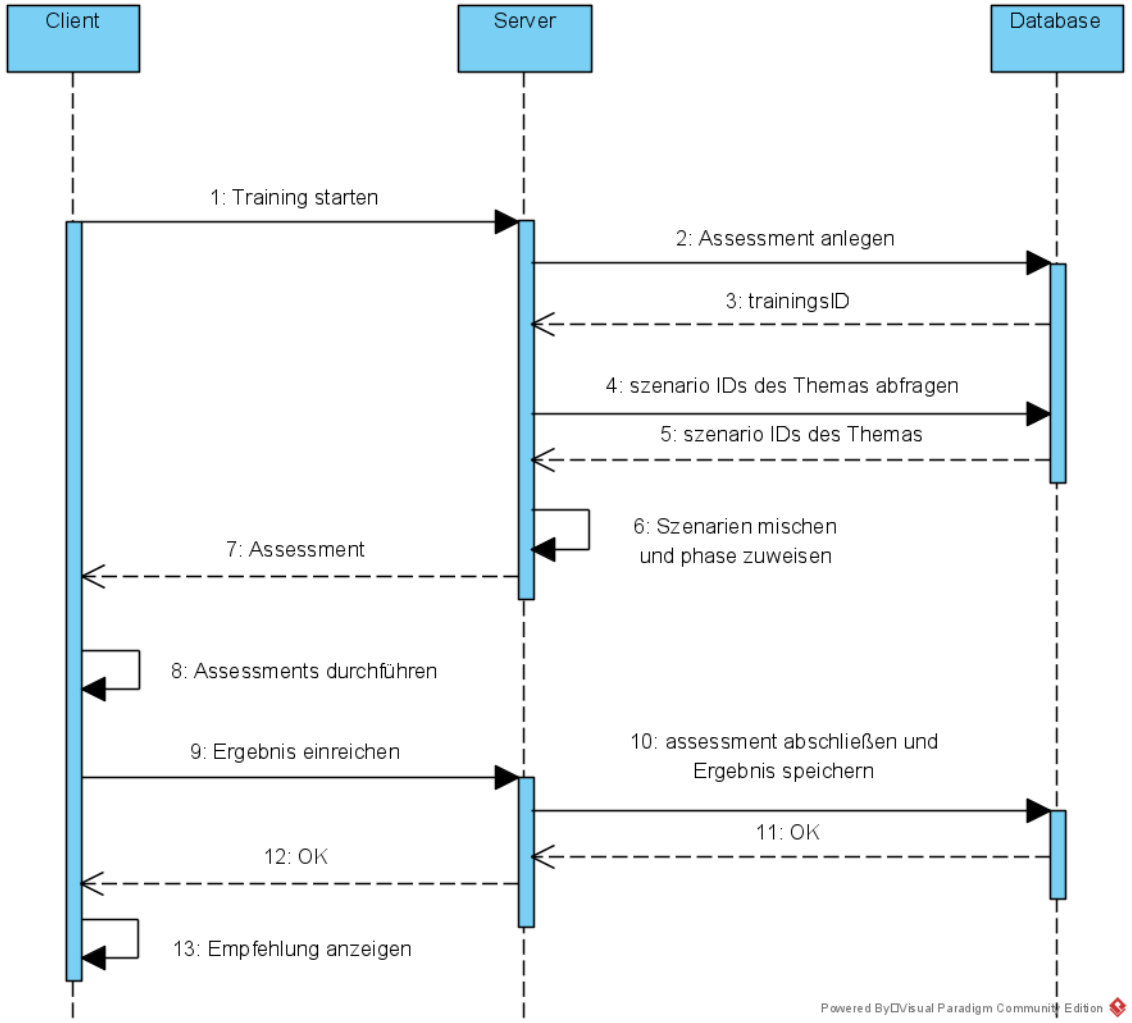
9.5.1 Process View Authentifizierung



9.5.2 Process View Autorisierung



9.5.3 Process View Training



9.6 Datenmodell

9.7 REST API

9.7.1 Meta Information

9.7.1.1 Alle Kapitel

endpoint: /api/v1/kapitel
method: GET
success response code: 200
response body:

```
[
    {
        "kapitelnummer": 1,
        "titel":"...",
```

```
        "beschreibung":"..."
    }, {
        "kapitelnummer": 2,
        "titel":"...",
        "beschreibung":"..."
    }, ...
]
```

9.7.1.2 Ein Kapitel

endpoint: `/api/v1/kapitel/{kapitelnummer}`
method: GET
success response code: 200
error response codes:

- 404 Not Found: kapitelnummer existiert nicht

response body:

```
{
    "kapitelnummer":1,
    "titel":"...",
    "beschreibung":"..."
}
```

9.7.1.3 Themen

endpoint: `/api/v1/kapitel/{kapitelnummer}/themen`
method: GET
success response code: 200
error response codes:

- 404 Not Found: kapitelnummer existiert nicht

response body:

```
[
    {
        "themanummer": 1,
        "titel":"..."
    }, {
        "themanummer": 2,
        "titel":"..."
    }, ...
]
```

9.7.1.4 Szenarien

endpoint: `/api/v1/kapitel/{kapitelnummer}/themen/{themanummer}/szenarien`
method: GET
success response code: 200
error response codes:

- 404 Not Found: kapitelnummer existiert nicht
- 404 Not Found: themanummer existiert nicht

response body:

```
[1, 54, 3, 23] //szenarioIDs
```

9.7.2 Szenarien

9.7.2.1 Szenario

endpoint: /api/v1/szenarien/{szenarioID}
method: GET
success response code: 200
error response codes:

- 404 Not Found: szenario existiert nicht

response body:

```
{
    "szenarioID":123,
    "kapitelnummer":1,
    "themanummer":2,
    "typ":"MCFrage",
    "frage":"...",
    "bild_url":"...", //optional, can be empty
    "antworten":[
        {
            "antwortID":1,
            "text":"...",
            "korrekt":false
        }, {
            "antwortID":2,
            "text":"...",
            "korrekt":true
        }, ...
    ]
}
```

oder

```
{
    "szenarioID":123,
    "kapitelnummer":1,
    "themanummer":2,
    "typ":"Zuordnung",
    "anweisung":"...",
    "eintraege":[
        {
            "eintragID":1,
            "begriff":"...",
            "beschreibung":"..."
        }, {
            "antwortID":2,
            "begriff":"...",
            "beschreibung":"..."
        }, ...
    ]
}
```

9.7.3 Authentification

9.7.3.1 Login

endpoint: /api/v1/login
method: POST
request header:
- Content-Type: application/x-www-form-urlencoded

request body:
benutzername={benutzername}&passwort={passwort}
success response code: 200
response header:
- Set-Cookie: access_token={access_token}; Expires={expiration date}; Secure; HttpOnly

error response code:
- 400: Bad request
- 403: Forbidden

9.7.3.2 Logout

endpoint: /api/v1/logout
method: POST
request header:
- Cookie: <cookie>

success response code: 200
error response code:
- 404: Benutzername nicht gefunden

9.7.3.3 Identification

endpoint: /api/v1/whoami
method: GET
request header:
- Cookie: <cookie>

success response code: 200
error response code:
- 400: Bad request
- 403: Forbidden
- 404: Not Found

response body:

```
{
    "benutzername":"maxmuster",
    "vorname":"max",
    "nachname":"muster",
    "rolle": "ANWENDER"
}
```

9.7.4 Assessments

9.7.4.1 Neues Training starten

endpoint: /api/v1/users/{benutzername}/assessments/trainings/kapitel/{kapitenummer}
method: POST
request header:

- Cookie: <cookie>

success response code: 201
error response codes:

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found : benutzername existiert nicht
- 404 Not Found : kapitelnummer existiert nicht

response body:

```
{
    "assessmentID":123,
    "vortraining":[4,9,2,34,2,3,5],
    "nachtraining":[6,23,53,15,43,54,64]
}
```

9.7.4.2 Training abschließen

endpoint: /api/v1/users/{benutzername}/assessments/trainings/{assessmentID}
method: PUT
request header:

- Cookie: <cookie>

request body:

```
{
    "vortraining": [
        {
            "szenarioID":12,
            "korrekt":true
        }, {
            "szenarioID":13,
            "korrekt":true
        }
        }, {
            "szenarioID":4,
            "korrekt":false
        }, ...
    ],
    "nachtraining": [
        {
            "szenarioID":22,
            "korrekt":true
        }, {
            "szenarioID":23,
            "korrekt":true
        }
        }, {
```

```
            "szenarioID":34,
            "korrekt":false
        }, ...
    ]
}
```

success response code: 200
error response codes:

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found : benutzername existiert nicht

9.7.4.3 Neue Prüfung starten

endpoint: /api/v1/users/{benutzername}/assessments/pruefungen
method: POST
request header:

- Cookie: <cookie>

success response code: 201
error response codes:

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found : benutzername existiert nicht

response body:

```
{
    "assessmentID":567,
    "szenarien":[4,9,2,34,2,3,5,6,23,53,15,43,54,64]
}
```

9.7.4.4 Prüfung abschließen

endpoint: /api/v1/users/{benutzername}/assessments/pruefungen/{assessmentID}
method: PUT
request header:

- Cookie: <cookie>

request body:

```
[
    {
        "szenarioID":12,
        "korrekt":true
    }, {
        "szenarioID":13,
        "korrekt":true
    }, {
        "szenarioID":4,
        "korrekt":false
    }, ...
]
```

success response code: 200
error response codes:

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found : benutzername existiert nicht
- 404 Not Found : AssessmentID existiert nicht

9.7.5 Assessments Abrufen (Ergebnisse)

9.7.5.1 Prüfungen Abrufen

endpoint: `/api/v1/users/{benutzername}/assessments/pruefungen`
method: GET
request param: ?max={max. anzahl}
request header:

- Cookie: <cookie>

success response code: 200
error response codes:

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found : benutzername existiert nicht

response body:

```
[
    {
        "assessmentID":567,
        "typ": "PRUEFUNG",
        "abgeschlossen": true,
        "zeitpunkt": YYYY-MM-DD HH:MM:SS,
        "korrekt":12,
        "inkorrekt":8
    }, {
        "assessmentID":568,
        "typ": "PRUEFUNG",
        "abgeschlossen": false,
        "zeitpunkt": YYYY-MM-DD HH:MM:SS,
        "korrekt":0,
        "inkorrekt":0
    }, ...
]
```

9.7.5.2 Prüfungsergebnis Abrufen

endpoint: /api/v1/users/{benutzername}/assessments/pruefungen/{assessmentID}
method: GET
request header:

- Cookie: <cookie>

success response code: 200
error response codes:

- 401 Unauthorized
- 403 Forbidden
- 404 Not Found : benutzername existiert nicht
- 404 Not Found : assessmentID existiert nicht
- 404 Not Found : assessment nicht abgeschlossen

response body:

```
{
    "assessmentID":567,
    "typ": "PRUEFUNG",
    "abgeschlossen": true,
    "zeitpunkt": YYYY-MM-DD HH:MM:SS,
    "szenarien": [
        {
            "szenarioID":12,
            "korrekt":true
        }, ...
    ]
}
```

9.7.5.3 Trainings abrufen

endpoint: /api/v1/users/{benutzername}/assessments/trainings
method: GET
request param: ?max={max. anzahl}
request header:

- Cookie: <cookie>

success response code: 200
error response codes:

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found : benutzername existiert nicht
- 404 Not Found : assessment nicht abgeschlossen

response body:

```
[
    {
        "assessmentID":123,
        "typ": "VORTRAINING",
        "abgeschlossen":true,
        "zeitpunkt": YYYY-MM-DDTHH:MM:SS,
        "korrekt":12,
        "inkorrekt":8
    }, {
```

```
        "assessmentID":123,
        "typ": "NACHTRAINING",
        "abgeschlossen":true,
        "zeitpunkt": YYYY-MM-DDTHH:MM:SS,
        "korrekt":16,
        "inkorrekt":4
    }, ...
]
```

9.7.5.4 Trainingsergebnis Abrufen

endpoint: /api/v1/users/{benutzername}/assessments/trainings/{assessmentID}
method: GET
request header:

- Cookie: <cookie>

success response code: 200
error response codes:

- 401 Unauthorized
- 403 Forbidden
- 404 Not Found : benutzername existiert nicht
- 404 Not Found : assessment existiert nicht
- 404 Not Found : assessment nicht abgeschlossen

response body:

```
[
    {
        "assessmentID":123,
        "typ": "VORTRAINING",
        "abgeschlossen": true,
        "zeitpunkt": YYYY-MM-DDTHH:MM:SS,
        "szenarien": [
            {
                "szenarioID":12,
                "korrekt":true
            }, ...
        ]
    }, {
        "assessmentID":123,
        "typ": "NACHTRAINING",
        "abgeschlossen": true,
        "zeitpunkt": YYYY-MM-DDTHH:MM:SS,
        "szenarien": [
            {
                "szenarioID":12,
                "korrekt":true
            }, ...
        ]
    }
]
```

9.7.6 Trainer Funktionen

9.7.6.1 User abfragen

endpoint: /api/v1/users
method: GET
request header:

- Cookie: <cookie>

success response code: 200
error response codes:

- 401 Unauthorized
- 403 Forbidden

response body:

```
[
    {
        "benutzername":"maxmuster1",
        "vorname":"Max",
        "nachname":"Muster"
    }, {
        "benutzername":"franziskafischer2",
        "vorname":"Franziska",
        "nachname":"Fischer"
    }, ...
]
```

9.8 Implementierung und Code-Basis

9.8.1 docker-compose.yml

```
version: "3.9"
services:
    seta_server:
        build: SETAServer
        image: seta_server:seta
        container_name: seta_server
        depends_on:
            - seta_database
            - seta_session_store
        networks:
            app_net:
        ports:
            - "443:443"

    seta_database:
        build: Database
        image: seta_database:seta
        container_name: seta_database
        networks:
            app_net:
        ports:
            - "3306"

    seta_database_client:
        image: mariadb:10.4
        networks:
            app_net:
```

```
        depends_on:
            - seta_database
        command: mysql -useta_admin -pedcc67bbd164aa7ee858794cccaf4e45 --port=3306 -
hSETA_Database -w seta

    seta_session_store:
        build: SessionStore
        image: seta_session_store:seta
        container_name: seta_session_store
        networks:
            app_net:
        ports:
            - "6379"

networks:
    app_net:
```

9.8.2 Dockerfiles

9.8.2.1 Dockerfile Server

```
FROM openjdk:11
COPY ./SETAServer /usr/src/SETAServer
COPY ./szenario_resources/images  /usr/src/SETAServer/src/main/resources/static/images
COPY ./seta-frontend/dist/seta-frontend  /usr/src/SETAServer/src/main/resources/static
EXPOSE 443

WORKDIR /usr/src/SETAServer
RUN chmod +x gradlew
CMD ["./gradlew", "build", "bootrun"]
```

9.8.2.2 Dockerfile Datenbank

```
FROM mariadb:10.4
ENV MYSQL_ROOT_PASSWORD=2474e6fa4caf89a76541575ad15de30b
ENV MYSQL_DATABASE=seta
ENV MYSQL_USER=seta_admin
ENV MYSQL_PASSWORD=edcc67bbd164aa7ee858794cccaf4e45

COPY config.cnf /etc/mysql/conf.d/config.cnf
RUN chmod 0444 /etc/mysql/conf.d/config.cnf

COPY setup /docker-entrypoint-initdb.d/
COPY szenarien /docker-entrypoint-initdb.d/

EXPOSE 3306
```

9.8.2.3 Dockerfile Session Storage

```
FROM redis
COPY redis.conf /usr/local/etc/redis/redis.conf
CMD [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
```

9.9 Datenbank

9.9.1 Tabellen Statements (Database\setup\001_setup_tables.sql)

```
USE seta;
CREATE TABLE Kapitel (
    kapitelnummer INT UNSIGNED,
    titel VARCHAR(255) NOT NULL,
    beschreibung VARCHAR(2000),
    CONSTRAINT PRIMARY KEY (kapitelnummer)
);

CREATE TABLE Thema (
    kapitelnummer INT UNSIGNED,
    themanummer INT UNSIGNED,
    titel VARCHAR(255) NOT NULL,
    CONSTRAINT PRIMARY KEY (kapitelnummer, themanummer),
    CONSTRAINT fkey_kapitelnummer FOREIGN KEY (kapitelnummer) REFERENCES Kapitel (kapitelnu
mmer)
);

CREATE TABLE Szenario (
    szenarioID INT UNSIGNED AUTO_INCREMENT,
    kapitelnummer INT UNSIGNED,
    themanummer INT UNSIGNED,
    CONSTRAINT PRIMARY KEY (szenarioID),
    CONSTRAINT fkey_szenario FOREIGN KEY (kapitelnummer, themanummer) REFERENCES Thema (kap
itelnummer, themanummer)
);

CREATE TABLE MC_Frage (
    szenarioID INT UNSIGNED,
    frage VARCHAR(1000) NOT NULL,
    bild_url VARCHAR(255),
    CONSTRAINT PRIMARY KEY (szenarioID),
    CONSTRAINT fkey_mcfrage FOREIGN KEY (szenarioID) REFERENCES Szenario (szenarioID)
);

CREATE TABLE MC_Antwort (
    antwortID INT UNSIGNED AUTO_INCREMENT,
    szenarioID INT UNSIGNED,
    text VARCHAR(1000) NOT NULL,
    korrekt BOOLEAN DEFAULT (False),
    CONSTRAINT PRIMARY KEY (antwortID, szenarioID),
    CONSTRAINT fkey_mcantwort FOREIGN KEY (szenarioID) REFERENCES MC_Frage (szenarioID)
);

CREATE TABLE Zuordnung (
    szenarioID INT UNSIGNED,
    anweisung VARCHAR(1000) NOT NULL,
    CONSTRAINT PRIMARY KEY (szenarioID),
    CONSTRAINT fkey_zuordnung FOREIGN KEY (szenarioID) REFERENCES Szenario (szenarioID)
);

CREATE TABLE Eintrag (
    eintragID INT UNSIGNED AUTO_INCREMENT,
    szenarioID INT UNSIGNED,
    begriff VARCHAR(255) NOT NULL,
    beschreibung VARCHAR(1000) NOT NULL,
    CONSTRAINT PRIMARY KEY (eintragID, szenarioID),
    CONSTRAINT fkey_eintrag FOREIGN KEY (szenarioID) REFERENCES Zuordnung (szenarioID)
```

```
);

CREATE TABLE User (
    benutzername VARCHAR(255),
    vorname VARCHAR(255) NOT NULL,
    nachname VARCHAR(255) NOT NULL,
    passwort VARCHAR(64) NOT NULL,
    rolle ENUM ('ANWENDER', 'TRAINER') NOT NULL,
    CONSTRAINT PRIMARY KEY (benutzername)
);

CREATE TABLE Assessment (
    assessmentID INT UNSIGNED AUTO_INCREMENT,
    typ ENUM('VORTRAINING', 'NACHTRAINING', 'PRUEFUNG'),
    benutzername VARCHAR(255),
    zeitpunkt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    abgeschlossen BOOLEAN NOT NULL DEFAULT (False),
    CONSTRAINT PRIMARY KEY (assessmentID, typ, benutzername),
    CONSTRAINT fkey_assessment FOREIGN KEY (benutzername) REFERENCES User (benutzername)
);

CREATE TABLE benutzer_beantwortet_szenario (
    szenarioID INT UNSIGNED,
    assessmentID INT UNSIGNED,
    typ ENUM('VORTRAINING', 'NACHTRAINING', 'PRUEFUNG'),
    benutzername VARCHAR(255),
    korrekt BOOLEAN NOT NULL,
    CONSTRAINT PRIMARY KEY (szenarioID, benutzername, assessmentID, typ),
    CONSTRAINT fkey_beantwortet_szenario FOREIGN KEY (szenarioID) REFERENCES Szenario (szen
arioID),
    CONSTRAINT fkey_beantwortet_assessment FOREIGN KEY (benutzername, assessmentID, typ) RE
FERENCES Assessment (benutzername, assessmentID, typ)
);
```

9.9.2 Nutzer Statements (Database\setup\001_setup_tables.sql)

```
CREATE USER 'server_access'@'%' IDENTIFIED BY '2dfb5c80912afcd5b561ae9b2c1a9d45';
GRANT SELECT, INSERT, DELETE, UPDATE ON seta.* TO 'server_access'@'%';

FLUSH PRIVILEGES;
```

9.9.3 View Statements (Database\setup\001_setup_tables.sql)

```
USE seta;
CREATE VIEW mc_szenario_view AS
SELECT sz.szenarioID, sz.kapitelnummer, sz.themanummer, mc.frage, mc.bild_url
FROM Szenario sz
JOIN MC_Frage mc
ON (sz.szenarioID=mc.szenarioID);

CREATE VIEW zuordnung_szenario_view AS
SELECT sz.szenarioID, sz.kapitelnummer, sz.themanummer, zu.anweisung
FROM Szenario sz
JOIN Zuordnung zu
ON (sz.szenarioID=zu.szenarioID);
```

9.9.4 Kapitel Statements (Database\setup\001_setup_tables.sql)

```
USE seta;
INSERT Kapitel (kapitelnummer, titel, beschreibung)
VALUES (1, 'Sicherer Umgang mit Computern und Informationen',
'Informationen sind das Kapital jedes Unternehmens. Alle Mitarbeiterinnen und Mitarbeiter m
üssen
über den Wert der Information Bescheid wissen. Der Missbrauch von Daten kann Wettbewerbsnac
hteile,
Umsatzeinbußen, Imageverluste oder rechtliche Probleme zur Folge haben. Daher ist es besond
ers
wichtig, Informationen und Computer vor unberechtigter Verwendung zu schützen.'),

(2, 'Passwörter – richtig auswählen und verwalten',
'Passwörter dienen dem Schutz von IT-
Geräten und Daten und verhindern unbefugte Zugriffe. Deswegen
sind die richtige Auswahl und der richtige Umgang wichtig: Passwörter müssen komplex sein
und geheimgehalten werden.'),

(3, 'Sicher unterwegs im Internet',
'Ebenso wie im realen Leben ist man auch im Internet mit Kriminellen und Betrügern konfront
iert.
Es liegt in Ihrer eigenen Verantwortung, solche Bedrohungen zu erkennen und entsprechend da
rauf
zu reagieren.'),

(4, 'E-Mails und Spam',
'Sicherer Umgang mit unerwünschten Mails.'),

(5, 'Gefährliche Schadprogramme',
'Schadprogramme wie z.B. Computer-
Viren enthalten verdeckte Funktionen, die durch Löschen, Überschreiben
oder sonstige Veränderungen Schäden an Betriebssystemen, Anwendungsprogrammen und
Daten erzeugen. Sie verursachen damit zusätzliche Arbeit und Kosten und haben einen negativ
en
Einfluss auf die Vertraulichkeit, Integrität und Verfügbarkeit von Daten oder Programmen.')
;
```

9.9.5 Themen Statements (Database\setup\001_setup_tables.sql)

```
USE seta;
INSERT Thema (kapitelnummer, themanummer, titel) VALUES
(1, 1, 'Sicherer Umgang mit personenbezogenen Daten'),
(1, 2, 'Clear Desk Policy'),
(1, 3, 'Datenträger und Papierdokumente richtig entsorgen'),
(1, 4, 'Sicherer Umgang mit mobilen IT-Geräten'),
(1, 5, 'Wechselmedien richtig verwenden'),
(1, 6, 'Social Engineering'),
(2, 1, 'Die richtige Auswahl'),
(2, 2, 'Der richtige Umgang'),
(2, 3, 'Passwort-Manager verwenden'),
(3, 1, 'Vorsichtsmaßnahmen'),
(3, 2, 'Verschlüsselte Datenübertragung'),
(3, 3, 'Tracking Cookies'),
(4, 1, 'Umgang mit unerwünschten Mails'),
(4, 2, 'Phishing-Mails'),
(4, 3, 'Gefälschte Absenderadressen'),
(4, 4, 'Sparsamer Einsatz der eigenen Mail-Adresse im Internet'),
(5, 1, 'Wie können Sie erkennen, dass Ihr PC infiziert ist?'),
(5, 2, 'Ransomware und Verschlüsselungstrojaner'),
```

```
(5, 3, 'Maßnahmen richtig setzen'),
(5, 4, 'Vireninfektion: Was tun?');
```

9.10 Server

9.10.1 application.properties

```
# HTTPS configuration
server.port=443
server.ssl.key-store-type=PKCS12
# The path to the keystore containing the certificate
server.ssl.key-store=classpath:keystore/seta_server.p12
server.ssl.key-store-password=ZITZqdYi7nyYbYjyA43v
# The alias mapped to the certificate
#server.ssl.key-alias=SETA Server
server.ssl.enabled=true

# database configuration
mariadb.url=seta_database
mariadb.user=server_access
mariadb.password=2dfb5c80912afcd5b561ae9b2c1a9d45

spring.jpa.hibernate.naming.implicit-
strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImpl
spring.jpa.hibernate.naming.physical-
strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

# session store configuration redis
seta.redis.hostname=seta_session_store
seta.redis.username=server_access
seta.redis.password=0be2dc6243f235f60a6300feab41bad8c2fdf0bed4982894e34a8b31f26ab6d
d

# application Configuration
training.szenarien_per_assessment=8
pruefung.szenarien_per_kapitel=8

logging.level.root=info
```

9.10.2 AuthenticationConfiguration.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.configuration;

import at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.UserRepository;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.services.DatabaseUserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.security.SecurityProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.http.HttpStatus;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.Authenticati
onManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
```

```
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigu
rerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.web.authentication.logout.HttpStatusReturningLogoutSuc
cessHandler;
import
org.springframework.session.security.web.authentication.SpringSessionRememberMeServ
ices;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import java.util.Arrays;

/**
 * Configuration class for user authentication using spring security
 */
@Configuration
@EnableWebSecurity
@Order(SecurityProperties.BASIC_AUTH_ORDER - 10)
public class AuthenticationConfiguration extends WebSecurityConfigurerAdapter {
    @Autowired
    private UserRepository userRepository;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        DaoAuthenticationProvider authenticationProvider = new
DaoAuthenticationProvider();
        authenticationProvider.setUserDetailsService(userDetailsService());
        authenticationProvider.setPasswordEncoder(passwordEncoder());
        auth.authenticationProvider(authenticationProvider);
    }

    /**
     * Configures accessible paths, their required roles, urls of the login/logout
page and handler
     */
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .csrf().disable()
                .authorizeRequests(authorize -> authorize
                        .mvcMatchers("/api/v1/users/{benutzername}/**")
                        .access("hasRole('TRAINER') or isAuthenticated() and
principal.username == #benutzername")
                        .mvcMatchers("/api/v1/users")
                        .hasRole("TRAINER")
                        .mvcMatchers("/api/v1/kapitel/**"
                                , "/api/v1/szenarien/**"
                                , "/api/v1/login*"
                                , "/api/v1/logout")
                        .permitAll())
                .formLogin(login -> login
                        .loginPage("/login.html")
                        .loginProcessingUrl("/api/v1/login")
                        .defaultSuccessUrl("/api/v1/whoami", true)
                        .failureUrl("/api/v1/whoami")
                        .usernameParameter("benutzername")
                        .passwordParameter("passwort"))
                .logout(logout -> logout
```

```
                        .logoutUrl("/api/v1/logout")
                        .logoutSuccessHandler(new
HttpStatusReturningLogoutSuccessHandler(HttpStatus.OK))
                        .invalidateHttpSession(true)
                        .deleteCookies("SESSION"))
                .cors(httpSecurityCorsConfigurer ->
httpSecurityCorsConfigurer.configurationSource(corsConfigurationSource()))
                .rememberMe((rememberMe) ->
rememberMe.rememberMeServices(rememberMeServices()));
    }

    @Override
    protected UserDetailsService userDetailsService() {
        return new DatabaseUserDetailsService(userRepository);
    }

    @Bean
    public static PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }


    /**
     * Enables spring session and session token functionality
     */
    @Bean
    public SpringSessionRememberMeServices rememberMeServices() {
        SpringSessionRememberMeServices rememberMeServices =
                new SpringSessionRememberMeServices();
        rememberMeServices.setAlwaysRemember(true);
        return rememberMeServices;
    }

    /**
     * Cors configuration for development
     */
    public CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOriginPatterns(Arrays.asList("*localhost*",
"*://192.168.1.*"));
        configuration.setAllowCredentials(true);
        configuration.setAllowedMethods(Arrays.asList("GET", "POST", "OPTIONS",
"PUT", "DELETE"));
        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);
        return source;
    }
}
```

9.10.3 Controller

9.10.3.1 AssessmentController.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.controllers;

import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.Kapitel;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.PruefungInitiali
sierung;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.TrainingInitiali
```

```
sierung;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.TrainingSubmissi
on;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results.Szenario
Ergebnis;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.AssessmentRepository;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.KapitelRepository;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.SzenarioRepository;
import at.ac.univie.unet.a01526005.SETAServer.api.v1.services.AssessmentServices;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Controller class with endpoints regarding assessments
 */
@RestController
@RequestMapping("/api/v1")
public class AssessmentController {
    /**
     * amount of szenarien per assessment in a training
     */
    @Value("${training.szenarien_per_assessment}")
    private int szenarienPerAssessment;

    /**
     * amount of szenarien for each kapitel in one pruefung
     */
    @Value("${pruefung.szenarien_per_kapitel}")
    private int szenarienPerKapitel;

    private SzenarioRepository szenarioRepository;

    private AssessmentRepository assessmentRepository;

    private AssessmentServices assessmentServices;

    private KapitelRepository kapitelRepository;


    @Autowired
    public AssessmentController(SzenarioRepository szenarioRepository,
                                AssessmentRepository assessmentRepository,
                                AssessmentServices assessmentServices,
                                KapitelRepository kapitelRepository) {
        this.szenarioRepository = szenarioRepository;
        this.assessmentRepository = assessmentRepository;
        this.assessmentServices = assessmentServices;
        this.kapitelRepository = kapitelRepository;
    }

    /**
     * Endpoint to create and store a new training for the given user and kapitel
```

```
     *
     * @return a newly created training
     */
    @ResponseStatus(HttpStatus.CREATED)

@PostMapping("/users/{benutzername}/assessments/trainings/kapitel/{kapitelnummer}")
    public TrainingInitialisierung startTraining(@PathVariable("benutzername")
String benutzername
            , @PathVariable("kapitelnummer") long kapitelnummer) {

        //get collection of szenarien for training
        List<Long> szenarien =
szenarioRepository.getSzenarienOfKapitel(kapitelnummer, szenarienPerAssessment *
2);
        Collections.shuffle(szenarien);

        // create and store a new training
        long assessmentID = assessmentRepository.insertTraining(benutzername);

        return new TrainingInitialisierung(assessmentID
                , szenarien.subList(0, szenarien.size() / 2)
                , szenarien.subList(szenarien.size() / 2, szenarien.size()));
    }

    /**
     * Endpoint to complete a previously started training with the given result
     */
    @PutMapping("/users/{benutzername}/assessments/trainings/{assessmentID}")
    public void completeTraining(@PathVariable("benutzername") String benutzername
            , @PathVariable("assessmentID") long assessmentID
            , @RequestBody TrainingSubmission ergebnis) {
        assessmentServices.completeTraining(benutzername, assessmentID, ergebnis);
    }

    /**
     * Endpoint to create a and store a new pruefung.
     */
    @ResponseStatus(HttpStatus.CREATED)
    @PostMapping("/users/{benutzername}/assessments/pruefungen")
    public PruefungInitialisierung startPruefung(@PathVariable("benutzername")
String benutzername) {
        //retrieve all kapitel
        List<Kapitel> allKapitel = kapitelRepository.getAllKapitel();

        //select and shuffle szenarien for each kapitel
        List<Long> szenarien = new ArrayList<>();
        for (Kapitel kapitel : allKapitel) {
            szenarien.addAll(szenarioRepository
                    .getSzenarienOfKapitel(kapitel.getKapitelnummer(),
szenarienPerKapitel));
        }
        Collections.shuffle(szenarien);

        //create and store pruefung
        long assessmentID = assessmentRepository.insertPruefung(benutzername);

        return new PruefungInitialisierung(assessmentID, szenarien);
    }

    /**
     * Endpoint to complete a previously created pruefung with the given result
     */
    @PutMapping("/users/{benutzername}/assessments/pruefungen/{assessmentID}")
```

```
    public void completePruefung(@PathVariable("benutzername") String benutzername
            , @PathVariable("assessmentID") long assessmentID
            , @RequestBody List<SzenarioErgebnis> ergebnis) {
        assessmentServices.completePruefung(benutzername, assessmentID, ergebnis);
    }

}
```

9.10.3.2 MetaInformationController.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.controllers;

import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.Kapitel;
import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.Thema;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.KapitelRepository;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.SzenarioRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

/**
 * Controller class with endpoints regarding meta information about kapitel, themen
and szenarien
 */
@RestController
@RequestMapping("/api/v1")
public class MetaInformationController {
    @Autowired
    private KapitelRepository kapitelRepository;

    @Autowired
    private SzenarioRepository szenarioRepository;

    @GetMapping("/kapitel")
    public List<Kapitel> getAllKapitel() {
        return kapitelRepository.getAllKapitel();
    }

    @GetMapping("/kapitel/{kapitelnummer}")
    public Kapitel getKapitel(@PathVariable("kapitelnummer") int kapitelnummer) {
        return kapitelRepository.getKapitel(kapitelnummer);
    }

    @GetMapping("kapitel/{kapitelnummer}/themen")
    public List<Thema> getSzenarien(@PathVariable("kapitelnummer") int
kapitelnummer) {
        return kapitelRepository.getThemen(kapitelnummer);
    }

    @GetMapping("kapitel/{kapitelnummer}/themen/{themanummer}/szenarien")
    public List<Long> getSzenarien(@PathVariable("kapitelnummer") int kapitelnummer
            , @PathVariable("themanummer") int themanummer) {
        return szenarioRepository.getSzenarienOfThema(kapitelnummer, themanummer);
    }
}
```

9.10.3.3 ResultController.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.controllers;

import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.AssessmentTyp;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.Assessment;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results.Assessme
ntResults;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results.Assessme
ntStatistics;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results.Szenario
Ergebnis;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.AssessmentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

/**
 * A controller class with endpoints to retrieve results and statistics of
previously
 * completed assessments
 */
@RestController
@RequestMapping("/api/v1")
public class ResultController {
    @Autowired
    private AssessmentRepository assessmentRepository;

    /**
     * Endpoint for the aggregated training szatistics of each assessment, that is
part of a training
     * Assessments are ordered from new to old
     * @param maximumAmount optional uri parameter, specifies the maximum amount of
trainings to return
     */
    @GetMapping("/users/{benutzername}/assessments/trainings")
    public List<AssessmentStatistics>
getAllTrainingsResults(@PathVariable("benutzername") String benutzername
            , @RequestParam("max") Optional<Integer> maximumAmount) {

        List<Assessment> trainingAssessments = assessmentRepository
                .getAllTrainingsAssessments(benutzername, maximumAmount);

        return fillAssessmentWithStatistics(benutzername, trainingAssessments);
    }

    /**
     * endpoint to retrieve the detailed result of the given assessment
     * @return array of assessment results with vortraining at index 0 and
nachtraining at 1
     */
    @GetMapping("/users/{benutzername}/assessments/trainings/{assessmentID}")
    public AssessmentResults[]
getDetailedTrainingsResult(@PathVariable("benutzername") String benutzername
            , @PathVariable("assessmentID") long assessmentID) {
```

```
        //get vortraining
        Assessment vortraining = assessmentRepository
                .getAssessment(benutzername, assessmentID,
AssessmentTyp.VORTRAINING);

        List<SzenarioErgebnis> vortrainingErgebnis = assessmentRepository
                .getSzenarioResultsOfAssessment(benutzername, assessmentID,
AssessmentTyp.VORTRAINING);

        //get nachtraining
        Assessment nachtraining = assessmentRepository.
                getAssessment(benutzername, assessmentID,
AssessmentTyp.NACHTRAINING);

        List<SzenarioErgebnis> nachtrainingErgebnis = assessmentRepository
                .getSzenarioResultsOfAssessment(benutzername, assessmentID,
AssessmentTyp.NACHTRAINING);

        return new AssessmentResults[]{
                AssessmentResults.fromAssessment(vortraining, vortrainingErgebnis),
                AssessmentResults.fromAssessment(nachtraining,
nachtrainingErgebnis)
        };
    }

    /**
     * Endpoint to retrieve all assessment statistics that are of type pruefung
     * @param max optional uri parameter, specifies the maximum amount of entries
to return
     * @return
     */
    @GetMapping("/users/{benutzername}/assessments/pruefungen")
    public ArrayList<AssessmentStatistics>
getAllPruefungsResults(@PathVariable("benutzername") String benutzername
            , @RequestParam Optional<Integer> max) {

        List<Assessment> assessments =
assessmentRepository.getAllPruefungsAssessments(benutzername, max);
        return fillAssessmentWithStatistics(benutzername, assessments);
    }

    /**
     * Endpoint to retrieve the detailed result of the given pruefungs assessment
     */
    @GetMapping("/users/{benutzername}/assessments/pruefungen/{assessmentID}")
    public AssessmentResults
getDetailedPruefungsResult(@PathVariable("benutzername") String benutzername
            , @PathVariable("assessmentID") long assessmentID) {
        Assessment pruefung = assessmentRepository.getAssessment(benutzername,
assessmentID, AssessmentTyp.PRUEFUNG);
        List<SzenarioErgebnis> ergebnis = assessmentRepository
                .getSzenarioResultsOfAssessment(benutzername, assessmentID,
AssessmentTyp.PRUEFUNG);
        return AssessmentResults.fromAssessment(pruefung, ergebnis);
    }

    /**
     * aggregate the results of the assessments to the Assessment statistics
     */
    private ArrayList<AssessmentStatistics> fillAssessmentWithStatistics(String
benutzername, List<Assessment> assessments) {
        ArrayList<AssessmentStatistics> statistics = new ArrayList<>();
```

```
        for (Assessment assessment : assessments) {
            List<SzenarioErgebnis> szenarien = assessmentRepository
                    .getSzenarioResultsOfAssessment(benutzername,
assessment.getAssessmentID(), assessment.getTyp());

            int[] countKorrektInkorrekt =
countKorrektAndInkorrektSzenarien(szenarien);
            statistics.add(AssessmentStatistics.fromAssessment(assessment
                    , countKorrektInkorrekt[0], countKorrektInkorrekt[1]));
        }
        return statistics;
    }

    /**
     * counts the amount of korrekt and inkorrekt szenarien
     *
     * @param szenarien szenarien of an assessment
     * @return array of size 2 with [korrekt, inkorrekt] counted szenarien
     */
    private int[] countKorrektAndInkorrektSzenarien(List<SzenarioErgebnis>
szenarien) {
        int[] count = {0, 0};

        for (SzenarioErgebnis ergebnis : szenarien) {
            count[ergebnis.isKorrekt() ? 0 : 1]++; //because an if statement is
just to long ;) #theartofcoding
        }
        return count;
    }

}
```

9.10.3.4 SzenarioController.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.controllers;

import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.MCFrage;
import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.Zuordnung;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.SzenarioRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.Collections;
import java.util.List;
import java.util.Random;

/**
 * Controller class with endpoints for retrieving szenarien
 */
@RestController
@RequestMapping("/api/v1")
public class SzenarioController {
    @Autowired
    private SzenarioRepository repository;

    /**
     * Endpoint to retrieve a Szenario with the given id
```

```
     *
     * @return szenario of type {@link MCFrage} or {@link Zuordnung}
     */
    @GetMapping("/szenarien/{szenarioID}")
    public ResponseEntity<?> getSzenario(@PathVariable("szenarioID") long
szenarioID) {
        Object szenario = repository.getSzenarioById(szenarioID);

        return ResponseEntity.ok(shuffleSzenarioAntworten(szenario));
    }

    /**
     * Returns a random szenario of the given kapitel
     */
    @GetMapping("/szenarien/uebung/kapitel/{kapitelnummer}")
    public ResponseEntity<?> getUebungsSzenario(@PathVariable("kapitelnummer") long
kapitelnummer) {
        List<Long> szenarien = repository.getSzenarienOfKapitel(kapitelnummer);
        int randomSzenario = new Random().nextInt(szenarien.size());
        Object szenario =
repository.getSzenarioById(szenarien.get(randomSzenario));
        return ResponseEntity.ok(shuffleSzenarioAntworten(szenario));
    }

    /**
     * shuffles the answeres of the given MCFrage or Zuordnung szenario
     * @return the szenario with shuffled answers
     */
    private Object shuffleSzenarioAntworten(Object szenario) {
        if (szenario instanceof MCFrage) {
            MCFrage mcFrage = (MCFrage) szenario;
            Collections.shuffle(mcFrage.getAntworten());
            return mcFrage;
        } else if (szenario instanceof Zuordnung) {
            Zuordnung zuordnung = (Zuordnung) szenario;
            Collections.shuffle(zuordnung.getEintraege());
            return zuordnung;
        }
        return szenario;
    }
}
```

9.10.3.5 UserController.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.controllers;

import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.BenutzerInformation;
import at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.server.ResponseStatusException;

import java.security.Principal;
import java.util.List;

/**
 * Controller class to access user information
 */
@RestController
```

```java
@RequestMapping("/api/v1")
public class UserController {
    @Autowired
    private UserRepository userRepository;

    /**
     * Endpoint to retrieve a list of all users, for the trainer users only
     */
    @GetMapping("/users")
    public List<BenutzerInformation> getUsers() {
        return userRepository.getAllUserInformation();
    }

    /**
     * Endpoint to retrieve the user information associated with the given
sessionToken
     * @return returns information about the currently logged in user
     */
    @GetMapping("/whoami")
    public BenutzerInformation rememberMe(Principal principal) {
        if (principal != null) {
            return userRepository.getUserInformation(principal.getName());
        }
        throw new ResponseStatusException(HttpStatus.FORBIDDEN, "Not
authenticated");
    }

}
```

9.10.4 Services

9.10.4.1 AssessmentServices.java

```java
package at.ac.univie.unet.a01526005.SETAServer.api.v1.services;

import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.AssessmentTyp;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.TrainingSubmissi
on;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results.Szenario
Ergebnis;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.AssessmentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.web.server.ResponseStatusException;
import reactor.core.publisher.Mono;

import java.util.ArrayList;
import java.util.List;

@Service
public class AssessmentServices {

    @Autowired
    private AssessmentRepository assessmentRepository;

    /**
     * Tries to set the first assessment of the training to complete anf if it
```

```
fails a exception is thrown.
     * After checking that given ids are valid the inserts proceed, with updating
the second assessment and
     * inserting user answers for both
     *
     * @param benutzername username
     * @param assessmentID assessment to complete
     * @param ergebnis     answers sent by the user
     */
    public void completeTraining(String benutzername, long assessmentID,
TrainingSubmission ergebnis) {
        Integer assessmentRowsUpdated = assessmentRepository
                .setAssessmentCompleted(benutzername, assessmentID,
AssessmentTyp.VORTRAINING)
                .block();

        if (assessmentRowsUpdated == null || assessmentRowsUpdated != 1) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, "benutzername
or assessmentID not found");
        }

        ArrayList<Mono<Integer>> rowsModified = new ArrayList<>();

        rowsModified.add(assessmentRepository
                .setAssessmentCompleted(benutzername, assessmentID,
AssessmentTyp.NACHTRAINING));


rowsModified.addAll(assessmentRepository.insertAssessmentResults(benutzername,
assessmentID
                , AssessmentTyp.VORTRAINING, ergebnis.getVortraining()));

rowsModified.addAll(assessmentRepository.insertAssessmentResults(benutzername,
assessmentID
                , AssessmentTyp.NACHTRAINING, ergebnis.getNachtraining()));

        for (Mono<Integer> response : rowsModified) {
            Integer rowsInserted = response.block();
            if (rowsInserted == null || rowsInserted != 1) {
                throw new RuntimeException("Could not store answer for
assessment");
            }
        }
    }

    /**
     * Tries to set the assessment of to complete and if it fails a exception is
thrown.
     * After checking that given ids are valid, the inserts proceed with
     * inserting user answers
     *
     * @param benutzername username
     * @param assessmentID assessment to complete
     * @param ergebnis     answers sent by the user
     */
    public void completePruefung(String benutzername, long assessmentID,
List<SzenarioErgebnis> ergebnis) {
        //set assessment to completed
        Integer assessmentRowsModified = assessmentRepository
                .setAssessmentCompleted(benutzername, assessmentID,
AssessmentTyp.PRUEFUNG)
                .block();
```

```
        //check for errors and then proceed
        if (assessmentRowsModified == null || assessmentRowsModified != 1) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, "benutzername
or assessmentID not found");
        }

        // insert szenario answers
        ArrayList<Mono<Integer>> resultInserts = assessmentRepository
                .insertAssessmentResults(benutzername, assessmentID,
AssessmentTyp.PRUEFUNG, ergebnis);

        for (Mono<Integer> response : resultInserts) {
            Integer rowsInserted = response.block();
            if (rowsInserted == null || rowsInserted != 1) {
                throw new RuntimeException("Could not store answer for
assessment");
            }
        }
    }

}
```

9.10.4.2 DatabaseUserDetailsService.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.services;
import at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories.UserRepository;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

/**
 * Service that acts as middleware between the applications user repository
 * and the spring security framework
 */
@Service
public class DatabaseUserDetailsService implements UserDetailsService {
    private UserRepository repository;

    public DatabaseUserDetailsService(UserRepository repository) {
        this.repository = repository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        UserDetails userDetails = repository
                .getUser(username);

        if (userDetails == null) {
            throw new UsernameNotFoundException(username);
        }
        return userDetails;
    }
}
```

9.10.5 Repositories

9.10.5.1 AssessmentRepository.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories;

import
```

```
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.AssessmentTyp;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.Assessment;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results.Szenario
Ergebnis;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.data.r2dbc.core.R2dbcEntityTemplate;
import org.springframework.data.relational.core.query.Query;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Repository;
import org.springframework.web.server.ResponseStatusException;
import reactor.core.publisher.Mono;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import static org.springframework.data.relational.core.query.Criteria.where;

/**
 * Repository class to to access  Assessments from the database
 */
@Repository
public class AssessmentRepository {
    @Autowired
    private R2dbcEntityTemplate entityTemplate;

    /**
     * Creates 2 new uncompleted assessments (vortraining and nachtraining) with
one assessmentID
     *
     * @param benutzername user who does the assessment
     * @return assessmentID of the newly created assessments
     */
    public Long insertTraining(String benutzername) {
        String sql = "INSERT Assessment (typ, benutzername) VALUES ('VORTRAINING',
:benutzername);" +
                " INSERT Assessment (assessmentID, typ, benutzername) VALUES
(LAST_INSERT_ID(), 'NACHTRAINING', :benutzername);" +
                " SELECT LAST_INSERT_ID() AS id ";

        return entityTemplate.getDatabaseClient()
                .sql(sql)
                .bind("benutzername", benutzername)
                .fetch()
                .one()
                .map(stringObjectMap -> ((BigInteger)
stringObjectMap.get("id")).longValue())
                .block();
    }

    /**
     * Creates and stores a new pruefung
     * @param benutzername of the user for the pruefung
     * @return the assessmentID of the pruefung
     */
    public Long insertPruefung(String benutzername) {
        String sql = "INSERT Assessment (typ, benutzername) VALUES ('PRUEFUNG',
:benutzername);" +
                " SELECT LAST_INSERT_ID() AS id ";
```

```
        return entityTemplate.getDatabaseClient()
                .sql(sql)
                .bind("benutzername", benutzername)
                .fetch()
                .one()
                .map(stringObjectMap -> ((BigInteger)
stringObjectMap.get("id")).longValue())
                .block();
    }

    /**
     * Update the complete flag in the database
     * @return rows modified, should be 1 on success
     */
    public Mono<Integer> setAssessmentCompleted(String benutzername, long
assessmentID, AssessmentTyp typ) {
        String sql = "UPDATE Assessment SET abgeschlossen=True " +
                "WHERE benutzername=:benutzername AND assessmentID=:assessmentID
AND typ=:typ";
        return entityTemplate
                .getDatabaseClient()
                .sql(sql)
                .bind("benutzername", benutzername)
                .bind("assessmentID", assessmentID)
                .bind("typ", typ.name())
                .fetch()
                .rowsUpdated();
    }

    /**
     * Inserts results for each szenario for an assessment of an user
     * Inserts are done reactively and separate to encourage concurrency of the
insert streams
     *
     * @return list of independent Monos containing amount of rows updated for each
insert which
     * should be 1 if successful
     */
    public ArrayList<Mono<Integer>> insertAssessmentResults(String benutzername,
long assessmentID
            , AssessmentTyp typ, List<SzenarioErgebnis> antworten) {
        String sql = "INSERT benutzer_beantwortet_szenario (benutzername,
assessmentID, typ, szenarioID, korrekt)" +
                "VALUES (:benutzername, :assessmentID, :typ, :szenarioID,
:korrekt)";


        ArrayList<Mono<Integer>> responses = new ArrayList<>();
        antworten.forEach(szenarioErgebnis -> {
                    responses.add(entityTemplate
                            .getDatabaseClient()
                            .sql(sql)
                            .bind("benutzername", benutzername)
                            .bind("assessmentID", assessmentID)
                            .bind("typ", typ.name())
                            .bind("szenarioID", szenarioErgebnis.getSzenarioID())
                            .bind("korrekt", szenarioErgebnis.isKorrekt())
                            .fetch()
                            .rowsUpdated());
                }
        );
        return responses;
```

```
    }

    /**
     * Accesses all stored assessments of type training
     * @param max maximum amount of trainings to return
     * @return assessments ordered from new to old
     */
    public List<Assessment> getAllTrainingsAssessments(String benutzername,
Optional<Integer> max) {
        Query query = Query.query(where("benutzername")
                .is(benutzername)
                .and("typ")
                .in(AssessmentTyp.VORTRAINING.name(),
AssessmentTyp.NACHTRAINING.name()))

.sort(Sort.by("assessmentID").descending().and(Sort.by("typ").descending()));

        if (max.isPresent()) {
            //2 assessments per training
            max = Optional.of(max.get() * 2);
        }

        return getAllAssessments(max, query);
    }

    /**
     * Accesses all stored assessments of type pruefung
     * @param max maximum amount of trainings to return
     * @return assessments ordered from new to old
     */
    public List<Assessment> getAllPruefungsAssessments(String benutzername,
Optional<Integer> max) {
        Query query = Query.query(where("benutzername")
                .is(benutzername)
                .and("typ")
                .in(AssessmentTyp.PRUEFUNG.name()))
                .sort(Sort.by("assessmentID").descending());

        return getAllAssessments(max, query);
    }

    /**
     * Executes the given query on the Assessment table of the database
     */
    private List<Assessment> getAllAssessments(Optional<Integer> max, Query query)
{
        if (max.isPresent()) {
            query = query.limit(max.get());
        }

        return entityTemplate
                .select(query, Assessment.class)
                .collectList()
                .block();
    }

    /**
     * Access a specific assessment from the database
     */
    public Assessment getAssessment(String benutzername, long assessmentID,
AssessmentTyp typ) {
        Assessment assessment = entityTemplate
                .select(Assessment.class)
```

```
                .matching(Query.query(where("benutzername")
                        .is(benutzername)
                        .and("assessmentID")
                        .is(assessmentID).and("typ").is(typ.name())))
                .one()
                .block();

        if (assessment == null) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, "assessment not
found");
        }
        return assessment;
    }

    /**
     * Access the results for each szenario within a specific assessment
     */
    public List<SzenarioErgebnis> getSzenarioResultsOfAssessment(String
benutzername, long assessmentID, AssessmentTyp typ) {
        return entityTemplate
                .select(SzenarioErgebnis.class)
                .matching(Query.query(
                        where("benutzername")
                                .is(benutzername)
                                .and("assessmentID")
                                .is(assessmentID)
                                .and("typ")
                                .is(typ.name())))
                .all()
                .collectList()
                .block();
    }
}
```

9.10.5.2 KapitelRepository.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories;

import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.Kapitel;
import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.Thema;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.r2dbc.core.R2dbcEntityTemplate;
import org.springframework.data.relational.core.query.Query;
import org.springframework.stereotype.Repository;

import java.util.List;

import static org.springframework.data.domain.Sort.by;
import static org.springframework.data.relational.core.query.Criteria.where;
import static
org.springframework.data.relational.core.query.CriteriaDefinition.empty;

/**
 * A repository class to access data about kapitel from the database
 */
@Repository
public class KapitelRepository {
    @Autowired
    private R2dbcEntityTemplate entityTemplate;

    public List<Kapitel> getAllKapitel() {
        return entityTemplate
```

```
                .select(Kapitel.class)
                .from("Kapitel")

.matching(Query.query(empty()).sort(by("kapitelnummer").ascending()))
                .all()
                .collectList()
                .block();
    }

    public Kapitel getKapitel(long kapitelnummer) {
        return entityTemplate
                .select(Kapitel.class)
                .from("Kapitel")
                .matching(Query.query(where("kapitelnummer").is(kapitelnummer)))
                .first().block();
    }

    public List<Thema> getThemen(long kapitelnummer) {
        return entityTemplate
                .select(Thema.class)
                .from("Thema")
                .matching(Query
                        .query(where("kapitelnummer")
                                .is(kapitelnummer))
                        .sort(by("themanummer").ascending()))
                .all()
                .collectList()
                .block();
    }
}
```

9.10.5.3 SzenarioRepository.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories;

import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.MCFrage;
import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.Szenario;
import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.Zuordnung;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.r2dbc.core.R2dbcEntityTemplate;
import org.springframework.data.relational.core.query.Query;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Repository;
import org.springframework.web.server.ResponseStatusException;

import java.util.List;
import java.util.Optional;

import static org.springframework.data.relational.core.query.Criteria.where;

/**
 * A repository class to access szenario data from the database
 */
@Repository
public class SzenarioRepository {
    @Autowired
    private R2dbcEntityTemplate entityTemplate;

    /**
     * @return a list of szenarioIDs of all szenarien within the given kapitel
     */
    public List<Long> getSzenarienOfKapitel(long kapitelnummer) {
```

```
        return entityTemplate.getDatabaseClient()
                .sql("SELECT szenarioID FROM Szenario WHERE kapitelnummer = ?")
                .bind(0, kapitelnummer)
                .fetch()
                .all()
                .map(stringObjectMap -> (Long) stringObjectMap.get("szenarioID"))
                .collectList()
                .block();
    }

    /**
     *
     * @param limit maximum amount of szenarien to return
     * @return a list of szenarioIDs of all szenarien within the given kapitel
     */
    public List<Long> getSzenarienOfKapitel(long kapitelnummer, int limit) {
        return entityTemplate.getDatabaseClient()
                .sql("SELECT szenarioID FROM Szenario WHERE kapitelnummer = ? LIMIT
?")
                .bind(0, kapitelnummer)
                .bind(1, limit)
                .fetch()
                .all()
                .map(stringObjectMap -> (Long) stringObjectMap.get("szenarioID"))
                .collectList()
                .block();
    }

    /**
     *
     * @return a list of szenarioIDs of all szenarien within the given thema
     */
    public List<Long> getSzenarienOfThema(long kapitelnummer, long themanummer) {
        return entityTemplate.getDatabaseClient()
                .sql("SELECT szenarioID FROM Szenario WHERE kapitelnummer = ? AND
themanummer = ?")
                .bind(0, kapitelnummer)
                .bind(1, themanummer)
                .fetch()
                .all()
                .map(stringObjectMap -> (Long) stringObjectMap.get("szenarioID"))
                .collectList()
                .block();
    }

    /**
     * Retrieves the szenario with the given id from the database
     * Returns a MCFrage or a Zuordnung szenario
     * When the szenario with the given id doesn't exist, a ResponseStatusException
with 404 is thrown
     *
     * @param szenarioID id of the sceenario to retrieve
     * @return A
     */
    public Object getSzenarioById(long szenarioID) {
        Optional<MCFrage> mcFrage = getMCFrageByID(szenarioID);
        if (mcFrage.isPresent()) {
            return mcFrage.get();
        } else {
            Optional<Zuordnung> zuordnung = getZuordnungByID(szenarioID);
            if (zuordnung.isPresent()) {
                return zuordnung.get();
            }
```

```
        }
        throw new ResponseStatusException(HttpStatus.NOT_FOUND, "Szenario not
found");
    }

    private Optional<MCFrage> getMCFrageByID(long szenarioID) {
        Optional<MCFrage> mcFrage = entityTemplate
                .select(MCFrage.class)
                .matching(Query.query(where("szenarioID").is(szenarioID)))
                .one()
                .blockOptional();

        mcFrage.ifPresent(frage -> frage.setAntworten(getMCAntworten(szenarioID)));
        return mcFrage;
    }

    private List<MCFrage.Antwort> getMCAntworten(long szenarioID) {
        return entityTemplate
                .select(MCFrage.Antwort.class)
                .matching(Query.query(where("szenarioID").is(szenarioID)))
                .all()
                .collectList()
                .block();
    }


    private Optional<Zuordnung> getZuordnungByID(long szenarioID) {
        Optional<Zuordnung> zuordnung = entityTemplate
                .select(Zuordnung.class)
                .matching(Query.query(where("szenarioID").is(szenarioID)))
                .one()
                .blockOptional();


        zuordnung.ifPresent(z -> z.setEintraege(getEintraege(szenarioID)));
        return zuordnung;
    }

    /**
     * Retrieves all eintraege from an zuordnungs szenario. Existence of the
Zuordnung should be
     * verified before calling this method
     *
     * @param szenarioID of the Zuordnung
     * @return list of Eintraege of the zuordnung
     */
    private List<Zuordnung.Eintrag> getEintraege(long szenarioID) {
        return entityTemplate
                .select(Zuordnung.Eintrag.class)
                .matching(Query.query(where("szenarioID").is(szenarioID)))
                .all()
                .collectList()
                .block();
    }
}
```

9.10.5.4 UserRepository.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.repositories;

import at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.BenutzerInformation;
import org.springframework.beans.factory.annotation.Autowired;
```

```java
import org.springframework.data.domain.Sort;
import org.springframework.data.r2dbc.core.R2dbcEntityTemplate;
import org.springframework.data.relational.core.query.Query;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Repository;

import java.util.List;

import static org.springframework.data.relational.core.query.Criteria.where;

/**
 * Repository class to access User information from the database
 */
@Repository
public class UserRepository {
    @Autowired
    private R2dbcEntityTemplate entityTemplate;

    public List<BenutzerInformation> getAllUserInformation() {
        return entityTemplate
                .select(BenutzerInformation.class)
                .matching(Query.empty().sort(Sort.by("nachname").ascending()))
                .all()
                .collectList()
                .block();
    }


    /**
     * Returns the user information for spring security
     */
    public UserDetails getUser(String benutzername) {
        return entityTemplate.getDatabaseClient()
                .sql("SELECT * FROM User WHERE benutzername=:benutzername")
                .bind("benutzername", benutzername)
                .fetch()
                .one()
                .map(stringObjectMap -> User.builder()
                        .username((String) stringObjectMap.get("benutzername"))
                        .password((String) stringObjectMap.get("passwort"))
                        .authorities("ROLE_" + (String)
stringObjectMap.get("rolle"))
                        .build())
                .block();
    }


    public BenutzerInformation getUserInformation(String benutzername) {
        return entityTemplate
                .selectOne(Query.query(where("benutzername").is(benutzername)),
BenutzerInformation.class)
                .block();
    }
}
```

9.10.6 Entities

9.10.6.1 BenutzerInformation.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities;

import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Table;

/**
 * Entity mapping to the User table of the database
 */
@Table("User")
public class BenutzerInformation {
    @Id
    private String benutzername;
    private String vorname;
    private String nachname;
    private String rolle;

    public BenutzerInformation(String benutzername, String vorname, String
nachname, String rolle) {
        this.benutzername = benutzername;
        this.vorname = vorname;
        this.nachname = nachname;
        this.rolle = rolle;
    }

    public String getBenutzername() {
        return benutzername;
    }

    public String getVorname() {
        return vorname;
    }

    public String getNachname() {
        return nachname;
    }

    public String getRolle() {
        return rolle;
    }
}
```

9.10.6.2 Kapitel.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities;

import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Table;


@Table
public class Kapitel {
    @Id
    private long kapitelnummer;
    private String titel;
    private String beschreibung;

    public Kapitel(int kapitelnummer, String titel, String beschreibung) {
        this.kapitelnummer = kapitelnummer;
        this.titel = titel;
```

```
        this.beschreibung = beschreibung;
    }

    public long getKapitelnummer() {
        return kapitelnummer;
    }

    public String getTitel() {
        return titel;
    }

    public String getBeschreibung() {
        return beschreibung;
    }
}
```

9.10.6.3 MCFrage.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities;

import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;

import java.util.List;

@Table("mc_szenario_view")
public class MCFrage extends Szenario {
    private String frage;
    private String bild_url;
    @Transient
    private List<Antwort> antworten;

    public MCFrage(long kapitelnummer, long themanummer, long szenarioID, String
frage, String bild_url) {
        super(kapitelnummer, themanummer, szenarioID, "MCSzenario");
        this.frage = frage;
        this.bild_url = bild_url == null ? "" : bild_url;
    }

    @Table("MC_Antwort")
    public static class Antwort {
        @Id
        @Column("antwortID")
        private long antwortID;
        private String text;
        private boolean korrekt;

        public Antwort(long antwortID, String text, boolean korrekt) {
            this.antwortID = antwortID;
            this.text = text;
            this.korrekt = korrekt;
        }

        public long getAntwortID() {
            return antwortID;
        }

        public String getText() {
            return text;
        }
```

```
        public boolean isKorrekt() {
            return korrekt;
        }
    }

    public void setAntworten(List<Antwort> antworten) {
        this.antworten = antworten;
    }

    public String getFrage() {
        return frage;
    }

    public String getBild_url() {
        return bild_url;
    }

    public List<Antwort> getAntworten() {
        return antworten;
    }
}
```

9.10.6.4 Szenario.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities;

import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;

@Table("Szenario")
public class Szenario {
    @Id
    @Column("szenarioID")
    private long szenarioID;
    private long kapitelnummer;
    private long themanummer;
    @Transient
    private final String TYP;

    public Szenario(long kapitelnummer, long themanummer, long szenarioID, String
TYP) {
        this.kapitelnummer = kapitelnummer;
        this.themanummer = themanummer;
        this.szenarioID = szenarioID;
        this.TYP = TYP;
    }

    public long getKapitelnummer() {
        return kapitelnummer;
    }

    public long getThemanummer() {
        return themanummer;
    }

    public long getSzenarioID() {
        return szenarioID;
```

```
    }

    public String getTYP() {
        return TYP;
    }
}
```

9.10.6.5 Thema.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities;

import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Table;

@Table
public class Thema {
    @Id
    private long themanummer;
    private String titel;

    public Thema(long themanummer, String titel) {
        this.themanummer = themanummer;
        this.titel = titel;
    }

    public long getThemanummer() {
        return themanummer;
    }

    public String getTitel() {
        return titel;
    }
}
```

9.10.6.6 Zuordnung.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities;

import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;

import java.util.List;

@Table("zuordnung_szenario_view")
public class Zuordnung extends Szenario {
    private String anweisung;
    @Transient
    private List<Eintrag> eintraege;

    public Zuordnung(long kapitelnummer, long themanummer, long szenarioID, String
anweisung) {
        super(kapitelnummer, themanummer, szenarioID, "Zuweisung");
        this.anweisung = anweisung;
    }

    @Table("Eintrag")
    public static class Eintrag {
        @Id
        @Column("eintragID")
        private long eintragID;
```

```java
        private String begriff;
        private String beschreibung;

        public Eintrag(long eintragID, String begriff, String beschreibung) {
            this.eintragID = eintragID;
            this.begriff = begriff;
            this.beschreibung = beschreibung;
        }

        public long getEintragID() {
            return eintragID;
        }

        public String getBegriff() {
            return begriff;
        }

        public String getBeschreibung() {
            return beschreibung;
        }
    }

    public void setEintraege(List<Eintrag> eintraege) {
        this.eintraege = eintraege;
    }

    public String getAnweisung() {
        return anweisung;
    }

    public List<Eintrag> getEintraege() {
        return eintraege;
    }
}
```

9.10.6.7 Assessment.java

```java
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments;

import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;

import java.time.LocalDateTime;

/**
 * Entity mapped to the Assessment table of the database
 * Database key comprises of the assessmentID and the typ
 */
@Table("Assessment")
public class Assessment {
    @Id
    @Column("assessmentID")
    private long assessmentID;
    private AssessmentTyp typ;
    private boolean abgeschlossen;
    private LocalDateTime zeitpunkt;

    public Assessment(long assessmentID, AssessmentTyp typ, boolean abgeschlossen,
LocalDateTime zeitpunkt) {
        this.assessmentID = assessmentID;
        this.typ = typ;
```

```
        this.abgeschlossen = abgeschlossen;
        this.zeitpunkt = zeitpunkt;
    }

    public long getAssessmentID() {
        return assessmentID;
    }

    public AssessmentTyp getTyp() {
        return typ;
    }

    public boolean isAbgeschlossen() {
        return abgeschlossen;
    }

    public LocalDateTime getZeitpunkt() {
        return zeitpunkt;
    }
}
```

9.10.6.8 AssessmentTyp.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments;

public enum AssessmentTyp {
    VORTRAINING,
    NACHTRAINING,
    PRUEFUNG
}
```

9.10.6.9 PruefungsInitialisierung.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments;

import java.util.List;

/**
 * A newly created pruefung. Contains the id of the assessment and a list of
szenarioIDs
 * that should be completed in this pruefung
 */
public class PruefungInitialisierung {
    private long assessmentID;
    private List<Long> szenarien;

    public PruefungInitialisierung(long assessmentID, List<Long> szenarien) {
        this.assessmentID = assessmentID;
        this.szenarien = szenarien;
    }

    public long getAssessmentID() {
        return assessmentID;
    }

    public List<Long> getSzenarien() {
        return szenarien;
    }
}
```

9.10.6.10 TrainingsInitialisierung.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments;

import java.util.List;

/**
 * A newly created Training
 * Contains lists of szenarioIDs of szenarien for each assessment that should be
 * answered in that assessment
 */
public class TrainingInitialisierung {
    private long assessmentID;
    private List<Long> vortraining;
    private List<Long> nachtraining;

    public TrainingInitialisierung(long assessmentID, List<Long> vortraining,
List<Long> nachtraining) {
        this.assessmentID = assessmentID;
        this.vortraining = vortraining;
        this.nachtraining = nachtraining;
    }

    public long getAssessmentID() {
        return assessmentID;
    }

    public List<Long> getVortraining() {
        return vortraining;
    }

    public List<Long> getNachtraining() {
        return nachtraining;
    }
}
```

9.10.6.11 TrainingsSubmission.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments;

import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results.Szenario
Ergebnis;

import java.util.List;

/**
 * Wrapper class to submit the user answers for a training
 * (Pruefung is submit with just a list of SzenarioErgebnis)
 */
public class TrainingSubmission {
    private List<SzenarioErgebnis> vortraining;
    private List<SzenarioErgebnis> nachtraining;

    public TrainingSubmission(List<SzenarioErgebnis> vortraining,
List<SzenarioErgebnis> nachtraining) {
        this.vortraining = vortraining;
        this.nachtraining = nachtraining;
    }

    public List<SzenarioErgebnis> getVortraining() {
        return vortraining;
    }
```

```
    public List<SzenarioErgebnis> getNachtraining() {
        return nachtraining;
    }
}
```

9.10.6.12 AssessmentResults.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results;

import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.Assessment;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.AssessmentTyp;

import java.time.LocalDateTime;
import java.util.List;

/**
 * A completed assessment with all results of the user answered szenarien
 */
public class AssessmentResults extends Assessment {
    private List<SzenarioErgebnis> szenarien;

    public AssessmentResults(long assessmentID, AssessmentTyp typ, boolean
abgeschlossen, LocalDateTime zeitpunkt, List<SzenarioErgebnis> szenarien) {
        super(assessmentID, typ, abgeschlossen, zeitpunkt);
        this.szenarien = szenarien;
    }

    public static AssessmentResults fromAssessment(Assessment assessment,
List<SzenarioErgebnis> szenarien) {
        return new AssessmentResults(assessment.getAssessmentID(),
assessment.getTyp()
                , assessment.isAbgeschlossen(), assessment.getZeitpunkt(),
szenarien);
    }

    public List<SzenarioErgebnis> getSzenarien() {
        return szenarien;
    }
}
```

9.10.6.13 AssessmentStatistics.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results;

import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.Assessment;
import
at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.AssessmentTyp;

import java.time.LocalDateTime;

/**
 * Assessment with aggregated results.
 * Contains the amount of correct and incorrect szenarien
 */
public class AssessmentStatistics extends Assessment {
    private int korrekt;
    private int inkorrekt;
```

```
    public AssessmentStatistics(long assessmentID, AssessmentTyp typ, boolean
abgeschlossen, LocalDateTime zeitpunkt, int korrekt, int inkorrekt) {
        super(assessmentID, typ, abgeschlossen, zeitpunkt);
        this.korrekt = korrekt;
        this.inkorrekt = inkorrekt;
    }

    public int getKorrekt() {
        return korrekt;
    }

    public int getInkorrekt() {
        return inkorrekt;
    }

    public static AssessmentStatistics fromAssessment(Assessment assessment, int
korrekt, int inkorrekt) {
        return new AssessmentStatistics(assessment.getAssessmentID(),
assessment.getTyp()
                , assessment.isAbgeschlossen(), assessment.getZeitpunkt(), korrekt,
inkorrekt);
    }
}
```

9.10.6.14 SzenarioErgebnis.java

```
package at.ac.univie.unet.a01526005.SETAServer.api.v1.entities.assessments.results;

import org.springframework.data.annotation.Id;
import org.springframework.data.relational.core.mapping.Column;
import org.springframework.data.relational.core.mapping.Table;

@Table("benutzer_beantwortet_szenario")
public class SzenarioErgebnis {
    @Id
    @Column("szenarioID")
    private long szenarioID;
    private boolean korrekt;

    public SzenarioErgebnis(long szenarioID, boolean korrekt) {
        this.szenarioID = szenarioID;
        this.korrekt = korrekt;
    }

    public long getSzenarioID() {
        return szenarioID;
    }

    public boolean isKorrekt() {
        return korrekt;
    }
}
```

9.11 Client

9.11.1 „assessment“ Component

9.11.1.1 assessment.component.html

```
<h3 class="title-sm">{{title}}</h3>
<h2 *ngIf="typeCheck.isSzenarioMCFrage(szenario)">{{szenario.frage}}</h2>
```

```
<h2 *ngIf="typeCheck.isSzenarioZuordnung(szenario)">{{szenario.anweisung}}</h2>
<div *ngIf="typeCheck.isSzenarioMCFrage(szenario) && szenario.bild_url != ''" class="img-
container">
    <a target="_blank" [href]="appDao.serverBaseUrl+szenario.bild_url">
        <img  [src]="appDao.serverBaseUrl+szenario.bild_url" alt="">
    </a>
</div>

<div class="content">
    <div *ngIf="typeCheck.isSzenarioMCFrage(szenario)" class="mcfrage">
        <div *ngFor="let antwort of szenario.antworten"
            (click)="toggleMcAntwort(antwort.antwortID)"
            [ngClass]="{'selected': isAusgewaehlt(antwort.antwortID), 'correct': feedBackMo
de && isCorrectlyAnswered(antwort), 'incorrect': feedBackMode && isIncorrectlyAnswered(antw
ort)}">
            {{antwort.text}}
        </div>
    </div>

    <div *ngIf="typeCheck.isSzenarioZuordnung(szenario)" class="zuordnung">
        <ng-container *ngFor="let eintrag of szenario.eintraege">
            <p>{{eintrag.beschreibung}}</p>
            <select
                [ngClass]="{'correct-
pill': feedBackMode && isZuweisungsAntwortCorrect(eintrag), 'incorrect-
pill': feedBackMode && !isZuweisungsAntwortCorrect(eintrag)}"
                [value]="getSelectedEintrag(eintrag)"
                (change)="selectedEintrag(eintrag, select.value)" #select>

                <option value="" [disabled]="feedBackMode"> </option>
                <option *ngFor="let optionEintrag of sortZuordnungsbegriffe(szenario.eintra
ege)"
                    [value]="optionEintrag.begriff"
                    [selected]="optionEintrag.begriff == getSelectedEintrag(eintrag)"
                    [disabled]="feedBackMode && !(optionEintrag.begriff == getSelectedEintr
ag(eintrag))"
                    [ngClass]="{'correct-
pill': feedBackMode && eintrag.begriff == optionEintrag.begriff}">{{optionEintrag.begriff}}
</option>
            </select>
        </ng-container>
    </div>
</div>
<div class="buttons">
    <button *ngIf="exitBtnText != ''" class="confirm-btn primary-btn" [ngClass]="{'red-
btn': !feedBackMode && exitBtnText != 'zu den Inhalten'}" (click)="exitBtnClicked()">{{exit
BtnText}}</button>
    <button *ngIf="confirmBtnText != ''" class="confirm-btn primary-dark-
btn" (click)="confirmButtonClicked()">{{confirmBtnText}}</button>
</div>


<nav *ngIf="szenarioFortschritt !== undefined && szenarioResults === undefined">
    <button *ngFor="let item of szenarioFortschritt; let i = index"
        [ngClass]="{'answered': item === true, 'current': currentSzenarioIndex === i}"
        (click)="szenarioNavClicked(i)">{{i+1}}</button>
</nav>

<nav *ngIf="szenarioResults !== undefined && szenarioFortschritt === undefined">
    <button *ngFor="let item of szenarioResults; let i = index"
```

```
        [ngClass]="{'correct-pill': item, 'incorrect-
pill': !item,  'current': currentSzenarioIndex === i}"
        (click)="szenarioNavClicked(i)">{{i+1}}</button>
</nav>
```

9.11.1.2 assessment.component.ts

```
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';
import { AssessmentNavigation } from '../../../model/assessmentNavigation';
import { MCAntwort } from '../../../model/szenarien/MCAntwort';
import { MCFrage } from '../../../model/szenarien/MCFrage';
import { Zuordnung } from '../../../model/szenarien/Zuordnung';
import { ZuordnungsEintrag } from '../../../model/szenarien/ZuordnungsEintrag';
import { AppDaoService } from '../../../services/app-dao.service';
import { TypeCheckService } from '../../../services/type-check.service';



/**
 * Reusable Assessment component that displays a Szenario as part of an assessment.
 * This component is used at several pages, anywhere to display a szenario
 *
 * all data, including the text for the buttons is passed via inputs, making this component
 * perfectly reusable
 *
 * It has a feedback mode that, when toggled to true, makes answered unselectable
 * shows the correct solution, and if answered the wrong selections
 *
 * The component handles szenarien of type MCFragen or Zuordnung itself and both can be set
 *
 */
@Component({
  selector: 'app-assessment',
  templateUrl: './assessment.component.html',
  styleUrls: ['./assessment.component.sass']
})
export class AssessmentComponent implements OnInit {
  /**
   * click events for the confirmation button.
   * passes a map with eintragID and selected begriffe if a Zuordnungsszenario is displayed

   * passes a set with selected MCAntwort IDs, if a MCFrage is displayed
   *
   */
  @Output() confirmBtn = new EventEmitter<Map<number, string> | Set<number>>();

  /**
   * click events for the exit button button.
   * passes a map with eintragID and selected begriffe if a Zuordnungsszenario is displayed
   * passes a set with selected MCAntwort IDs, if a MCFrage is displayed
   *
   */
  @Output() exitBtn = new EventEmitter<Map<number, string> | Set<number>>();

  /**
   * passess a event if the bottom navigation bar is clicked
   */
  @Output() szenarioNav = new EventEmitter<AssessmentNavigation>();

  /**
```

```
 * the title of the page to display
 */
@Input() title: string = "";

/**
 * text of the (right) confirmation button
 */
@Input() confirmBtnText: string = "Weiter";

/**
 * text of the (left) exit button
 */
@Input() exitBtnText: string = "";

/**
 * the szenario, that is currently displayed
 */
@Input() szenario?: MCFrage | Zuordnung;

/**
 * current user selected answers, if a MCFrage is displayed
 * Set with selected answerIDs
 */
@Input() mcAntwortSelected!: Set<number>;

/**
 * current user selected answers, if a Zuordnung is displayed
 * A map with eintragID and the selected begriff
 */
@Input() zuordnungsAuswahl!: Map<number, string>;

/**
 * the index of the currently selected szenario
 */
@Input() currentSzenarioIndex: number = -1;

/**
 * a list used with feedBackMode = false signaling if a szenario has been answered
 * boolean at a index signals a answer the szenario with the same index
 */
@Input() szenarioFortschritt?: boolean[];

/**
 * a list used in feedback mode signaling if the szenario has been answered correctly
 */
@Input() szenarioResults?: boolean[];

@Input() feedBackMode: boolean = false;

constructor(
  public appDao: AppDaoService,
  public typeCheck: TypeCheckService
) { }

ngOnInit(): void {

}

confirmButtonClicked() {
  if (this.typeCheck.isSzenarioMCFrage(this.szenario)) {
    this.confirmBtn.emit(this.mcAntwortSelected);
```

```
    } else if (this.typeCheck.isSzenarioZuordnung(this.szenario)) {
      this.confirmBtn.emit(this.zuordnungsAuswahl);
    }
  }

  exitBtnClicked() {
    if (this.typeCheck.isSzenarioMCFrage(this.szenario)) {
      this.exitBtn.emit(this.mcAntwortSelected);

    } else if (this.typeCheck.isSzenarioZuordnung(this.szenario)) {
      this.exitBtn.emit(this.zuordnungsAuswahl);

    }
  }

  szenarioNavClicked(position: number) {
    if (this.typeCheck.isSzenarioMCFrage(this.szenario)) {
      this.szenarioNav.emit({ answer: this.mcAntwortSelected, nextSzenarioIndex: position }
);

    } else if (this.typeCheck.isSzenarioZuordnung(this.szenario)) {
      this.szenarioNav.emit({ answer: this.zuordnungsAuswahl, nextSzenarioIndex: position }
);
    }
  }

  /**
   * Selects or unselects the MCAntwort only if its not feedback mode
   * @param antwortID of the selected MCAntwort
   */
  toggleMcAntwort(antwortID: number) {
    if (!this.feedBackMode) {
      if (this.mcAntwortSelected.has(antwortID)) {
        this.mcAntwortSelected.delete(antwortID);
      } else {
        this.mcAntwortSelected.add(antwortID);
      }
    }

  }

  isAusgewaehlt(antwortID: number): boolean {
    return this.mcAntwortSelected.has(antwortID);
  }

  isCorrectlyAnswered(antwort: MCAntwort): boolean {
    if (this.typeCheck.isSzenarioMCFrage(this.szenario)) {
      let ausgewaehlt = this.mcAntwortSelected.has(antwort.antwortID);

      return ausgewaehlt && antwort.korrekt;
    }
    return false;
  }

  isIncorrectlyAnswered(antwort: MCAntwort): boolean {
    if (this.typeCheck.isSzenarioMCFrage(this.szenario)) {
      let ausgewaehlt = this.mcAntwortSelected.has(antwort.antwortID);

      return ausgewaehlt && !antwort.korrekt || !ausgewaehlt && antwort.korrekt;
    }
```

```
    return false;
  }

  /**
   * stored the selected begriff to the eintrag
   * @param eintrag
   * @param value begriff
   */
  selectedEintrag(eintrag: ZuordnungsEintrag, value: string) {
    this.zuordnungsAuswahl.set(eintrag.eintragID, value);
  }

  /**
   * returns the user selected begriff for a eintrag
   */
  getSelectedEintrag(eintrag: ZuordnungsEintrag): string {
    let selected = this.zuordnungsAuswahl.get(eintrag.eintragID);

    if (selected !== undefined) {
      return selected;
    }
    return "";
  }

  isZuweisungsAntwortCorrect(eintrag: ZuordnungsEintrag): boolean {
    return eintrag.begriff == this.zuordnungsAuswahl.get(eintrag.eintragID);
  }

  /**
   * sorts and returns the given eintraege alphabetically
   */
  sortZuordnungsbegriffe(eintrag: ZuordnungsEintrag[]): ZuordnungsEintrag[] {
    return eintrag.sort((a, b) => a.begriff.localeCompare(b.begriff))
  }
}
```

9.11.2 „pruefung-result“ Component

9.11.2.1 pruefung-result.component.ts

```
<h2 class="title">Prüfungsergebnis: </h2>

<div class="scroll-box">
    <div class="content-box">
        <div class="item-box gesammt-box">
            <h3>Gesammt: </h3>
            <h3>{{getPercent(amountGesammtKorrekt, pruefung.szenarien.length)}}% ({{amountG
esammtKorrekt}}/{{pruefung.szenarien.length}})</h3>
        </div>
        <div *ngFor="let kapitel of kapitelErfolg; let i = index" class="item-box">
            <ng-container *ngIf="kapitel.gesammt !==0" >
                <div class="kapitel-header">
                    <h3>Kapitel {{i+1}}</h3>
                    <h3>{{getPercent(kapitel.korrekt, kapitel.gesammt)}}% ({{kapitel.korrek
t}}/{{kapitel.gesammt}})</h3>
                </div>
                <nav>
                    <button *ngFor="let szenario of szenarioKapitelZuweisung[i]"
```

```
                        [ngClass]="{'correct-pill': szenario.korrekt, 'incorrect-
pill': !szenario.korrekt, 'disabled': btnDisabled}"
                        (click)="onClickSzenarioNav(szenario.szenarioID)"
                    >{{szenario.anzeigeNummer}}</button>
                </nav>
            </ng-container>
        </div>
    </div>

</div>
```

9.11.2.2 pruefung-result.component.ts

```
import { Component, EventEmitter, Input, OnChanges, OnInit, Output } from '@angular/core';
import { AssessmentResult } from '../../../model/api/AssessmentResult';
import { Szenario } from '../../../model/szenarien/Szenario';
import { AssessmentApiClientService } from '../../../services/assessment-api-
client.service';

/**
 * Component to display the result of a pruefung
 */
@Component({
  selector: 'app-pruefung-result',
  templateUrl: './pruefung-result.component.html',
  styleUrls: ['./pruefung-result.component.sass']
})
export class PruefungResultComponent implements OnInit, OnChanges {
  @Output() szenarioNavClicked = new EventEmitter<number>();
  @Input() pruefung!: AssessmentResult;
  @Input() btnDisabled: boolean = true;
  /**
   * mapping of szenarioIDs to their kapitel number. outer array is of size 5, one entry fo
r each kapitel
   * inner array is a list of szenarioIDs belonging to the kapitel with kapitelnumber index
+1
   */
  szenarioKapitelZuweisung: Array<Array<any>> = [];

  /**
   * array of result statistics for each kapitel.
   */
  kapitelErfolg: Array<any> = [];
  amountGesammtKorrekt: number = 0;

  constructor(
    private apiClient: AssessmentApiClientService
  ) { }

  ngOnInit(): void {
    this.updateResults();
  }

  ngOnChanges() {
    this.updateResults();
  }


  onClickSzenarioNav(szenarioID: number) {
```

```
  if (!this.btnDisabled) {
    this.szenarioNavClicked.emit(szenarioID);
  }
}

getPercent(korrekt: number, gesammt: number): number {
  if(korrekt===0 || gesammt ===0) {
    return 0;
  }
  return Math.ceil((korrekt / gesammt) * 100);
}

updateResults() {
  let szenarien:Szenario[] = [];

  for (let szenarioResult of this.pruefung.szenarien) {
    this.apiClient.requestSzenario(szenarioResult.szenarioID).subscribe(szenario => {
      szenarien.push(szenario);
      if (szenarien.length === this.pruefung.szenarien.length) {
        this.setKapitelValues(szenarien);
      }
    });
  }
}

/**
 * set the kapitelinformation
 */
setKapitelValues(szenarien: Szenario[]): void {
  // discard old values
  let amountGesammtKorrekt = 0;
  let newKapitelZuweisung: Array<Array<any>> = [];
  let newKapitelErfolg: Array<any> =[];

  for (let i = 0; i < 5; i++) {
    newKapitelZuweisung.push([])
    newKapitelErfolg.push({
      gesammt: 0,
      korrekt: 0
    });
  }


  for (let i = 0; i < this.pruefung.szenarien.length; i++) {
    //count amount gesammt korrekt
    if (this.pruefung.szenarien[i].korrekt) {
      amountGesammtKorrekt++;
    }

    //store szenarien to kapitel
    let szenarioID = this.pruefung.szenarien[i].szenarioID;
    let kapitelnummer = this.getKapitelnummerOfSzenario(szenarien, szenarioID);
    newKapitelZuweisung[kapitelnummer-1]!.push({
      anzeigeNummer: i + 1,
      szenarioID: szenarioID,
      korrekt: this.pruefung.szenarien[i].korrekt
    });

    // count amount of szenarien and correct for percentage
    newKapitelErfolg[kapitelnummer-1].gesammt++;
    if (this.pruefung.szenarien[i].korrekt) {
```

```
        newKapitelErfolg[kapitelnummer-1].korrekt++;
      }
    }

    this.amountGesammtKorrekt = amountGesammtKorrekt;
    this.szenarioKapitelZuweisung = newKapitelZuweisung;
    this.kapitelErfolg = newKapitelErfolg;
  }

  /**
   * return the kapitelnummer of the szenario with the given id, from the given szenarien
   */
  private getKapitelnummerOfSzenario(szenarien: Szenario[], id:number):number {
    for(let szenario of szenarien) {
      if(szenario.szenarioID === id) {
        return szenario.kapitelnummer;
      }
    }
    console.error("Szenario with id "+id+" not found in requested szenarien");
    return -1
  }

}
```

9.11.3 „training-result“ Component

9.11.3.1 training-result.component.html

```
<h2 class="title">Trainingsergebnis: </h2>
<h5 class="title-
sm"><b>Kapitel: </b>{{training.kapitelnummer}}. {{training.kapitelname}}</h5>
<h3>Ergebnisse im Vergleich:</h3>
<div>
    <div class="item-box">
        <div class="assessment-header">
            <h4>Vortrainings-Assessment:</h4>
            <h4>{{training.vortraining.prozent}}% ({{training.vortraining.korrekt}}/{{train
ing.vortraining.total}})</h4>
        </div>
        <nav>
            <button *ngFor="let szenario of training.vortraining.antworten; let i = index"
                [ngClass]="{'correct-pill': szenario.korrekt, 'incorrect-
pill': !szenario.korrekt, 'disabled': btnDisabled}"
                (click)="navigateToSzenario(szenario.szenarioID)">{{i+1}}</button>
        </nav>
    </div>
    <div class="item-box">
        <div class="assessment-header">
            <h4>Nachtrainings-Assessment:</h4>
            <h4>{{training.nachtraining.prozent}}% ({{training.nachtraining.korrekt}}/{{tra
ining.nachtraining.total}})</h4>
        </div>
        <nav>
            <button *ngFor="let szenario of training.nachtraining.antworten; let i = index"
                [ngClass]="{'correct-pill': szenario.korrekt, 'incorrect-
pill': !szenario.korrekt, 'disabled': btnDisabled}"
                (click)="navigateToSzenario(szenario.szenarioID)">{{i+1}}</button>
        </nav>
    </div>
```

```
</div>

<h3>Empfehlung:</h3>
<p style="white-space: pre-line">
    {{getEmpfehlungsText()}}
</p>
```

9.11.3.2 training-result.component.ts

```
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';
import { AggregatedTrainingResult } from '../../../model/AggregatedTrainingsResult';

@Component({
  selector: 'app-training-result',
  templateUrl: './training-result.component.html',
  styleUrls: ['./training-result.component.sass']
})
export class TrainingResultComponent implements OnInit {
  @Output() szenarioNavClicked = new EventEmitter<number>();

  @Input() btnDisabled: boolean = true;
  @Input() training!:AggregatedTrainingResult;

  constructor() { }

  ngOnInit(): void {
  }

  getEmpfehlungsText(){
    let prozentDiff = this.training.nachtraining.prozent - this.training.vortraining.prozen
t;
    if (this.training.nachtraining.prozent == 100) {
      return this.getVollePunktzahlText();
    }
    if( prozentDiff>0) {
      return this.getPositiveEmpfehlungText(prozentDiff);
    } else {
      return this.getNegativEmpfehlungText();
    }
  }

  navigateToSzenario(szenarioID:number){
    if(!this.btnDisabled) {
      this.szenarioNavClicked.emit(szenarioID);
    }
  }

  getNegativEmpfehlungText():string {
  return "Sie konnten sich durch dieses Training leider nicht verbessern. \
    Versuchen Sie beim nächsten Mal etwas mehr Übungsaufgaben durchzuführen, um so besser z
u werden. \
    \n\nDirekt nach einem Assessment können Sie sich Ihre falschen und die zugehörigen rich
tigen Antworten ansehen. Nutzen Sie das, um aus Ihren Fehlern zu lernen. ";
  }

  getPositiveEmpfehlungText(prozentDiff:number):string {
    return "Sie konnten sich durch dieses Training um "+prozentDiff
      + "% verbessern. Nur weiter so!";
  }
```

```
  getVollePunktzahlText():string{
    return "Ausgezeichnet!\n\n Sie haben zum Schluss die volle Punktzahl erreicht. \
    Wenn Sie in den anderen Kapiteln genau so gut abschneiden können Sie eine Prüfung durfü
hren.";
  }

}
```

9.11.4 „pruefung-page“ Component

9.11.4.1 pruefung-page.component.html

```
<div *ngIf="pruefungsController.phase == PruefungsPhase.Beendet" class="section-darker">
    <h2 class="title">Wie läuft eine Prüfung Ab?</h2>
    <div class="info">
        <h3>Modus:</h3>
        <p>Prüfungen bestehen aus zufälligen Aufgaben aller Kapitel. Dabei werden von jedem
 Kapitel 8 Aufgaben
            ausgewählt, die Sie
            beantworten müssen. Nachdem Sie alle Aufgaben beantwortet haben, werden diese a
ufgelöst und Sie können sich
            Ihre und die
            korrekten Antworten ansehen.
        </p>
        <h3>Aufgaben:</h3>
        <p>
            Es gibt zwei unterschiedliche Arten von Aufgaben, Multiple Choice Fragen und Zu
ordnungen:
            <br><br>
            In Multiple Choice Aufgaben wird Ihnen eine Frage mit zugehörigen Antworten ang
ezeigt. Sie müssen die
            korrekten
            Antworten auswählen, wobei eine, mehrere oder alle Antworten richtig sein könne
n.
            <br><br>
            In Zuordnungs-
Aufgaben werden Ihnen mehrere Texte angezeigt. Neben jedem Text befindet sich ein Menü, in
dem
            Sie einen
            Begriff auswählen können. Sie müssen den angezeigten Texten ihre zugehörigen Be
griffe zuordnen. Ein Begriff
            kann auch
            mehrmals verwendet und daher mehreren Texten zugeordnet werden.

        </p>
    </div>
</div>
<div *ngIf="pruefungsController.phase == PruefungsPhase.Beendet" class="section-lighter">
    <h2 class="title">Starten Sie eine Prüfung:</h2>
    <div class="kapitel">
        <h3>Hinweis:</h3>
        <p>Prüfungen dauern 40-60 Minuten.<br>
            Abgebrochene Prüfungen werden nicht gewertet.</p>
    </div>
    <button class="secondary-
btn" (click)="pruefungsController.startPruefung()">Prüfung Starten</button>
</div>
<div *ngIf="pruefungsController.phase != PruefungsPhase.Beendet"
    [ngClass]="{'flex-column': pruefungsController.phase == PruefungsPhase.Feedback}"
```

```
    class="full-section-lighter pruefung-box">
    <app-pruefung-result class="result-screen"
        *ngIf="pruefungsController.phase == PruefungsPhase.Feedback"
        [pruefung]="pruefungsController.pruefungsResult"
        [btnDisabled]="false"
        (szenarioNavClicked)=pruefungsController.showSzenarioResultWithId($event)>
    </app-pruefung-result>
    <app-assessment  [ngClass]="{'feedback-
mode': (pruefungsController.phase == PruefungsPhase.Feedback)}"
        *ngIf="pruefungsController.phase == PruefungsPhase.Pruefung  || pruefungsController
.phase == PruefungsPhase.Feedback"
        [title]="pruefungsController.phase"
        [confirmBtnText]="pruefungsController.confirmBtnText"
        [exitBtnText]="pruefungsController.exitBtnText" [szenario]="pruefungsController.sze
nario"
        [mcAntwortSelected]="pruefungsController.mcAntwortSelected"
        [zuordnungsAuswahl]="pruefungsController.zuordnungsAuswahl"
        [currentSzenarioIndex]="pruefungsController.pruefung.aktuellesSzenario"
        [szenarioFortschritt]="pruefungsController.szenarioFortschritt"
        [szenarioResults]="pruefungsController.szenarioResults"
        [feedBackMode]="pruefungsController.phase == PruefungsPhase.Feedback"
        (confirmBtn)="pruefungsController.confirmBtnClicked($event)"
        (exitBtn)="pruefungsController.exitBtnClicked($event)"
        (szenarioNav)="pruefungsController.chooseSzenario($event)">
    </app-assessment>

</div>
```

9.11.4.2 pruefung-controller.service.ts

```
import { Injectable } from '@angular/core';
import { ApiSzenarioAntwort } from '../model/api/ApiSzenarioAntwort';
import { AssessmentResult } from '../model/api/AssessmentResult';
import { AssessmentNavigation } from '../model/assessmentNavigation';
import { Pruefung } from '../model/Pruefung';
import { MCFrage } from '../model/szenarien/MCFrage';
import { MCFrageBeantwortet } from '../model/szenarien/MCFrageBeantwortet';
import { Zuordnung } from '../model/szenarien/Zuordnung';
import { ZuordnungBeantwortet } from '../model/szenarien/ZuordnungBeantwortet';
import { PruefungsPhase } from '../pages/pruefung/PruefungsPhasen';
import { AssessmentApiClientService } from './assessment-api-client.service';
import { ModelConverterService } from './model-converter.service';
import { TypeCheckService } from './type-check.service';


/**
 * Controller service for pruefungs page component.
 * Packaged into a own service to utilize lazy loading
 *
 * pruefung has 3 phases: beendet, pruefung, feedback
 */
@Injectable({
  providedIn: 'root'
})
export class PruefungsControllerService {
  pruefung: Pruefung = new Pruefung(-
1, []); // dummy pruefung to not having to check undefined all the time
  phase: PruefungsPhase = PruefungsPhase.Beendet;

  /**
```

```
   * currently displayed scenario
   * bound to the Input of the assessment component
   */
  szenario?: MCFrage | Zuordnung;

  /**
   * set with user selected answeres of the currently displayed MC szenario (if a mc szenar
io is displayed)
   * numbers in set are the antwortIDs of the selected answers
   * bound to the Input of the assessment component
   */
  mcAntwortSelected: Set<number> = new Set();

  /**
   * user selected answeres of the currently displayed zuordnung (if one is displayed)
   * eintragID mapped to begriff as string
   */
  zuordnungsAuswahl: Map<number, string> = new Map();

  /**
   * list of szenarien with an user answer, (true: is answered, false: is not answered)
   * index refers to index of pruefung.enthalteneSzenarien, where the szenarioID is stored
   */
  szenarioFortschritt?: boolean[];

  /**
   * list of szenario results for the feedback phase. true is correctly answered false is i
ncorrect
   * index refers to index of pruefung.enthalteneSzenarien, where the szenarioID is stored
   */
  szenarioResults?: boolean[];
  exitBtnText: string = "Abgeben";
  confirmBtnText: string = "Weiter";

  /**
   * Assessmentresult object with the pruefung result
   * bound to the Input at the pruefung result component
   */
  pruefungsResult!: AssessmentResult;


  constructor(
    private apiClient: AssessmentApiClientService,
    public typeCheck: TypeCheckService,
    private modelConverter: ModelConverterService) { }

  /**
   * initiate a new pruefung and configure all data for the assessment component
   */
  startPruefung() {
    this.apiClient.startPruefungRequest().subscribe(data => {
      this.pruefung = new Pruefung(
        (data.assessmentID as number),
        (data.szenarien as number[])
      );
      console.log("pruefung started with szenarien: " + this.pruefung.enthalteneSzenarien);

      this.exitBtnText = "Abgeben";
      this.confirmBtnText = "Weiter";
      this.phase = PruefungsPhase.Pruefung;
      this.szenarioFortschritt = [];
```

```
      for (let i = 0; i < this.pruefung.enthalteneSzenarien.length; i++) {
        this.szenarioFortschritt.push(false);
      }

      this.setAktuellesSzenario();
    });
  }


  /**
   * turn in this pruefung and switch to feedback phase
   */
  submitPruefung() {
    if (this.phase == PruefungsPhase.Pruefung) {
      let results = this.modelConverter.convertAssessmentToResults(this.pruefung.enthaltene
Szenarien, this.pruefung.szenarienBeantwortet);

      this.apiClient.submitPruefungRequest(this.pruefung.assessmentID, results)
        .subscribe(() => console.log("pruefung was submit"));

      this.setPruefungResult(results);
      this.setSzenarioResult(results);
      this.szenarioFortschritt = undefined;
      this.exitBtnText = "Beenden";
      this.phase = PruefungsPhase.Feedback;
    }
  }

  setPruefungResult(results: ApiSzenarioAntwort[]){
    this.pruefungsResult = {
      assessmentID: this.pruefung.assessmentID,
      abgeschlossen: true,
      typ: "PRUEFUNG",
      zeitpunkt: "---",
      szenarien: results
    }
  }

  setSzenarioResult(results: ApiSzenarioAntwort[]){
    this.szenarioResults = [];
    for(let szenarioID of this.pruefung.enthalteneSzenarien) {
      for (let result of results) {
        if(result.szenarioID === szenarioID) {
          this.szenarioResults.push(result.korrekt);
        }
      }
    }
    if (this.szenarioResults.length !== results.length) {
      throw new Error("pruefungs result and szenarios didn't match");
    }
  }

  /**
   * load the currently selected szenario into the assessment component
   * if a szenario isn't stored in a local variable yet, request it from the api
   */
  setAktuellesSzenario() {
    console.log("loading szenario");

    let szenarioID = this.pruefung.enthalteneSzenarien[this.pruefung.aktuellesSzenario];
    let storedSzenario = this.pruefung.getSzenarioById(szenarioID);
```

```
    if (storedSzenario !== undefined) {
      this.loadSzenarioWithAnswers(storedSzenario);
      console.log("stored szenario was loaded");

    } else {
      this.apiClient.requestSzenario(szenarioID).subscribe(
        (szenario: MCFrage | Zuordnung) => {
          this.szenario = szenario;
          this.mcAntwortSelected = new Set();
          this.zuordnungsAuswahl = new Map();
          console.log("szenario received: " + szenario.szenarioID);

        }
      );
    }
  }

  /**
   * configure the assessment component with the given szenario
   */
  loadSzenarioWithAnswers(szenario: MCFrageBeantwortet | ZuordnungBeantwortet) {
    this.szenario = szenario;
    if (this.typeCheck.isSzenarioMCFrageBeantwortet(szenario)) {
      this.mcAntwortSelected = szenario.ausgewaehlt;
      this.zuordnungsAuswahl = new Map();
    } else {
      this.mcAntwortSelected = new Set();
      this.zuordnungsAuswahl = szenario.zugeordnet;
    }
  }

  /**
   * store the given szenario answered and select the following szenario
   * @param answer map with eintragIDs and selected begriffe if it is a zuordnung, set with
 selected antwortIDs on a MCFrage
   */
  answerCurrentAndNext(answer: Map<number, string> | Set<number>) {
    this.storeSzenarioAnswer(answer);
    this.pruefung.aktuellesSzenario++;
    this.setAktuellesSzenario();
  }

  /**
   * store the given szenario answered and select requested szenario
   * @param szenarioNav
   */
  answerCurrentAndSelect(szenarioNav: AssessmentNavigation) {
    this.storeSzenarioAnswer(szenarioNav.answer);
    this.pruefung.aktuellesSzenario = szenarioNav.nextSzenarioIndex;
    this.setAktuellesSzenario();
  }

  /**
   * actually store the given answered to the selected szenario
   */
  private storeSzenarioAnswer(answer: Map<number, string> | Set<number>) {
    if (answer instanceof Set && this.typeCheck.isSzenarioMCFrage(this.szenario)) {
      this.pruefung.szenarienBeantwortet[this.pruefung.aktuellesSzenario] = this.createMCFr
ageBeantwortet(answer, this.szenario);
```

```
    } else if (answer instanceof Map && this.typeCheck.isSzenarioZuordnung(this.szenario))
{
      this.pruefung.szenarienBeantwortet[this.pruefung.aktuellesSzenario] = this.createZuor
dnungBeantwortet(answer, this.szenario);
    }

    if (this.szenarioFortschritt !== undefined && answer.size !== 0) {
      console.log("store szenario answer, its size is: " + answer.size);

      this.szenarioFortschritt[this.pruefung.aktuellesSzenario] = true;
    }
  }

  private createMCFrageBeantwortet(userAntwort: Set<number>, szenario: MCFrage): MCFrageBea
ntwortet {
    return {
      szenarioID: szenario.szenarioID,
      kapitelnummer: szenario.kapitelnummer,
      themanummer: szenario.themanummer,
      typ: szenario.typ,
      frage: szenario.frage,
      bild_url: szenario.bild_url,
      antworten: szenario.antworten,
      ausgewaehlt: userAntwort
    };
  }

  private createZuordnungBeantwortet(userAntwort: Map<number, string>, szenario: Zuordnung)
: ZuordnungBeantwortet {

    return {
      szenarioID: szenario.szenarioID,
      kapitelnummer: szenario.kapitelnummer,
      themanummer: szenario.themanummer,
      typ: szenario.typ,
      anweisung: szenario.anweisung,
      eintraege: szenario.eintraege,
      zugeordnet: userAntwort
    }
  }

  /**
   * at pruefungs phase, anwer current szenario and select next
   * at feedback phase just load next szenario
   */
  confirmBtnClicked(event: Map<number, string> | Set<number>) {
    if (this.phase == PruefungsPhase.Pruefung) {
      //if at last szenario, submit pruefung
      if (this.pruefung.aktuellesSzenario === this.pruefung.enthalteneSzenarien.length - 1)
 {
        this.storeSzenarioAnswer(event);
        this.submitPruefung()

      } else {
        this.answerCurrentAndNext(event)
      }
    } else if (this.phase == PruefungsPhase.Feedback) {
      this.pruefung.aktuellesSzenario = (this.pruefung.aktuellesSzenario+1)%this.pruefung.e
nthalteneSzenarien.length;
      this.setAktuellesSzenario();
    }
```

```
  }

  /**
   * answer current szenario and submit pruefung
   */
  exitBtnClicked(event: Map<number, string> | Set<number>) {
    if (this.phase == PruefungsPhase.Pruefung) {
      this.storeSzenarioAnswer(event);
      this.submitPruefung()
    } else if (this.phase == PruefungsPhase.Feedback) {
      this.pruefungBeenden();
    }
  }

  /**
   * called when the szenario navbar at the assessment component is clicked
   */
  chooseSzenario(szenarioIndex: AssessmentNavigation) {
    if (this.phase == PruefungsPhase.Pruefung) {
      this.answerCurrentAndSelect(szenarioIndex);

    } else if (this.phase == PruefungsPhase.Feedback) {
      this.pruefung.aktuellesSzenario = szenarioIndex.nextSzenarioIndex;
      this.setAktuellesSzenario();
      console.log("aktuelle szenario index is: " + this.pruefung.aktuellesSzenario);
    }
  }

  /**
   * display the selected szenario
   */
  showSzenarioResultWithId(szenarioID:number){
    for (let i = 0; i< this.pruefung.enthalteneSzenarien.length; i++) {
      if (this.pruefung.enthalteneSzenarien[i] === szenarioID) {
        this.pruefung.aktuellesSzenario = i;
      }
    }
    this.setAktuellesSzenario();
    console.log("aktuelle szenario index is: " + this.pruefung.aktuellesSzenario);
  }


  /**
   * reset all values for a new pruefung
   */
  private pruefungBeenden(){
    this.szenario = undefined;
    this.mcAntwortSelected = new Set();
    this.zuordnungsAuswahl = new Map();
    this.szenarioFortschritt = undefined;
    this.szenarioResults = undefined;
    this.exitBtnText = "Abgeben";
    this.phase = PruefungsPhase.Beendet;
  }
}
```

9.11.5 „training-page“ Component

9.11.5.1 training-page.component.html

```
<div *ngIf="controller.phase == TrainingsPhase.Beendet" class="section-darker introduction-
left">
    <h2 class="title">Wie läuft ein Training Ab?</h2>
    <div class="info">
        <h3>Trainings bestehen aus vier Phasen:</h3>
        <ol>
            <li>Vortrainings-Assessment</li>
            <li>Lernen der Inhalte</li>
            <li>Nachtrainings-Assessment</li>
            <li>Empfehlung</li>
        </ol>
        <h3>Assessments:</h3>
        <p>
            Sie müssen eine Reihe von zufällig ausgewählten Aufgaben lösen. Aufgaben sind M
ultiple-Choice oder
            Zuordnungs-Aufgaben.
            Nachdem Sie alle Aufgaben eines Assessments beantwortet haben, werden diese auf
gelöst und Sie können sich
            Ihre und die
            korrekten Antworten ansehen. Danach geht es zur nächsten Phase.
            <br><br>
            In Multiple Choice Aufgaben wird Ihnen eine Frage mit zugehörigen Antworten ang
ezeigt. Sie müssen die
            korrekten
            Antworten auswählen, wobei eine, mehrere oder alle Antworten richtig sein könne
n.
            <br><br>
            In Zuordnungs-
Aufgaben werden Ihnen mehrere Texte angezeigt. Neben jedem Text befindet sich ein Menü, in
dem
            Sie einen
            Begriff auswählen können. Sie müssen den angezeigten Texten ihre zugehörigen Be
griffe zuordnen. Ein Begriff
            kann auch
            mehrmals verwendet und daher mehreren Texten zugeordnet werden.
        </p>
        <h3>Lernen der Inhalte</h3>
        <p>Beim lernen der Inhalte werden Ihnen IT-
Sicherheitsrelevante Informationen, die auch unter “Inhalte”
            verfügbar sind,
            angezeigt. Sie können sich diese Inhalte ansehen und Übungsaufgaben machen. Übu
ngsaufgaben sind Aufgaben wie
            in den
            Assessments, die aber nach beantwortung sofort korrekt aufgelöst werden.</p>
    </div>
</div>
<div *ngIf="controller.phase == TrainingsPhase.Beendet" class="section-
lighter introduction-right">
    <h2 class="title">Starten Sie ein Training:</h2>
    <div class="kapitel">
        <label for="kapitel">Kapitel auswählen:</label>
        <select name="kapitel" id="kapitel" #kapitelnummer>
            <option value="1">1. Computer und Information</option>
            <option value="2">2. Passwortsicherheit</option>
            <option value="3">3. Sicher im Internet</option>
            <option value="4">4. E-Mails und Spam</option>
            <option value="5">5. Vieren und Malware</option>
        </select>
```

```
    </div>
    <button class="secondary-
btn" (click)="controller.startTraining(kapitelnummer.value)">Training Starten</button>
</div>
<div *ngIf="controller.phase != TrainingsPhase.Beendet" class="section-darker phase-guide">
    <h2 class="title-sm">Trainingsphasen:</h2>
    <div class="phasen-box">
        <div class="phasen-item"
            [ngClass]="{'active': controller.phase == TrainingsPhase.Vortraining || control
ler.phase == TrainingsPhase.VortrainingFeedback}">
            <span>Vortrainings-Assessment</span>
        </div>
        <div class="phasen-item"
            [ngClass]="{'active': controller.phase == TrainingsPhase.Inhalte || controller.
phase == TrainingsPhase.Ueben}">
            <span>Training</span>
        </div>
        <div class="phasen-item"
            [ngClass]="{'active': controller.phase == TrainingsPhase.Nachtraining || contro
ller.phase == TrainingsPhase.NachtrainingFeedback}">
            <span>Nachtrainings-Assessment</span>
        </div>
        <div class="phasen-
item" [ngClass]="{'active': controller.phase == TrainingsPhase.Empfehlung}">
            <span>Empfehlung</span>
        </div>
    </div>

</div>
<div *ngIf="controller.phase == TrainingsPhase.Empfehlung && !controller.showAssessmentScre
en" class="section-lighter content-container">
    <app-training-result
        *ngIf="controller.trainingResult !== undefined"
        [training]="controller.trainingResult"
        [btnDisabled]="false"
        (szenarioNavClicked)="controller.empfehlungSzenarioNavClicked($event)"
        class="content-box">
    </app-training-result>
    <div *ngIf="controller.trainingResult !== undefined" class="content-nav">
        <button class="secondary-btn" (click)="controller.beenden()">Fertig</button>
    </div>
</div>
<div class="section-lighter content-container inhalte-
box " [ngClass]="{'hidden': controller.phase != TrainingsPhase.Inhalte}">
    <div class="content-box">
        <app-kapitel1 *ngIf="controller.kapitelnummer === 1" ></app-kapitel1>
        <app-kapitel2 *ngIf="controller.kapitelnummer === 2" ></app-kapitel2>
        <app-kapitel3 *ngIf="controller.kapitelnummer === 3"></app-kapitel3>
        <app-kapitel4 *ngIf="controller.kapitelnummer === 4"></app-kapitel4>
        <app-kapitel5 *ngIf="controller.kapitelnummer === 5"></app-kapitel5>
    </div>
    <div class="content-nav">
        <button class="primary-dark-
btn" (click)="controller.initAssessment(TrainingsPhase.Nachtraining)">zur nächsten Training
sphase</button>
        <button class="primary-
btn" (click)="controller.initUebung()">Zu den Übungsaufgaben</button>
    </div>
</div>
<div *ngIf="controller.showAssessmentScreen" class="section-lighter content-container">
    <app-assessment [title]="controller.phase"
```

```
        [confirmBtnText]="controller.assessmentConfirmBtnText"
        [exitBtnText]="controller.assessmentExitBtnText"
        [szenario]="controller.currentSzenario"
        [mcAntwortSelected]="controller.mcAntwortSelected"
        [zuordnungsAuswahl]="controller.zuordnungsAuswahl"
        [currentSzenarioIndex]="controller.training.aktuellesSzenario"
        [szenarioFortschritt]="controller.szenarioFortschritt"
        [szenarioResults]="controller.szenarioResults"
        [feedBackMode]="controller.feedBackMode"

        (confirmBtn)="controller.confirmBtnClicked($event)"
        (exitBtn)="controller.exitBtnClicked($event)"
        (szenarioNav)="controller.szenarioNavClicked($event)">

    </app-assessment>
</div>
```

9.11.5.2 training-controller.service.ts

```
import { Injectable } from '@angular/core';
import { AggregatedTrainingResult } from '../model/AggregatedTrainingsResult';
import { ApiSzenarioAntwort } from '../model/api/ApiSzenarioAntwort';
import { AssessmentNavigation } from '../model/assessmentNavigation';
import { MCFrage } from '../model/szenarien/MCFrage';
import { MCFrageBeantwortet } from '../model/szenarien/MCFrageBeantwortet';
import { Zuordnung } from '../model/szenarien/Zuordnung';
import { ZuordnungBeantwortet } from '../model/szenarien/ZuordnungBeantwortet';
import { Training } from '../model/Training';
import { TrainingsPhase } from '../pages/training/TrainingsPhase';
import { AssessmentApiClientService } from './assessment-api-client.service';
import { ModelConverterService } from './model-converter.service';
import { TypeCheckService } from './type-check.service';



/**
 * Controller for a training. Built as a service to utilise lazy loading
 */
@Injectable({
  providedIn: 'root'
})
export class TrainingControllerService {
  /**
   * current of the 8 trainings pases
   */
  phase: TrainingsPhase = TrainingsPhase.Beendet;
  training: Training = new Training(-1, [], []);
  kapitelnummer: number = 0;
  showAssessmentScreen: boolean = false;

  assessmentConfirmBtnText: string = "";
  assessmentExitBtnText: string = "";

  /**
   * currently loaded szenario
   */
  currentSzenario?: MCFrage | Zuordnung;

  /**
   * user selected answers if the current szenario is MCFrage
```

```
   */
  mcAntwortSelected: Set<number> = new Set();

  /**
   * user selected answers if current szenario is Zuordnung
   */
  zuordnungsAuswahl: Map<number, string> = new Map();

  /**
   * list of szenarien with an user answer, (true: is answered, false: is not answered)
   * index refers to index of the Training.vortrainingSzenarien or
   * Training.nachtrainingSzenarien depending on the phase
   * array, where the szenarioID is stored
   */
  szenarioFortschritt?: boolean[];

  /**
   * list of szenario results for the feedback phase. true is correctly answered false is i
ncorrect
   * index refers to index of the Training.vortrainingSzenarien or
   * Training.nachtrainingSzenarien depending on the phase
   * array, where the szenarioID is stored
   */
  szenarioResults?: boolean[];
  feedBackMode: boolean = false;

  /**
   * Assessmentresult object with the trainings result
   * bound to the Input at the training result component
   */
  trainingResult?: AggregatedTrainingResult;

  constructor(
    private apiClient: AssessmentApiClientService,
    private typeCheck: TypeCheckService,
    private modelConverter: ModelConverterService
  ) { }

  /**
   * initiate a new training with the first trainings phase
   */
  startTraining(kapitelnummer: string) {
    if (this.phase == TrainingsPhase.Beendet) {
      this.kapitelnummer = Number(kapitelnummer);

      //call api and init training
      this.apiClient.startTraining(this.kapitelnummer).subscribe(assessment => {
        this.training = new Training(assessment.assessmentID, assessment.vortraining, asses
sment.nachtraining);
        console.log("training successfully started: " + JSON.stringify(assessment));
        this.initAssessment(TrainingsPhase.Vortraining);
      });
    }
  }

  /**
   * submit the answers of both training assessments
   */
  private submitTraining() {
    let vortrainingResult = this.modelConverter
```

```
      .convertAssessmentToResults(this.training.vortrainingSzenarien, this.training.vortrai
ningBeantwortet);
    let nachtrainingResult = this.modelConverter
      .convertAssessmentToResults(this.training.nachtrainingSzenarien, this.training.nachtr
ainingBeantwortet);

    this.apiClient.submitTraining(this.training.assessmentID, vortrainingResult, nachtraini
ngResult)
      .subscribe(() => console.log("training result sent to server"));

    this.apiClient.getKapitel(this.kapitelnummer).subscribe(kapitel => {
      this.trainingResult = {
        kapitelnummer: this.kapitelnummer,
        kapitelname: kapitel.titel,
        assessmentID: this.training.assessmentID,
        vortraining: this.aggregateTrainingsResults(vortrainingResult),
        nachtraining: this.aggregateTrainingsResults(nachtrainingResult)
      };
      console.log("Training result was set");
    });
  }

  /**
   * initialize all values for a vortrainings or nachtrainingsAssessment
   * @param phase Vortraining or Nachtraining
   */
  initAssessment(phase: TrainingsPhase) {
    this.currentSzenario = undefined;
    let amountSzenarien: number = 0;
    if (phase == TrainingsPhase.Vortraining) {
      amountSzenarien = this.training.vortrainingSzenarien.length;
    } else if (phase == TrainingsPhase.Nachtraining) {
      amountSzenarien = this.training.nachtrainingSzenarien.length;
    }

    this.showAssessmentScreen = true;
    this.assessmentConfirmBtnText = "Weiter";
    this.assessmentExitBtnText = "Abgeben";

    this.szenarioFortschritt = [];
    for (let i = 0; i < amountSzenarien; i++) {
      this.szenarioFortschritt.push(false);
    }

    this.szenarioResults = undefined;
    this.feedBackMode = false;
    this.training.aktuellesSzenario = 0;

    this.setAktuellesSzenario(phase);
    this.phase = phase;
    console.log(phase + " initialized");
  }

  /**
   * initialize all values for a vortrainings or nachtrainings feedback phase
   * @param phase Vortraining or Nachtraining
   */
  private initAssessmentFeedback(phase: TrainingsPhase) {
    this.showAssessmentScreen = true;
    this.assessmentConfirmBtnText = "Weiter";
    this.assessmentExitBtnText = "zur nächsten Trainingsphase";
```

```
  this.szenarioFortschritt = undefined;
  this.szenarioResults = this.createSzenarioResult(phase);
  this.feedBackMode = true;
  this.training.aktuellesSzenario = 0;

  this.setAktuellesSzenario(phase);
  this.phase = phase;

  console.log(phase + " feedback initialized");
}

private initInhalte() {
  this.showAssessmentScreen = false;
  this.phase = TrainingsPhase.Inhalte;
}

/**
 * initialize all values for a uebungsszenario
 */
initUebung() {
  this.showAssessmentScreen = true;
  this.assessmentConfirmBtnText = "Weiter";
  this.assessmentExitBtnText = "zu den Inhalten";
  this.szenarioFortschritt = undefined;
  this.szenarioResults = undefined;

  if (this.currentSzenario === undefined) {
    this.setNextUebungsSzenario();
  }
  this.phase = TrainingsPhase.Ueben;
}

private initEmpfehlung() {
  this.submitTraining();
  this.szenarioFortschritt = undefined;
  this.szenarioResults = undefined;
  this.assessmentConfirmBtnText = "Zurück";
  this.assessmentExitBtnText = "";
  this.feedBackMode = true;
  this.currentSzenario = undefined;
  this.showAssessmentScreen = false;
  this.phase = TrainingsPhase.Empfehlung;
}

/**
 * reset all values so a new training can be started
 */
beenden(){
  this.phase = TrainingsPhase.Beendet;
  this.assessmentConfirmBtnText = "";
  this.assessmentExitBtnText = "";
  this.showAssessmentScreen = false;

  this.currentSzenario = undefined;
  this.mcAntwortSelected = new Set();
  this.zuordnungsAuswahl = new Map();
  this.szenarioFortschritt = undefined;
  this.szenarioResults = undefined;
  this.feedBackMode = false;
  this.trainingResult = undefined;
```

```
  }

  /**
   * create a array with results for  the assessment of the given phase
   * values are true if the szeanrio at this index is correctly answered and false otherwis
e
   * indices refer to the vortrainingSzenarien or nachtrainingSzenarien array of the Traini
ng object, that contain the szenarioID
   *
   */
  private createSzenarioResult(phase: TrainingsPhase): boolean[] {
    let results: ApiSzenarioAntwort[];
    let enthalteneSzenarien: number[];
    if (phase == TrainingsPhase.VortrainingFeedback) {
      enthalteneSzenarien = this.training.vortrainingSzenarien;
      results = this.modelConverter.convertAssessmentToResults(enthalteneSzenarien, this.tr
aining.vortrainingBeantwortet);

    } else if (phase == TrainingsPhase.NachtrainingFeedback) {
      enthalteneSzenarien = this.training.nachtrainingSzenarien;
      results = this.modelConverter.convertAssessmentToResults(enthalteneSzenarien, this.tr
aining.nachtrainingBeantwortet);

    } else {
      throw new Error("invalid  phase in creating results");
    }

    let szenarioResults: boolean[] = [];
    for (let szenarioID of enthalteneSzenarien) {
      for (let result of results) {
        if (result.szenarioID === szenarioID) {
          szenarioResults.push(result.korrekt);
        }
      }
    }
    if (szenarioResults.length !== results.length) {
      throw new Error("result and szenarios didn't match");
    }
    return szenarioResults;
  }

  confirmBtnClicked(antworten: Map<number, string> | Set<number>) {
    switch (this.phase) {
      case TrainingsPhase.Vortraining: this.assessmentConfirmed(antworten); break;
      case TrainingsPhase.Nachtraining: this.assessmentConfirmed(antworten); break;
      case TrainingsPhase.VortrainingFeedback: this.feedbackConfirmed(); break;
      case TrainingsPhase.NachtrainingFeedback: this.feedbackConfirmed(); break;
      case TrainingsPhase.Ueben: this.uebungConfirmed(); break;
      case TrainingsPhase.Empfehlung: this.backToEmpfehlung(); break;
    }
  }

  exitBtnClicked(antworten: Map<number, string> | Set<number>) {
    switch (this.phase) {
      case TrainingsPhase.Vortraining: this.assessmentExited(antworten); break;
      case TrainingsPhase.Nachtraining: this.assessmentExited(antworten); break;
      case TrainingsPhase.VortrainingFeedback: this.feedbackExited(); break;
      case TrainingsPhase.NachtrainingFeedback: this.feedbackExited(); break;
      case TrainingsPhase.Ueben: this.uebenExited(); break;
    }
  }
```

```
  szenarioNavClicked(target: AssessmentNavigation) {
    switch (this.phase) {
      case TrainingsPhase.Vortraining: this.assessmentSzenarioNavigation(target); break;
      case TrainingsPhase.Nachtraining: this.assessmentSzenarioNavigation(target); break;
      case TrainingsPhase.VortrainingFeedback: this.feedbackSzenarioNavigation(target.nextS
zenarioIndex); break;
      case TrainingsPhase.NachtrainingFeedback: this.feedbackSzenarioNavigation(target.next
SzenarioIndex); break;
    }
  }

  /**
   * show the szenario with the given id on the assessment component
   */
  empfehlungSzenarioNavClicked(szenarioID:number) {
    let szenario = this.training.getNachtrainingSzenarioById(szenarioID);
    if (szenario === undefined ) {
      szenario = this.training.getVortrainingSzenarioById(szenarioID);
    }

    if (szenario !== undefined) {
      this.loadSzenarioWithAnswers(szenario);
      console.log("stored szenario was loaded");
      this.showAssessmentScreen = true;

    } else {
      this.apiClient.requestSzenario(szenarioID).subscribe(
        (szenario: MCFrage | Zuordnung) => {
          this.currentSzenario = szenario;
          this.mcAntwortSelected = new Set();
          this.zuordnungsAuswahl = new Map();
          console.log("szenario received: " + szenario.szenarioID);
          this.showAssessmentScreen = true;
        }
      );
    }
  }

  private backToEmpfehlung(){
    this.showAssessmentScreen = false;
    this.currentSzenario = undefined;
  }

  /**
   * confirm btn clicked in vor or nachtrainings assessment
   * @param antworten
   */
  private assessmentConfirmed(antworten: Map<number, string> | Set<number>) {
    this.storeSzenarioAnswer(antworten);
    this.training.aktuellesSzenario++;
    if (this.phase == TrainingsPhase.Vortraining && this.training.aktuellesSzenario === thi
s.training.vortrainingSzenarien.length) {
      this.initAssessmentFeedback(TrainingsPhase.VortrainingFeedback);

    } else if (this.phase == TrainingsPhase.Nachtraining && this.training.aktuellesSzenario
 === this.training.nachtrainingSzenarien.length) {
      this.initAssessmentFeedback(TrainingsPhase.NachtrainingFeedback);

    } else {
      this.setAktuellesSzenario(this.phase);
```

```
    }
  }

  private feedbackConfirmed() {
    if (this.phase == TrainingsPhase.VortrainingFeedback) {
      this.training.aktuellesSzenario = (this.training.aktuellesSzenario + 1) % this.traini
ng.vortrainingSzenarien.length;

    } else if (this.phase == TrainingsPhase.NachtrainingFeedback) {
      this.training.aktuellesSzenario = (this.training.aktuellesSzenario + 1) % this.traini
ng.nachtrainingSzenarien.length;
    }
    this.setAktuellesSzenario(this.phase);
  }

  private uebungConfirmed() {
    if (!this.feedBackMode) {
      this.feedBackMode = true;
    } else {
      this.feedBackMode = false;
      this.setNextUebungsSzenario();
    }
  }

  private assessmentExited(antworten: Map<number, string> | Set<number>) {
    this.storeSzenarioAnswer(antworten);
    if (this.phase == TrainingsPhase.Vortraining) {
      this.initAssessmentFeedback(TrainingsPhase.VortrainingFeedback);
    } else if (this.phase == TrainingsPhase.Nachtraining) {
      this.initAssessmentFeedback(TrainingsPhase.NachtrainingFeedback);
    }
  }

  private feedbackExited() {
    if (this.phase == TrainingsPhase.VortrainingFeedback) {
      this.currentSzenario = undefined;
      this.feedBackMode = false;
      this.initInhalte();

    } else if (this.phase == TrainingsPhase.NachtrainingFeedback) {
      this.initEmpfehlung();
    }
  }

  private uebenExited() {
    this.initInhalte();
  }

  private assessmentSzenarioNavigation(target: AssessmentNavigation) {
    this.storeSzenarioAnswer(target.answer);
    this.training.aktuellesSzenario = target.nextSzenarioIndex;
    this.setAktuellesSzenario(this.phase);
  }

  private feedbackSzenarioNavigation(szenarioIndex: number) {
    this.training.aktuellesSzenario = szenarioIndex;
    this.setAktuellesSzenario(this.phase);
  }

  private setNextUebungsSzenario(requestCount?: number): void {
    console.log("requested uebungsszenario");
```

```
    this.apiClient.requestUebungsszenario(this.kapitelnummer).subscribe(
      (szenario: MCFrage | Zuordnung) => {
        //if the new scenario is the same as the last one, request another one but max 10 t
imes
        if (this.currentSzenario !== undefined && this.currentSzenario.szenarioID === szena
rio.szenarioID && requestCount !== 10) {
          this.setNextUebungsSzenario(requestCount === undefined ? 1 : requestCount + 1);
        } else {
          this.mcAntwortSelected = new Set();
          this.zuordnungsAuswahl = new Map();
          this.currentSzenario = szenario;
          console.log("uebungsszenario received: " + szenario.szenarioID);
        }
      }
    );
  }

  private setAktuellesSzenario(assessment: TrainingsPhase) {
    console.log("loading szenario");
    let storedSzenario: MCFrageBeantwortet | ZuordnungBeantwortet | undefined;
    let szenarioID: number;

    if (assessment == TrainingsPhase.Vortraining || assessment == TrainingsPhase.Vortrainin
gFeedback) {
      szenarioID = this.training.vortrainingSzenarien[this.training.aktuellesSzenario];
      storedSzenario = this.training.getVortrainingSzenarioById(szenarioID);


    } else if (assessment == TrainingsPhase.Nachtraining || assessment == TrainingsPhase.Na
chtrainingFeedback) {
      szenarioID = this.training.nachtrainingSzenarien[this.training.aktuellesSzenario];
      storedSzenario = this.training.getNachtrainingSzenarioById(szenarioID);

    } else {
      throw new Error("Wanted to set aktuelles szenario, but assessment type was invalid");
    }

    if (storedSzenario !== undefined) {
      this.loadSzenarioWithAnswers(storedSzenario);
      console.log("stored szenario was loaded");

    } else {
      this.apiClient.requestSzenario(szenarioID).subscribe(
        (szenario: MCFrage | Zuordnung) => {
          this.currentSzenario = szenario;
          this.mcAntwortSelected = new Set();
          this.zuordnungsAuswahl = new Map();
          console.log("szenario received: " + szenario.szenarioID);

        }
      );
    }
  }


  private loadSzenarioWithAnswers(szenario: MCFrageBeantwortet | ZuordnungBeantwortet) {
    this.currentSzenario = szenario;
    if (this.typeCheck.isSzenarioMCFrageBeantwortet(szenario)) {
      this.mcAntwortSelected = szenario.ausgewaehlt;
      this.zuordnungsAuswahl = new Map();
```

```
    } else {
      this.mcAntwortSelected = new Set();
      this.zuordnungsAuswahl = szenario.zugeordnet;
    }
  }

  private storeSzenarioAnswer(answer: Map<number, string> | Set<number>) {
    if (this.phase == TrainingsPhase.Vortraining) {
      if (answer instanceof Set && this.typeCheck.isSzenarioMCFrage(this.currentSzenario))
{
        this.training.vortrainingBeantwortet[this.training.aktuellesSzenario] = this.create
MCFrageBeantwortet(answer, this.currentSzenario);

      } else if (answer instanceof Map && this.typeCheck.isSzenarioZuordnung(this.currentSz
enario)) {
        this.training.vortrainingBeantwortet[this.training.aktuellesSzenario] = this.create
ZuordnungBeantwortet(answer, this.currentSzenario);
      }

      if (this.szenarioFortschritt !== undefined && answer.size !== 0) {
        console.log("store szenario answer, its size is: " + answer.size);

        this.szenarioFortschritt[this.training.aktuellesSzenario] = true;
      }
    } else if (this.phase == TrainingsPhase.Nachtraining) {
      if (answer instanceof Set && this.typeCheck.isSzenarioMCFrage(this.currentSzenario))
{
        this.training.nachtrainingBeantwortet[this.training.aktuellesSzenario] = this.creat
eMCFrageBeantwortet(answer, this.currentSzenario);

      } else if (answer instanceof Map && this.typeCheck.isSzenarioZuordnung(this.currentSz
enario)) {
        this.training.nachtrainingBeantwortet[this.training.aktuellesSzenario] = this.creat
eZuordnungBeantwortet(answer, this.currentSzenario);
      }

      if (this.szenarioFortschritt !== undefined && answer.size !== 0) {
        console.log("store szenario answer, its size is: " + answer.size);

        this.szenarioFortschritt[this.training.aktuellesSzenario] = true;
      }
    }
  }

  private createMCFrageBeantwortet(userAntwort: Set<number>, szenario: MCFrage): MCFrageBea
ntwortet {
    return {
      szenarioID: szenario.szenarioID,
      kapitelnummer: szenario.kapitelnummer,
      themanummer: szenario.themanummer,
      typ: szenario.typ,
      frage: szenario.frage,
      bild_url: szenario.bild_url,
      antworten: szenario.antworten,
      ausgewaehlt: userAntwort
    };
  }

  private createZuordnungBeantwortet(userAntwort: Map<number, string>, szenario: Zuordnung)
: ZuordnungBeantwortet {
```

```
    return {
      szenarioID: szenario.szenarioID,
      kapitelnummer: szenario.kapitelnummer,
      themanummer: szenario.themanummer,
      typ: szenario.typ,
      anweisung: szenario.anweisung,
      eintraege: szenario.eintraege,
      zugeordnet: userAntwort
    }
  }

  private aggregateTrainingsResults(assessmentResult:ApiSzenarioAntwort[]){
    let gesammtKorrekt = 0;
    assessmentResult.forEach(result=> {
      if(result.korrekt) {
        gesammtKorrekt++;
      }
    });
    return {
      prozent: Math.ceil((gesammtKorrekt / (assessmentResult.length)) * 100),
      korrekt: gesammtKorrekt,
      total: assessmentResult.length,
      antworten: assessmentResult
    }
  }

}
```

9.11.6 „fortschritt-page“ Component

9.11.6.1 fortschritt-page.component.html

```
<div class="section-darker">
    <div class="header">
        <h2 class="title">Fortschritt:</h2>
        <div class="typ-auswahl">
            <button (click)="trainingsAnsicht=true" [ngClass]="{'active': trainingsAnsicht}
"
                id="training-btn">Trainings</button>
            <button (click)="trainingsAnsicht=false" [ngClass]="{'active': !trainingsAnsich
t}"
                id="prüfung-btn">Prüfungen</button>
        </div>
    </div>
    <div class="scroll-container">
        <nav *ngIf="trainingsAnsicht" class="trainings">
            <div *ngFor="let training of trainingsStatistics" class="item active"
                [ngClass]="{'disabled': !training.vortraining.abgeschlossen, 'active': acti
veTraining !== undefined && activeTraining.assessmentID === training.vortraining.assessment
ID}"
                (click)="loadTrainingResult(training)">
                <div class="item-title">
                    <h4>{{timestampToDate(training.vortraining.zeitpunkt)}}</h4>
                    <h4>{{timestampToTime(training.vortraining.zeitpunkt)}}</h4>
                </div>
                <div *ngIf="training.vortraining.abgeschlossen" class="details">
                    <span class="assessment-
title">Vortraining: {{getPercent(training.vortraining.korrekt,
                        training.vortraining.inkorrekt)}}%</span>
```

```
                    <span class="assessment-
title">Nachtraining: {{getPercent(training.nachtraining.korrekt,
                        training.nachtraining.inkorrekt)}}%</span>
                </div>
                <div *ngIf="!training.vortraining.abgeschlossen" class="details">
                    <span>Nicht Abgeschlossen</span>
                </div>
            </div>
        </nav>

        <nav *ngIf="!trainingsAnsicht" class="prüfung">
            <div *ngFor="let pruefung of pruefungsStatistics" class="item"
                [ngClass]="{'disabled': !pruefung.abgeschlossen, 'active': activePruefung !
== undefined && activePruefung.assessmentID === pruefung.assessmentID}"
                (click)="loadPruefungsResult(pruefung)">
                <div class="item-title">
                    <h4>{{timestampToDate(pruefung.zeitpunkt)}}</h4>
                    <h4>{{timestampToTime(pruefung.zeitpunkt)}}</h4>
                </div>
                <div *ngIf="pruefung.abgeschlossen" class="details">
                    <span>Ergebnis: {{getPercent(pruefung.korrekt, pruefung.inkorrekt)}}%
                        ({{pruefung.korrekt}}/{{pruefung.korrekt + pruefung.inkorrekt}})</s
pan>
                </div>
                <div *ngIf="!pruefung.abgeschlossen" class="details">
                    <span>Nicht Abgeschlossen</span>
                </div>
            </div>
        </nav>
    </div>

</div>

<div class="section-lighter">
    <app-training-
result *ngIf="trainingsAnsicht && activeTraining != undefined" [training]="activeTraining">
    </app-training-result>
    <app-pruefung-
result *ngIf="!trainingsAnsicht && activePruefung != undefined" [pruefung]="activePruefung"
>
    </app-pruefung-result>
</div>
```

9.11.6.2 fortschritt-page.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AggregatedTrainingResult } from '../../../model/AggregatedTrainingsResult';
import { AggregatedTrainingStatistics } from '../../../model/AggregatedTrainingStatistics';
import { AssessmentResult } from '../../../model/api/AssessmentResult';
import { AssessmentStatistics } from '../../../model/api/AssessmentStatistics';
import { AppDaoService } from '../../../services/app-dao.service';
import { AssessmentApiClientService } from '../../../services/assessment-api-
client.service';

@Component({
  selector: 'app-fortschritt-page',
  templateUrl: './fortschritt-page.component.html',
  styleUrls: ['./fortschritt-page.component.sass']
})
```

```
export class FortschrittPageComponent implements OnInit {
  trainingsAnsicht: boolean = true
  pruefungsStatistics: AssessmentStatistics[] = [];
  trainingsStatistics: AggregatedTrainingStatistics[] = [];
  activePruefung?: AssessmentResult;
  activeTraining?: AggregatedTrainingResult;

  constructor(
    private router:Router,
    private appDao:AppDaoService,
    private apiClient: AssessmentApiClientService) { }

  ngOnInit(): void {
    if(!this.appDao.user.isLoggedIn()) {
      this.router.navigate(['/']);
    }
    this.loadAssessments();

  }

  loadAssessments() {
    if (this.pruefungsStatistics.length === 0) {

      this.apiClient.getPruefungsAssessmentStatistics(this.appDao.user.getBenutzername(), 1
0)
        .subscribe(statistics => {
          this.pruefungsStatistics = statistics;
          console.log("pruefungs statistik loaded");
        });
    }

    if (this.trainingsStatistics.length === 0) {
      this.apiClient.getTrainingsAssessmentStatistics(this.appDao.user.getBenutzername(), 2
0)
        .subscribe(statistics => {
          this.trainingsStatistics = this.aggregateTrainingsStatistics(statistics);
          console.log("trainings statistik loaded");
        });
    }
  }

  loadTrainingResult(statistics:AggregatedTrainingStatistics){
    if(statistics.vortraining.abgeschlossen && !(this.activeTraining !== undefined && this.
activeTraining.assessmentID === statistics.vortraining.assessmentID)){
      this.apiClient.getTrainingsAssessmentResults(this.appDao.user.getBenutzername(), stat
istics.vortraining.assessmentID)
        .subscribe(results => this.setAggregatedTrainingsResult(results, statistics));
    }
  }


  loadPruefungsResult(pruefung:AssessmentStatistics){
    this.activePruefung = undefined;
    if (pruefung.abgeschlossen) {
      console.log("requesting pruefungsresult of " + pruefung.assessmentID);
      this.apiClient.getPruefungsAssessmentResults(this.appDao.user.getBenutzername(), prue
fung.assessmentID)
        .subscribe(result => {
          this.activePruefung = result;
      });
    }
```

```
  }


  timestampToTime(timestamp: string): string {
    let time = timestamp.replace(/.*T/g, ' ').split(':');
    return time[0] + ":" + time[1];
  }

  timestampToDate(timestamp: string): string {
    let date = timestamp.replace(/T.*/, ' ').split('-');
    return date[2] + "." + date[1] + ". " + date[0];
  }

  getPercent(korrekt:number, inkorrekt:number):number {
    if(korrekt==0) {
      return 0;
    }
    return Math.ceil((korrekt / (korrekt+inkorrekt))*100);
  }


  aggregateTrainingsStatistics(combinedList:AssessmentStatistics[]):AggregatedTrainingStati
stics[] {
    let aggregated: AggregatedTrainingStatistics[] = [];
    for(let vortraining of combinedList) {
      if(vortraining.typ == "VORTRAINING") {
        for(let nachtraining of combinedList) {
          if(nachtraining.typ == "NACHTRAINING" && nachtraining.assessmentID === vortrainin
g.assessmentID) {
            aggregated.push({
              vortraining: vortraining,
              nachtraining:nachtraining
            });
          }
        }
      }
    }
    return aggregated;
  }

  setAggregatedTrainingsResult(results: AssessmentResult[], statistics: AggregatedTrainingS
tatistics) {
    if (results[0].typ != "VORTRAINING" || results[1].typ != "NACHTRAINING") {
      console.error("no valid assessment results received at loading training");
      return;
    }

    //kapitel nummer and name is needed, retrieved from server
    this.apiClient.requestSzenario(results[0].szenarien[0].szenarioID).subscribe(szenario=>
{
      this.apiClient.getKapitel(szenario.kapitelnummer).subscribe(kapitel=>{
        this.activeTraining = {
          kapitelnummer: kapitel.kapitelnummer,
          kapitelname: kapitel.titel,
          assessmentID: results[0].assessmentID,
          vortraining: {
            korrekt: statistics.vortraining.korrekt,
            total: statistics.vortraining.korrekt + statistics.vortraining.inkorrekt,
            prozent: this.getPercent(statistics.vortraining.korrekt, statistics.vortraining
.inkorrekt),
            antworten: results[0].szenarien
```

```
          },
          nachtraining: {
            korrekt: statistics.nachtraining.korrekt,
            total: statistics.nachtraining.korrekt + statistics.nachtraining.inkorrekt,
            prozent: this.getPercent(statistics.nachtraining.korrekt, statistics.nachtraini
ng.inkorrekt),
            antworten: results[1].szenarien
          }
        }
      });
    });
  }
}
```

9.11.7 „monitoring“ Component

9.11.7.1 monitoring.component.html

```
<div class="section-darker">
    <h2 class="title">Personal:</h2>
    <div class="scroll-container">
        <nav>
            <div *ngFor="let user of allUsers" [ngClass]="{'active': activeUser == user.ben
utzername}"
                (click)="loadPruefungen(user.benutzername)" class="item active">
                <div class="item-title">
                    <h4>{{user.vorname}} {{user.nachname}}</h4>
                </div>
                <div class="details">
                    <span>Letzte Prüfung:</span>
                    <b>{{user.letztePruefung}}</b>
                </div>
            </div>
        </nav>
    </div>

</div>
<div class="section-dark">
    <h2 class="title">Prüfungen:</h2>
    <div class="scroll-container">
        <nav>
            <div *ngFor="let pruefung of pruefungsStatistics"
                [ngClass]="{'disabled': !pruefung.abgeschlossen, 'active': activePruefung !
== undefined && activePruefung.assessmentID === pruefung.assessmentID}"
                (click)="loadPruefungsResult(pruefung)" class="item">
                <div class="item-title">
                    <h4>{{timestampToDate(pruefung.zeitpunkt)}}</h4>
                    <h4>{{timestampToTime(pruefung.zeitpunkt)}}</h4>
                </div>
                <div class="details">
                    <span *ngIf="pruefung.abgeschlossen">
                        Ergebnis: {{getPercent(pruefung.korrekt, pruefung.inkorrekt)}}%
                        ({{pruefung.korrekt}}/{{pruefung.korrekt + pruefung.inkorrekt}})
                    </span>
                    <span *ngIf="!pruefung.abgeschlossen">Nicht Abgeschlossen</span>
                </div>
            </div>
        </nav>
    </div>
```

```
</div>
<div class="section-lighter">
    <app-pruefung-result *ngIf="activePruefung !== undefined" [pruefung]="activePruefung">
    </app-pruefung-result>
</div>
```

9.11.7.2 monitoring.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AssessmentResult } from '../../model/api/AssessmentResult';
import { AssessmentStatistics } from '../../model/api/AssessmentStatistics';
import { AppDaoService } from '../../services/app-dao.service';
import { AssessmentApiClientService } from '../../services/assessment-api-client.service';

@Component({
  selector: 'app-monitoring',
  templateUrl: './monitoring.component.html',
  styleUrls: ['./monitoring.component.sass']
})
export class MonitoringComponent implements OnInit {
  allUsers: any[] = [];
  activeUser: string = "";

  pruefungsStatistics: AssessmentStatistics[] = [];
  activePruefung?: AssessmentResult;

  constructor(
    private router: Router,
    private appDao: AppDaoService,
    private apiClient: AssessmentApiClientService
  ) { }

  ngOnInit(): void {
    if (!this.appDao.user.isLoggedIn()) {
      this.router.navigate(['/']);
    }
    this.loadAllUsers();
  }

  loadAllUsers() {
    this.activePruefung = undefined;
    this.activeUser = "";
    this.allUsers = [];
    this.apiClient.getUsers().subscribe(apiUsers => {
      this.allUsers = apiUsers;
      for (let i = 0; i < this.allUsers.length; i++) {
        this.apiClient.getPruefungsAssessmentStatistics(this.allUsers[i].benutzername, 1)
          .subscribe(pruefung =>
            this.allUsers[i].letztePruefung = pruefung[0] ? this.timestampToDate(pruefung[0
].zeitpunkt) : ''
          );
      }
    });
  }

  loadPruefungen(benutzername: string) {
    this.activePruefung = undefined;
    this.activeUser = benutzername;
    console.log("requesting pruefungen");
```

```
    this.apiClient.getPruefungsAssessmentStatistics(benutzername, 10)
      .subscribe(statistics => {
        this.pruefungsStatistics = statistics;
        console.log("pruefungs statistik loaded");
      });
  }

  loadPruefungsResult(pruefung: AssessmentStatistics) {
    this.activePruefung = undefined;
    if (pruefung.abgeschlossen) {
      console.log("requesting pruefungsresult of " + pruefung.assessmentID);
      this.apiClient.getPruefungsAssessmentResults(this.activeUser, pruefung.assessmentID)
        .subscribe(result => {
          this.activePruefung = result;
        });
    }


  }


  timestampToTime(timestamp: string): string {
    let time = timestamp.replace(/.*T/g, ' ').split(':');
    return time[0]+":"+time[1];
  }

  timestampToDate(timestamp: string): string {
    let date = timestamp.replace(/T.*/, ' ').split('-');
    return date[2]+"."+date[1] + ". "+ date[0];
  }

  getPercent(korrekt: number, inkorrekt: number): number {
    if (korrekt == 0) {
      return 0;
    }
    return Math.ceil((korrekt / (korrekt + inkorrekt)) * 100);
  }

}
```