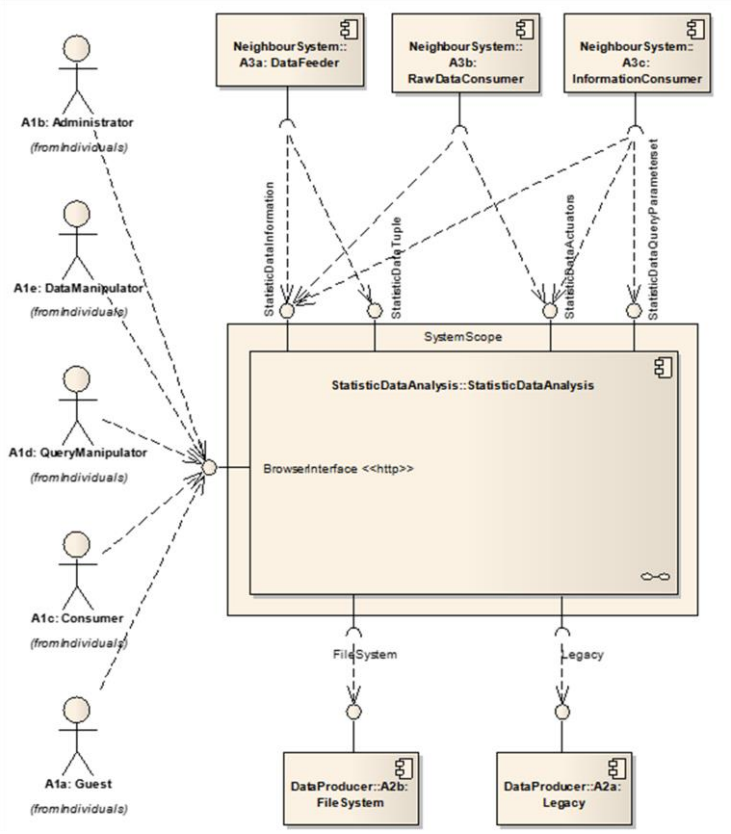


Statistik-Daten Auswertung

Masterarbeit HSR-MAS-SE 2016

Anforderungsspezifikation

Version:	006
Status:	RELEASE
Release Date:	26.08.2018
Classification:	PUBLIC

Document-ID: n/a
anforderungsspezifikation_public.docx

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Zu diesem Dokument

1.1 Zweck

Die Anforderungsspezifikation zeigt die Abgrenzung zwischen der „Statistik-Daten Auswertung“ und deren Umsysteme, deren Anwendungsfälle, die logischen Datenstrukturen der Systemschnittstellen, die nicht-funktionellen Anforderungen und den Kern der für die Funktionserfüllung notwendigen Information und Abläufe.

1.2 Gültigkeit

Während der Inception Phase des Projekts „Statistik-Daten Auswertung“ wird eine erste Stufe der Anforderungsspezifikation ausgearbeitet. Diese Stufe hält den Focus auf das geplante «Minimal Viable Product».

In der Elaborations-Phase wird diese Anforderungsspezifikation sukzessive erweitert, so dass schlussendlich alle Anforderungen für das Endprodukt abgedeckt sind.

Das Dokument wurde gemäss den Richtlinien [5] am 14.05.2018 freigeben. Inhaltlich werden keine weiteren Änderungen am Dokument mehr vorgenommen. Änderungen betreffen nur die Form und Gliederung, sowie das Aufdatieren von Referenzen. Sämtliche Änderungen der Anforderungen sind ab diesem Zeitpunkt in der Architekturdokumentation [4] festgehalten.

1.3 Verfügbarkeit des Dokumentes

Dieses Dokument befindet sich im Projektverzeichnis: 10_Anforderungsspezifikation

2 Weiterführende Informationen

2.1 Definitionen, Akronyme und Abkürzungen

Sämtliche Definitionen, Akronyme und Abkürzungen werden zentral in einem Projektglossar dokumentiert. Das Projektglossar wird kontinuierlich angepasst.

Das Projektglossar befindet sich im Projektverzeichnis: 09_Glossar

2.2 Ergänzende Dokumente

Ref.	Beschreibung
[1]	Projektantrag: «Statistik und Event-Log Auswertung», VER011
[2]	Projektleitdokumentation: «Statistik-Daten Auswertung, Projektleitdokumentation», VER005
[3]	Architekturdokumentation: «Statistik-Daten Auswertung, Architekturdokumentation», VER006
[4]	Lehrbuch der Objektmodellierung - Analyse und Entwurf mit der UML2, 2. Auflage, Heide Balzert ISBN: 978-3-8274-2903-2

2.3 Prozessbezogene Dokumente

Ref.	Beschreibung
[5]	Masterarbeit MAS-SE 2016, Richtlinie MAS-SE

2.4 Versionshistorie

(nicht öffentlich)

2.5 Verteilung

Ver.	Name	Rolle / Titel / Link
006	MAS-Arbeit	MAS-Arbeit

3 Einführung und Projektziel

3.1 Vision / Ziel

Die bestehende Client-Datenbank-Architektur zur Speicherung von Aktuatorkenngrössen (Statistik-Daten) soll durch eine flexiblere Architektur ersetzt werden. Für die Datenanalyse sollen die Statistik-Daten aufbereitet und rollenspezifisch zur Verfügung gestellt werden. Das Abfragen und Analysieren der Daten soll für jeden Akteur ohne fremdes Zutun möglich sein. Bisher waren Akteure oft auf die Hilfe von Personen mit Datenbank-Fachwissen angewiesen.

3.2 Stakeholder

(nicht öffentlich)

4 Systemabgrenzung

4.1 Systemkontext

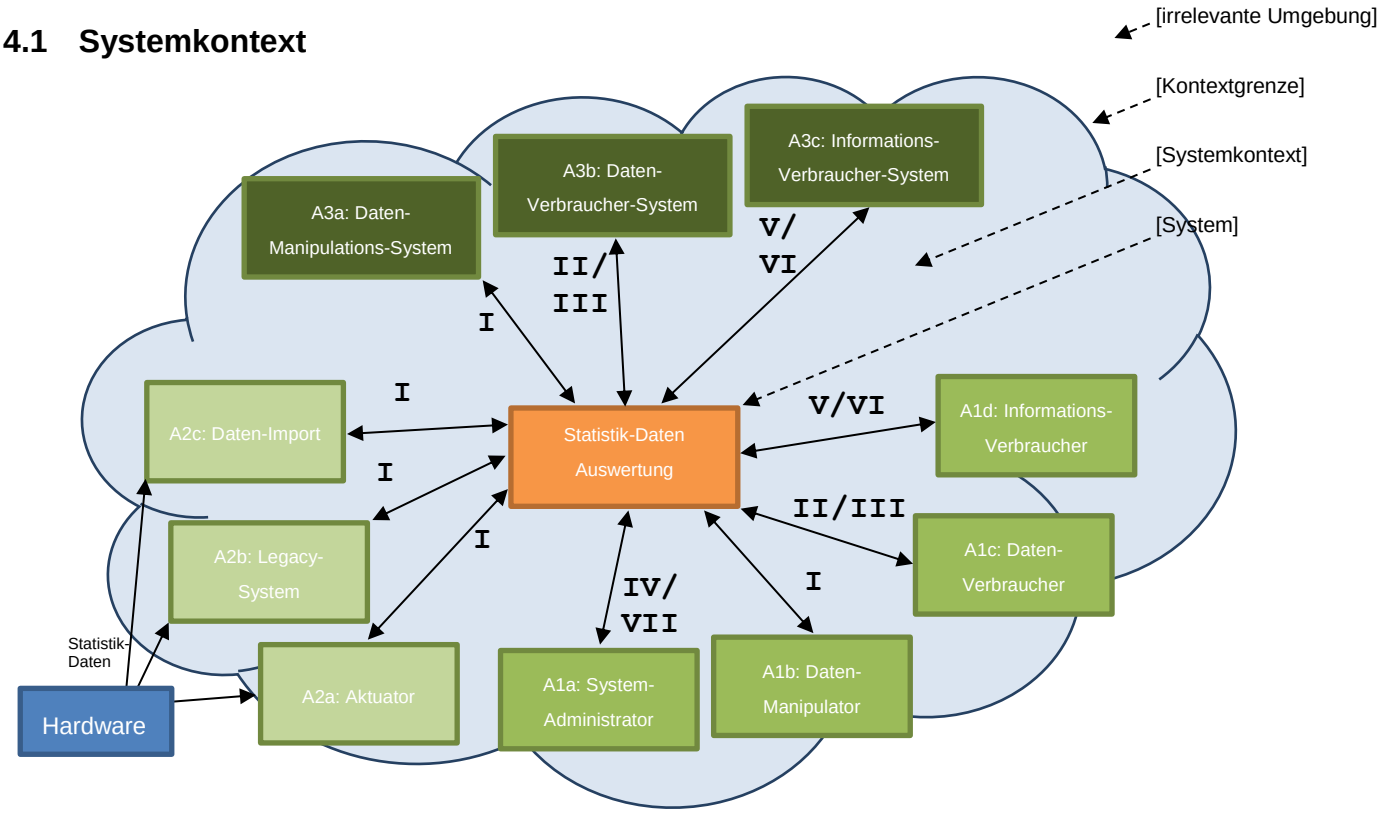


Abbildung 1: Kontextdiagramm

Aktuatoren speichern im Betrieb diverse Kenngrössen (z.B. Anzahl un-/berechtigte Medien). Diese Statistik-Daten sollen mittels Adapter ausgelesen und dem «Statistik-Daten Auswertung»-System übergeben werden. Zudem könnten Statistik-Daten auch von übergeordneten Hostsystemen kommen oder aber von einem Mitarbeiter importiert und/oder editiert werden.

Es soll die Möglichkeit bestehen, dass verschiedene Akteure die Daten selektiv abfragen können. Zudem soll das System über eine Report-Funktion verfügen, die es erlaubt, eine automatische Ist-Soll-Analyse der Daten durchzuführen.

Das Abfragen und Analysieren der Daten soll für jeden Akteur ohne fremdes Zutun möglich sein. Das neue System muss daher die Rollen der verschiedenen Akteure unterstützen.

4.1.1 Zentrale Datenstrukturen (vgl. Kap: 7)

	System empfängt	**System versendet**
I	Statistik-Daten-Tupel vgl. 7.2.1	Statistik-Daten-Tupel-Ack vgl. 7.2.2
II	Statistik-Daten-Selektions-Parameterset vgl. 7.2.3	Statistik-Daten-Selektion vgl. 7.2.4
III	Statistik-Daten-Selektions-Parameterset-Req vgl. 7.2.5	Statistik-Daten-Selektions-Parameterset vgl. 7.2.3
IV	Statistik-Daten-Selektions-Parameterset vgl. 7.2.3	Statistik-Daten-Selektions-Parameterset-Ack vgl. 7.2.6
V	Statistik-Daten-Report-Parameterset vgl. 7.2.7	Statistik-Daten-Report vgl. 7.2.8
VI	Statistik-Daten-Report-Parameterset-Req vgl. 7.2.9	Statistik-Daten-Report-Parameterset vgl. 7.2.7
VII	Statistik-Daten-Report-Parameterset vgl. 7.2.7	Statistik-Daten-Report-Parameterset-Ack vgl. 7.2.10

5 Akteure

5.1 A1: Natürliche Person

5.1.1 A1a: System-Administrator

Der System-Administrator verwaltet und konfiguriert das System. Da heisst, er erstellt Statistik-Daten-Selektions-Parametersets und Statistik-Daten-Report-Parametersets.

5.1.2 A1b: Daten-Manipulator

Der Daten-Manipulator verändert und löscht einzelne Statistik-Daten-Tupel.

5.1.3 A1c: Daten-Verbraucher

Der Daten-Verbraucher konsumiert Statistik-Daten-Selektionen.

5.1.4 A1d: Informations-Verbraucher

Der Informations-Verbraucher konsumiert Statistik-Daten-Reports.

5.1.5 Rollenzuteilung

Stakeholder	System-Administrator	Daten-Manipulator	Daten-Verbraucher	Informations-Verbraucher
Entwickler (S3, S4)	X	X	X	
Supporter (S5)	X		X	
Test-Verifikation-Validation-Team (S6)	X		X	X
Projektleiter (S7)				X
Feldtest-Manager (S6)	X	X	X	X

5.2 A2: Datenquellen

Statistik-Daten werden während der Laufzeit eines Aktuators erzeugt und intern abgespeichert. Diese Statistik-Daten können auf unterschiedliche Weise ausgelesen werden. Nach einer automatischen oder manuellen Aufbereitung, können die Daten als Statistik-Daten-Tupel dem «Statistik-Daten Auswertungs»-System übergeben werden.

5.2.1 A2a: Aktuator

Der Aktuator liefert zu einem bestimmen Zeitpunkt ein bestimmtes Statistik-Daten-Tupel.

5.2.2 A2b: Legacy-System

Das Legacy-System liefert sämtlich bisher erfassten Statistik-Daten-Tupels. (Migration)

5.2.3 A2c: Daten-Import

Für Testzwecke sollen Statistik-Daten-Tupel via Dateisystem importiert werden können.

5.3 A3: Verwaltungs-/Hostsysteme

(nicht öffentlich)

5.3.1 A3a: Daten-Manipulations-System

Ein Daten-Manipulations-System erzeugt und verändert einzelne Statistik-Daten-Tupel. Das «Statistik-Daten Auswertungs»-System kennt ein allfälliges Daten-Manipulations-System nicht. Es wird eine Schnittstelle angeboten, die durch ein allfälliges Daten-Manipulations-System angesprochen werden kann.

5.3.2 A3b: Daten-Verbraucher-System

Ein Daten-Verbraucher-System konsumiert Statistik-Daten-Selektionen. Das «Statistik-Daten Auswertungs»-System kennt ein allfälliges Daten-Verbraucher-System nicht. Es wird eine Schnittstelle angeboten, die durch ein allfälliges Daten-Verbraucher-System angesprochen werden kann.

5.3.3 A3c: Informations-Verbraucher-System

Ein Informations-Verbraucher-System konsumiert Statistik-Daten-Reports. Das «Statistik-Daten Auswertungs»-System kennt ein allfälliges Informations-Verbraucher-System nicht. Es wird eine Schnittstelle angeboten, die durch ein allfälliges Informations-Verbraucher-System angesprochen werden kann.

6 Anwendungsfälle

Die im folgenden aufgeführten Anwendungsfälle beruhen darauf, dass die Infrastruktur bereits alle fehlerhaften Eingaben, welche in den physischen Schnittstellen entstehen können, bereinigt hat. Somit beschreiben die Anwendungsfälle die Abläufe und Regeln der Administration und natürlich deren Kern.

Anwendungsfälle sind aus Sicht des Systems formuliert, d.h.: Das System nimmt Daten entgegen und gibt bekannt. Der Akteur startet eine Aktion z.B. mit dem Aufruf von «entgegennehmen()» oder «bekannt geben()».

6.1 Übersicht

Das folgende Diagramm ist eine Übersicht der essentiellen Anwendungsfälle.

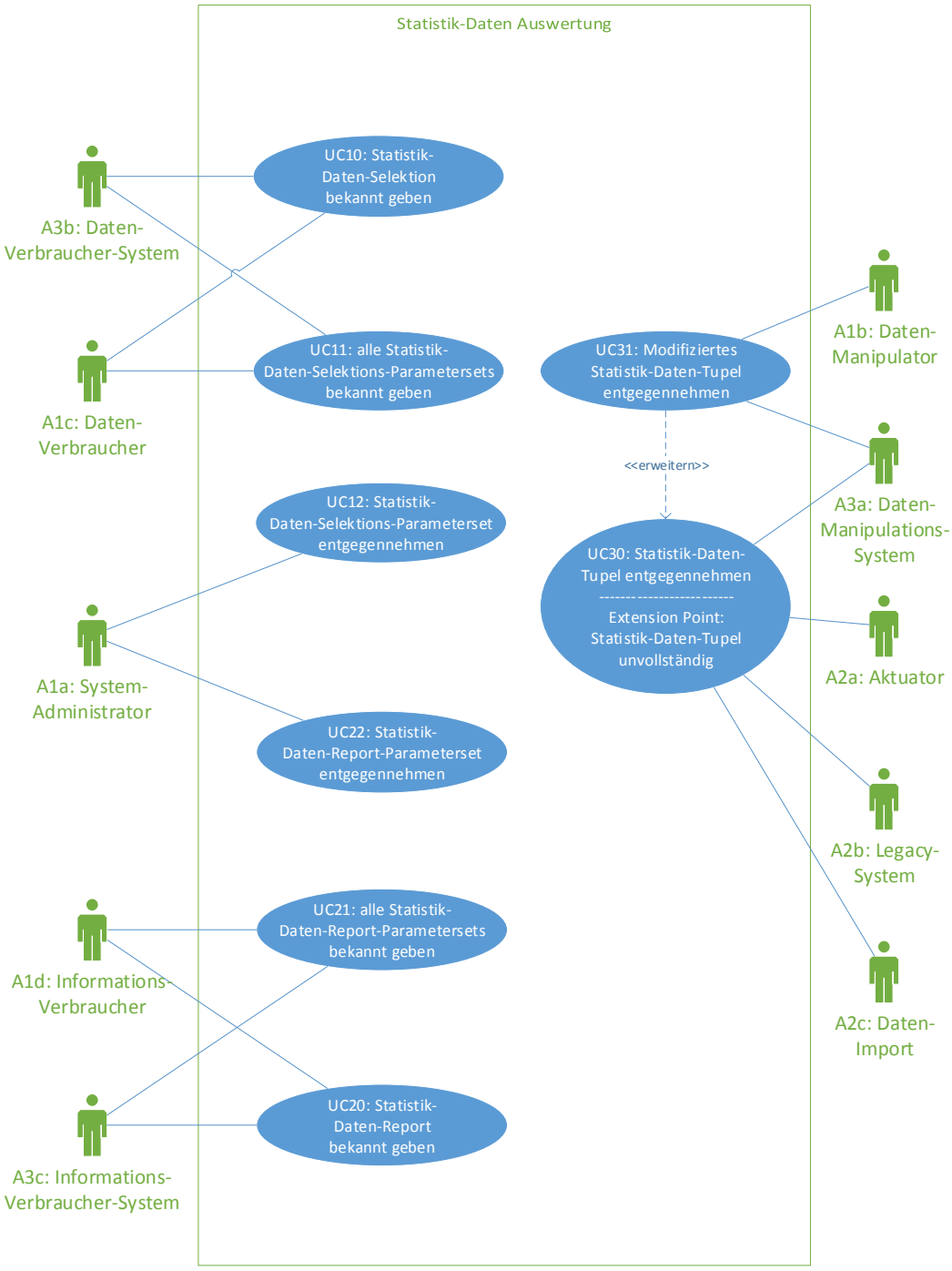


Abbildung 2: Übersicht Anwendungsfälle

6.2 UC10: Statistik-Daten-Selektion bekannt geben

6.2.1 Beschreibung

Abschnitt	Inhalt
Bezeichner	UC10
Anwendungsfallname	Statistik-Daten-Selektion bekannt geben
Autoren	Projektleitung
Prioritäten	Hoch
Kritikalität	Mittel
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	Auf Anfrage gibt das System dem Akteur die gewünschte Statistik-Daten-Selektion bekannt.
Auslösendes Ereignis	Akteur setzt mittels Statistik-Daten-Selektions-Parameterset eine Anfrage ab.
Akteure	A1c: Daten-Verbraucher A3b: Daten-Verbraucher-System
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Selektions-Parameterset vgl. 7.2.3
Datenstruktur (Antwort): System → Akteur	Statistik-Daten-Selektion vgl. 7.2.4
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt gewünschte Statistik-Daten-Selektion entgegen.
Hauptszenario	1) Anfrage zur Bekanntgabe einer Statistik-Daten-Selektion durch den Akteur. 2) System verwendet das gesendete Statistik-Daten-Selektions-Parameterset zur Erstellung der gewünschten Statistik-Daten-Selektion. 3) System gibt die Statistik-Daten-Selektion dem Akteur bekannt.
Alternativszenarien	-
Ausnahmeszenarien	1) Anfrage zur Bekanntgabe einer Statistik-Daten-Selektion durch den Akteur. 2) System verwendet das gesendete Statistik-Daten-Selektions-Parameterset zur Erstellung der gewünschten Statistik-Daten-Selektion. 3) Fehlerfall! 4) System gibt dem Akteur bekannt, dass das Statistik-Daten-Selektions-Parameterset nicht verarbeitet werden konnte.
Fehlerfall	• Verbindungsfehler können detektiert und dem Benutzer bekannt gegeben werden. • Ein fehlendes Statistik-Daten-Selektions-Parameterset bei der Anfrage kann detektiert und dem Benutzer bekannt gegeben werden. • Ein nicht verarbeitbares Statistik-Daten-Selektions-Parameterset kann detektiert und dem Benutzer bekannt gegeben werden.

6.2.2 Ablauf

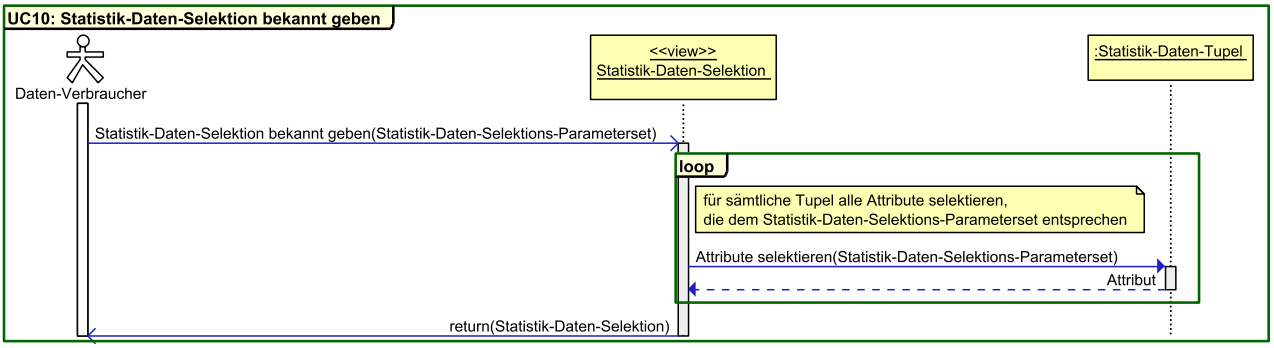


Abbildung 3: UC10 - Statistik-Daten-Selektion bekannt geben

6.3 UC11: Alle Statistik-Daten-Selektions-Parametersets bekannt geben

6.3.1 Beschreibung

Abschnitt	Inhalt
Bezeichner	UC11
Anwendungsfallname	Alle Statistik-Daten-Selektions-Parametersets bekannt geben
Autoren	Projektleitung
Prioritäten	Mittel
Kritikalität	Mittel
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	Auf Anfrage gibt das System dem Akteur sämtliche Statistik-Daten-Selektions-Parametersets bekannt.
Auslösendes Ereignis	Akteur setzt mittels Statistik-Daten-Selektions-Parameterset-Req eine Anfrage ab.
Akteure	A1c: Daten-Verbraucher A3b: Daten-Verbraucher-System
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Selektions-Parameterset-Req vgl. 7.2.5
Datenstruktur (Antwort): System → Akteur	Statistik-Daten-Selektions-Parameterset vgl. 7.2.3
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt sämtliche Statistik-Daten-Selektions-Parametersets entgegen.
Hauptszenario	1) Anfrage zur Bekanntgabe sämtlicher Statistik-Daten-Selektions-Parametersets durch den Akteur. 2) System gibt sämtliche Statistik-Daten-Selektions-Parametersets dem Akteur bekannt.

Alternativszenarien	-
Ausnahmeszenarien	-
Fehlerfall	• Verbindungsfehler können detektiert und dem Benutzer bekannt gegeben werden.

6.3.2 Ablauf

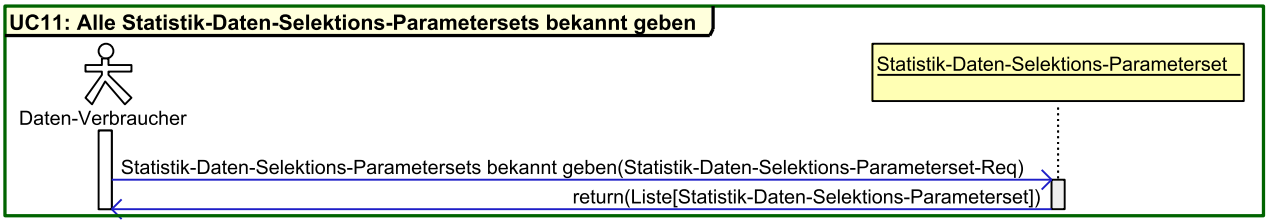


Abbildung 4: UC11 - Alle Statistik-Daten-Selektions-Parametersets bekannt geben

6.4 UC12: Statistik-Daten-Selektions-Parameterset entgegennehmen

6.4.1 Beschreibung

Abschnitt	Inhalt
Bezeichner	UC12
Anwendungsfallname	Statistik-Daten-Selektions-Parameter entgegennehmen
Autoren	Projektleitung
Prioritäten	Mittel
Kritikalität	Mittel
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	Das System nimmt ein Statistik-Daten-Selektions-Parameterset entgegen, welches vom Akteur übermittelt wurde.
Auslösendes Ereignis	Akteur übermittelt Statistik-Daten-Selektions-Parameterset.
Akteure	A1a: System-Administrator
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Selektions-Parameterset vgl. 7.2.3
Datenstruktur (Antwort): System → Akteur	Statistik-Daten-Selektions-Parameterset-Ack vgl. 7.2.8
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt triviale Empfangsbestätigung zur Kenntnis.
Hauptszenario	1) Eingabe eines Statistik-Daten-Selektions-Parametersets durch den Akteur.

	2) System verarbeitet das Statistik-Daten-Selektions-Parameterset. 3) System gibt triviale Empfangsbestätigung dem Akteur bekannt.
Alternativszenarien	-
Ausnahmeszenarien	1) Eingabe eines Statistik-Daten-Selektions-Parametersets durch den Akteur. 2) System überprüft das Statistik-Daten-Selektions-Parameterset. 3) System gibt dem Akteur bekannt, dass die Eingabe nicht verarbeitet werden konnte.
Fehlerfall	• Nicht-verarbeitbare Statistik-Daten-Selektions-Parametersets können detektiert und dem Benutzer bekannt gegeben werden.

6.4.2 Ablauf

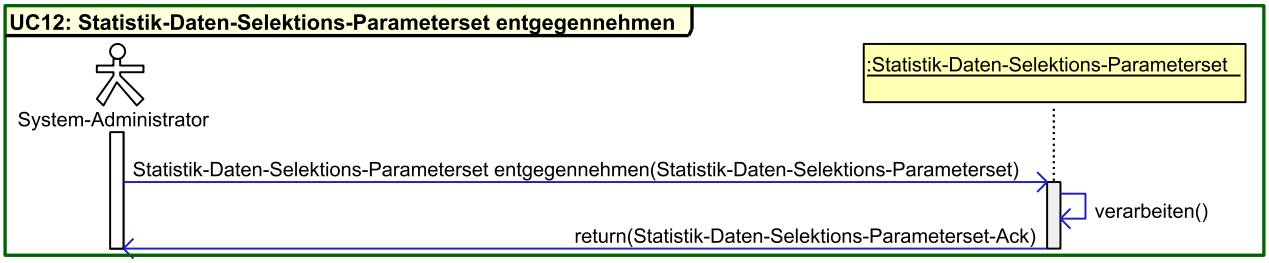


Abbildung 5: UC12 - Statistik-Daten-Selektions-Parameterset entgegennehmen

6.5 UC20: Statistik-Daten-Report bekannt geben

6.5.1 Beschreibung

Abschnitt	Inhalt
Bezeichner	UC20
Anwendungsfallname	Statistik-Daten-Report bekannt geben
Autoren	Projektleitung
Prioritäten	Mittel
Kritikalität	Mittel
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	Auf Anfrage gibt das System dem Akteur den gewünschten Statistik-Daten-Report bekannt.
Auslösendes Ereignis	Akteur setzt mittels Statistik-Daten-Report-Parameterset eine Anfrage ab.
Akteure	A1d: Informations-Verbraucher A3c: Informations-Verbraucher-System
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Report-Parameterset vgl. 7.2.7
Datenstruktur (Antwort):	Statistik-Daten-Report vgl. 7.2.8

System → Akteur	
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt den gewünschten Statistik-Daten-Report entgegen.
Hauptszenario	1) Anfrage zur Bekanntgabe eines Statistik-Daten-Reports durch den Akteur. 2) System verwendet das gesendete Statistik-Daten-Report-Parameterset zur Erstellung des gewünschten Statistik-Daten-Reports. 3) System gibt den Statistik-Daten-Report dem Akteur bekannt.
Alternativszenarien	-
Ausnahmeszenarien	1) Anfrage zur Bekanntgabe eines Statistik-Daten-Reports durch den Akteur. 2) System verwendet das gesendete Statistik-Daten-Report-Parameterset zur Erstellung des gewünschten Statistik-Daten-Reports. 3) Fehlerfall! 4) System gibt dem Akteur bekannt, dass das Statistik-Daten-Report-Parameterset nicht verarbeitet werden konnte.
Fehlerfall	• Verbindungsfehler können detektiert und dem Benutzer bekannt gegeben werden. • Ein fehlendes Statistik-Daten-Report-Parameterset bei der Anfrage kann detektiert und dem Benutzer bekannt gegeben werden. • Ein nicht verarbeitbares Statistik-Daten-Report-Parameterset kann detektiert und dem Benutzer bekannt gegeben werden.

6.5.2 Ablauf

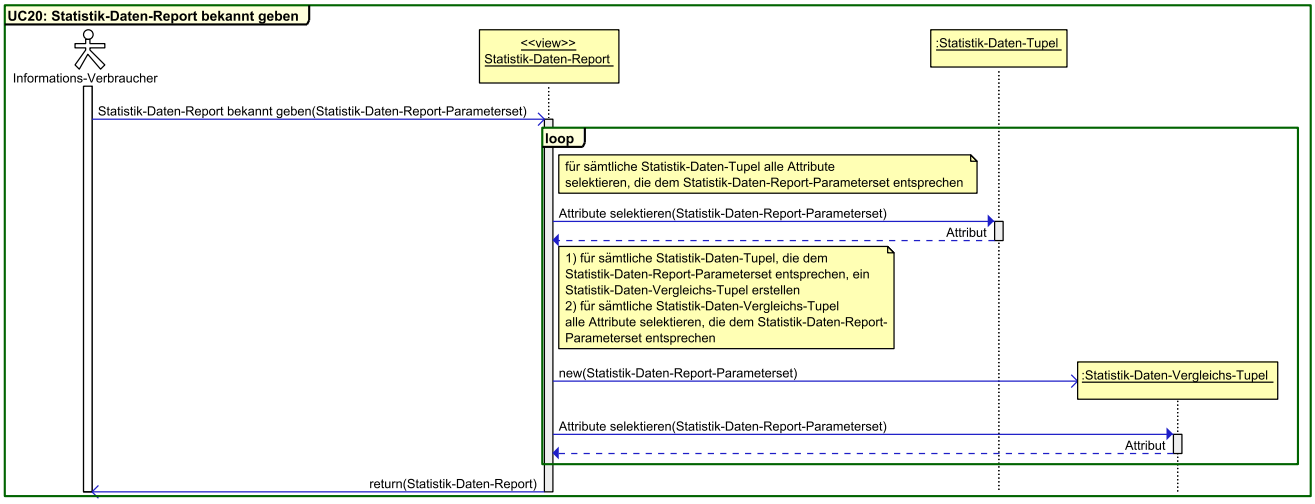


Abbildung 6: UC20 - Statistik-Daten-Report bekannt geben

6.6 UC21: Alle Statistik-Daten-Report-Parametersets bekannt geben

6.6.1 Beschreibung

Abschnitt	Inhalt
Bezeichner	UC21
Anwendungsfallname	Alle Statistik-Daten-Report-Parametersets bekannt geben
Autoren	Projektleitung
Prioritäten	Mittel
Kritikalität	Mittel
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	Auf Anfrage gibt das System dem Akteur sämtliche Statistik-Daten-Report-Parametersets bekannt.
Auslösendes Ereignis	Akteur setzt mittels Statistik-Daten-Report-Parameterset-Req eine Anfrage ab.
Akteure	A1d: Informations-Verbraucher A3c: Informations-Verbraucher-System
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Report-Parameterset-Req vgl. 7.2.9
Datenstruktur (Antwort): System → Akteur	Statistik-Daten-Report-Parameterset vgl. 7.2.7
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt sämtliche Statistik-Daten-Report-Parametersets entgegen.
Hauptszenario	1) Anfrage zur Bekanntgabe sämtlicher Statistik-Daten-Report-Parametersets durch den Akteur. 2) System gibt sämtliche Statistik-Daten-Report-Parametersets dem Akteur bekannt.
Alternativszenarien	-
Ausnahmeszenarien	-
Fehlerfall	• Verbindungsfehler können detektiert und dem Benutzer bekannt gegeben werden.

6.6.2 Ablauf

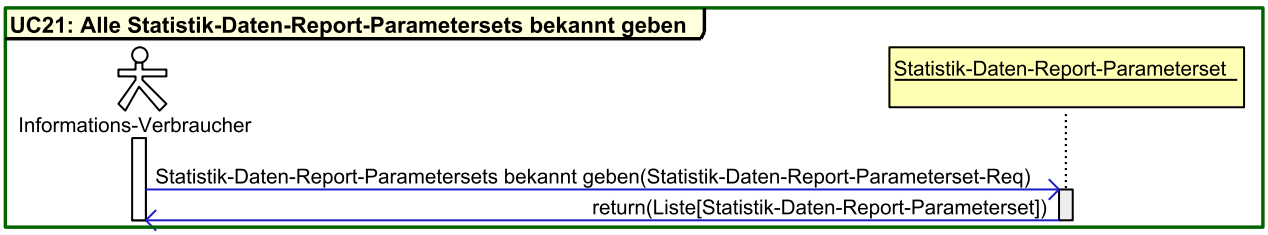


Abbildung 7: UC21 - Alle Statistik-Daten-Report-Parametersets bekannt geben

6.7 UC22: Statistik-Daten-Report-Parameterset entgegennehmen

6.7.1 Beschreibung

Abschnitt	Inhalt
Bezeichner	UC22
Anwendungsfallname	Statistik-Daten-Report-Parameterset entgegennehmen
Autoren	Projektleitung
Prioritäten	Mittel
Kritikalität	Mittel
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	Das System nimmt ein Statistik-Daten-Report-Parameterset entgegen, welches vom Akteur übermittelt wurde.
Auslösendes Ereignis	Akteur übermittelt Statistik-Daten-Report-Parameterset.
Akteure	A1a: System-Administrator
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Report-Parameterset vgl. 7.2.7
Datenstruktur (Antwort): System → Akteur	Statistik-Daten-Report-Parameterset-Ack vgl. 7.2.10
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt triviale Empfangsbestätigung zur Kenntnis.
Hauptszenario	1) Eingabe des Statistik-Daten-Report-Parametersets durch den Akteur. 2) System verarbeitet das Statistik-Daten-Report-Parameterset. 3) System gibt triviale Empfangsbestätigung dem Akteur bekannt.
Alternativszenarien	-
Ausnahmeszenarien	1) Eingabe eines Statistik-Daten-Report-Parametersets durch den Akteur. 2) System überprüft das Statistik-Daten-Report-Parameterset. 3) System gibt dem Akteur bekannt, dass die Eingabe nicht verarbeitet werden konnte.
Fehlerfall	• Nicht-verarbeitbare Statistik-Daten-Report-Parametersets können detektiert und dem Benutzer bekannt gegeben werden.

6.7.2 Ablauf

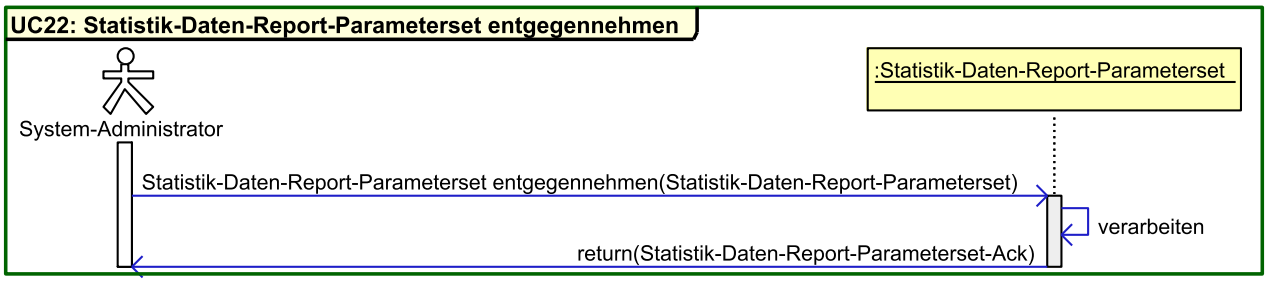


Abbildung 8: UC22 - Statistik-Daten-Report-Parameterset entgegennehmen

6.8 Anforderungsspezifikation

6.8.1 Beschreibung

Abschnitt	Inhalt
Bezeichner	UC30
Anwendungsfallname	Statistik-Daten-Tupel entgegennehmen
Autoren	Projektleitung
Prioritäten	Hoch
Kritikalität	Hoch
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	Der Akteur übermittelt dem System ein Statistik-Daten-Tupel, welches vom System gespeichert wird.
Auslösendes Ereignis	Akteur übermittelt Statistik-Daten-Tupel.
Akteure	A2a: Aktuator A2b: Legacy-System A2c: Daten-Import A3a: Daten-Manipulations-System
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Tupel vgl. 7.2.1
Datenstruktur (Antwort): System → Akteur	Statistik-Daten-Tupel-Ack vgl. 7.2.2
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt Speicherung des Statistik-Daten-Tupels zur Kenntnis.
Hauptszenario	1) Übermittlung des Statistik-Daten-Tupels durch den Akteur. 2) System überprüft das Statistik-Daten-Tupel. 3) System speichert das Statistik-Daten-Tupel.

	4) System gibt Speicherung des Statistik-Daten-Tupels dem Akteur bekannt.
Alternativszenarien	1) Übermittlung des Statistik-Daten-Tupels durch den Akteur. 2) System überprüft das Statistik-Daten-Tupel. 3) Extension Point: «Statistik-Daten-Tupel unvollständig» 4) System ermöglicht dem Akteur den UC31b durchzuführen. 5) Nach UC31b wird das Hauptszenario ab 2) ausgeführt.
Ausnahmeszenarien	1) Übermittlung des Statistik-Daten-Tupels durch den Akteur. 2) System überprüft das Statistik-Daten-Tupel. 3) System gibt dem Akteur bekannt, dass das Statistik-Daten-Tupel nicht verarbeitet werden konnte.
Fehlerfall	• Verbindungsfehler können detektiert und dem Benutzer bekannt gegeben werden. • Nicht-verarbeitbare Statistik-Daten-Tupels können detektiert und dem Benutzer bekannt gegeben werden. • Nicht erfolgreiche Speicherung eines Statistik-Daten-Tupels kann detektiert und dem Benutzer bekannt gegeben werden.

6.8.2 Ablauf

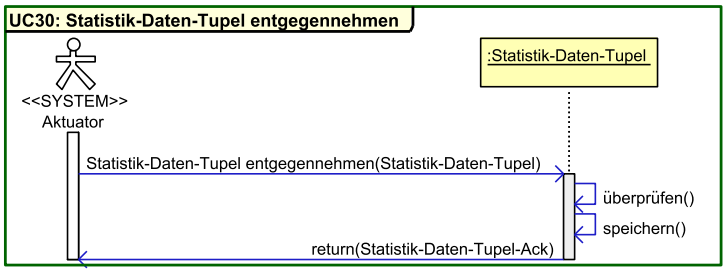


Abbildung 9: UC30 - Statistik-Daten-Tupel entgegennehmen

6.9 UC31: Modifiziertes Statistik-Daten-Tupel entgegennehmen

6.9.1 Beschreibung

Abschnitt	Inhalt
Bezeichner	UC31
Anwendungsfallname	Modifiziertes Statistik-Daten-Tupel entgegennehmen
Autoren	Projektleitung
Prioritäten	Mittel
Kritikalität	Hoch
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	UC31a) Ein bereits vorhandenes Tupel muss modifiziert werden (update). UC31b) Ein in UC30 übermitteltes Statistik-Daten-Tupel ist unvollständig. Der Akteur kann ein modifiziertes/vollständiges Statistik-Daten-Tupel übermitteln.

Auslösendes Ereignis	UC31a) Akteur übermittelt ein modifiziertes Statistik-Daten-Tupel UC31b) Extension Point «Statistik-Daten-Tupel unvollständig» UC30
Akteure	UC31a) A1b: Daten-Manipulator UC31b) A3a: Daten-Manipulations-System
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Tupel vgl. 7.2.1
Datenstruktur (Antwort): System → Akteur	Statistik- Daten-Tupel-Ack vgl. 7.2.2
Vorbedingungen	UC31a) - UC31b) Ein in UC30 übermitteltes Statistik-Daten-Tupel ist unvollständig. Der Extension Point «Statistik-Daten-Tupel unvollständig» wird ausgelöst.
Nachbedingungen	UC31a) - UC31b) Nach Abschluss von UC31, wird UC30 fortgesetzt.
Ergebnis	-
Hauptszenario	1) Übermittlung des modifizierten Statistik-Daten-Tupels durch den Akteur. 2) System setzt UC30 fort: «System überprüft das Statistik-Daten-Tupel.»
Alternativszenarien	-
Ausnahmeszenarien	-
Fehlerfall	vgl. UC30

6.9.2 Ablauf

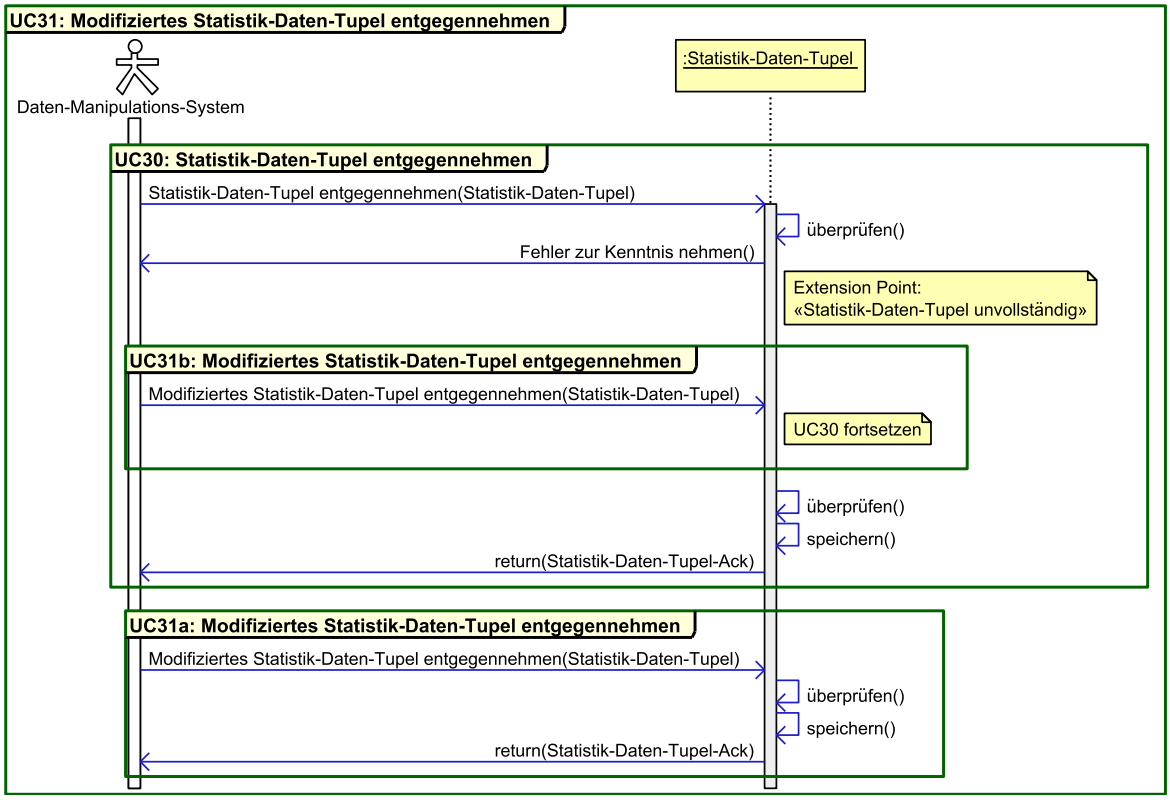


Abbildung 10: UC31 - Modifiziertes Statistik-Daten-Tupel entgegennehmen

7 Datenstrukturen der Systemschnittstellen

7.1 Einführung

In einem ersten Schritt sind die Anforderungen auf das Minimal Viable Product zugeschnitten. Gemäss Projektantrag [1] ersetzt das System eine Client-Datenbank-Architektur, das MVP muss daher in einem ersten Schritt, die SQL Abfragen eines Datenbank-Admin-Tools abstrahieren.

7.2 Logische Datenstrukturen

7.2.1 Statistik-Daten-Tupel

(nicht öffentlich)

7.2.2 Statistik-Daten-Tupel-Ack

(nicht öffentlich)

7.2.3 Statistik-Daten-Selektions-Parameterset

(nicht öffentlich)

7.2.4 Statistik-Daten-Selektion

(nicht öffentlich)

7.2.5 Statistik-Daten-Selektions-Parameterset-Req

(nicht öffentlich)

7.2.6 Statistik-Daten-Selektions-Parameterset-Ack

(nicht öffentlich)

7.2.7 Statistik-Daten-Report-Parameterset

(nicht öffentlich)

7.2.8 Statistik-Daten-Report

(nicht öffentlich)

7.2.9 Statistik-Daten-Report-Parameterset-Req

(nicht öffentlich)

7.2.10 Statistik-Daten-Report-Parameterset-Ack

(nicht öffentlich)

7.3 Physische Systemschnittstellen

Folgende Wireframes zeigen eine mögliche physische Repräsentation der logischen Datenstrukturen und die Schnittstellen des Systems.

7.3.1 UC10, UC11, UC12 Statistik-Daten-Selektion

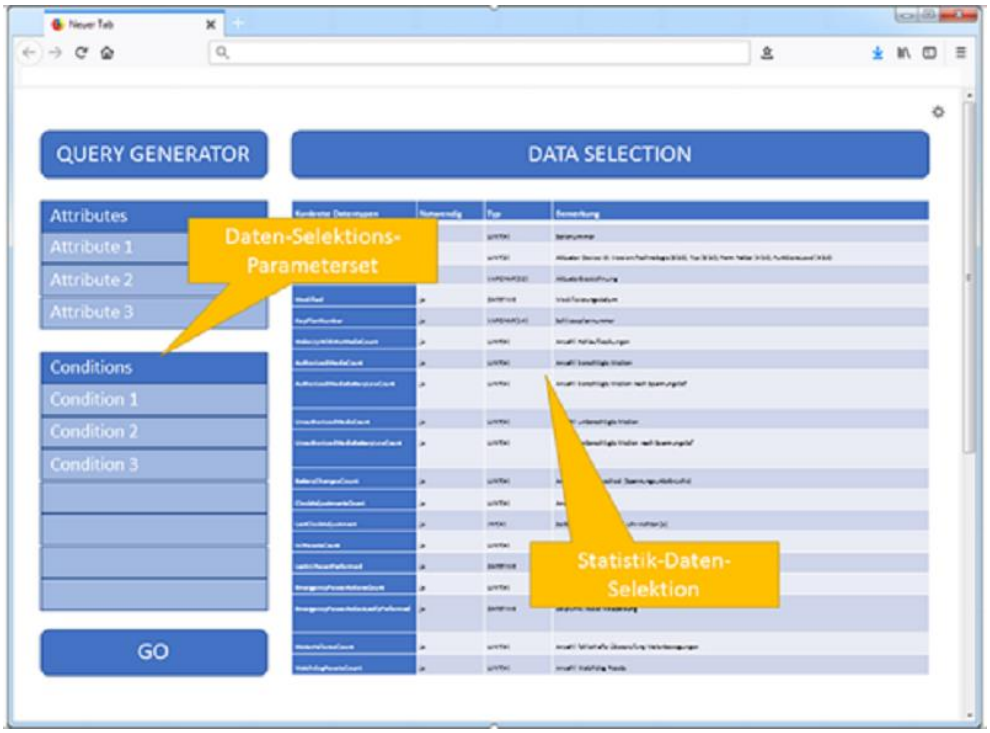


Abbildung 11: Daten-Selektion

7.3.2 UC20, UC21, UC23 Statistik-Daten-Report

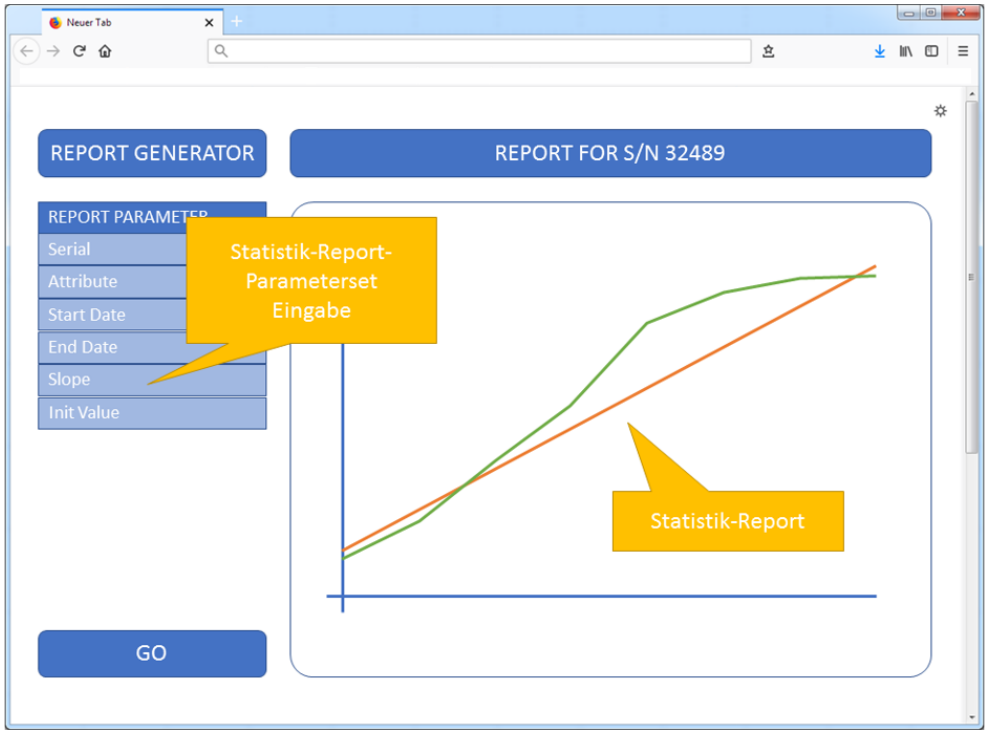


Abbildung 12: Report-Funktion

7.3.3 UC30, UC31 Statistik-Daten-Tupel

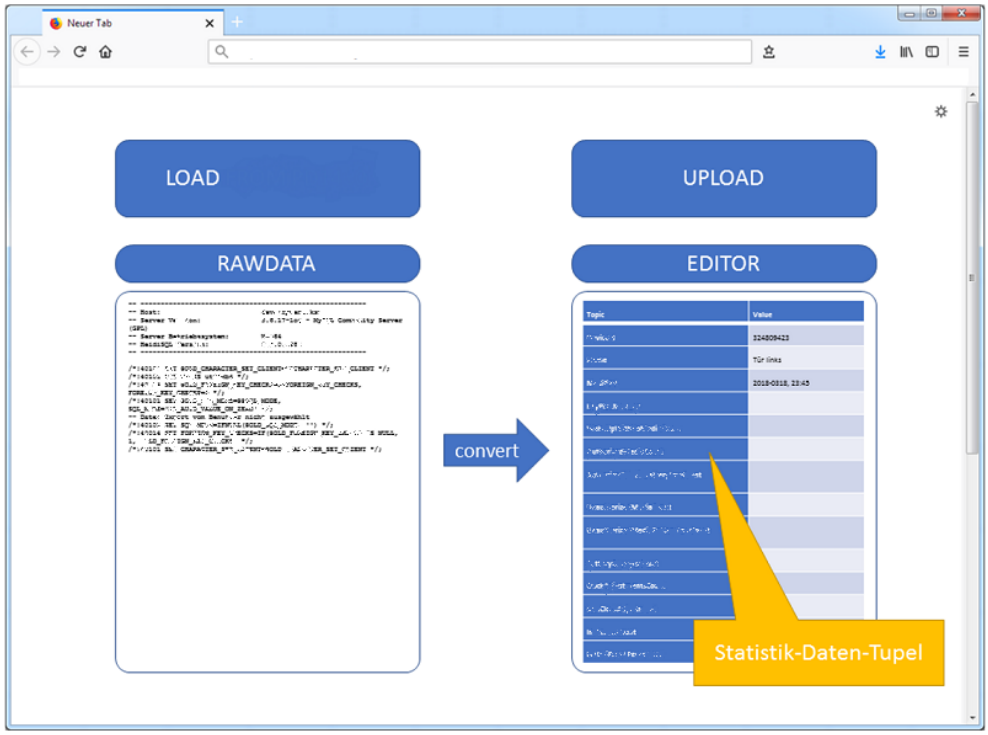


Abbildung 13: Daten-Eingabe

8 Nicht-funktionale Systemanforderungen

8.1 Benutzbarkeit

NF01: Die Durchführung der Anwendungsfälle sollte für einen Mitarbeiter mittels Tutorial innerhalb 30 Minuten selbstständig erlernbar sein.

NF02: Die Reports sollen komplette Artefakte darstellen, d.h. keine Nachbearbeitung oder Konvertierungen benötigen. Die genaue Form des Artefakts (z.B. pdf, Excel-Dokument) muss noch geklärt werden.

8.2 Verfügbarkeit

NF03: Der Zugriff auf das System muss während den Bürozeiten (7:00 – 18:00) gewährleistet sein.

NF04: Das System muss von 5 Benutzern gleichzeitig benutzt werden können.

8.3 Robustheit

NF05: Das System prüft die eingegebenen Daten auf Vollständigkeit. Sind die eingegebenen Daten nicht vollständig, wird der Benutzer darüber informiert.

8.4 Zuverlässigkeit

NF06: Die Fehlertoleranz des Systems gemessen in MTBF (Mean Time Between Failure) soll mindestens 30 Tage im ersten Betriebsjahr sein, danach ist eine Reevaluation notwendig.

NF07: Statistik-Daten, die vom System geprüft und verarbeitet wurden, müssen nach einem Betriebsausfall rekonstruierbar sein, sofern die Randbedingung geben ist, dass ein allfälliges Datenbank- und Betriebssystem gegen einen Betriebsausfall gesichert ist.

8.5 Effizienz

NF08: Eine Datenabfrage muss innerhalb 5 Sekunden eine gültige Antwort zurückgeben.

NF09: Eine Reportabfrage muss innerhalb 5 Sekunden eine gültige Antwort zurückgeben.

8.6 Systemsicherheit

NF10: Der Zugriff auf die Statistik-Daten ist rollenbasiert und mittels Passwort geschützt.

NF11: Das System darf nur von Mitarbeitern bedient und benutzt werden.

NF12: Das System darf nur in einem Netzwerk betrieben werden, welches vor Fremdzugriffen geschützt ist. (z.B. Firewall)

8.7 Änderbarkeit

NF13: Änderungen am System müssen innerhalb eines Arbeitstages verbreitet werden können.
d.h. Deployment innerhalb eines Arbeitstages

8.8 Betriebliche Anforderungen

NF14: Das System muss auf einem virtuellen Microsoft Windows Server (ab 2008) lauffähig sein.

NF15: Für die Wartung des Systems dürfen nicht mehr als 12 Arbeitstage pro Jahr aufgewendet werden.

9 Klassenbeschreibungen

9.1 Einführung

Abfragen entsprechen im Wesentlichen einer Datenbank-Sicht (View), daher wurde das essenzielle Klassendiagramm ähnlich moduliert, wie das für Views von Datenbanken gilt.

Gemäss Balzert [4] verbinden Abhängigkeiten (Dependency) eine View mit der zugehörigen Tabelle. Das heisst, dass jeweils mit Verwendung der jeweiligen Parameter entweder eine Statistik-Daten-Selektion oder ein Statistik-Daten-Report aus den verfügbaren Statistik-Daten-Tupels erzeugt wird.

Zudem wird das Statistik-Daten-Tupel um mindestens eine Statistik-Daten-Tupel-Modifikation erweitert, aus welcher ersichtlich ist, wann der Datenbank Eintrag entstanden ist oder zusätzlich modifiziert wurde.

9.2 Essenzielles Klassendiagramm

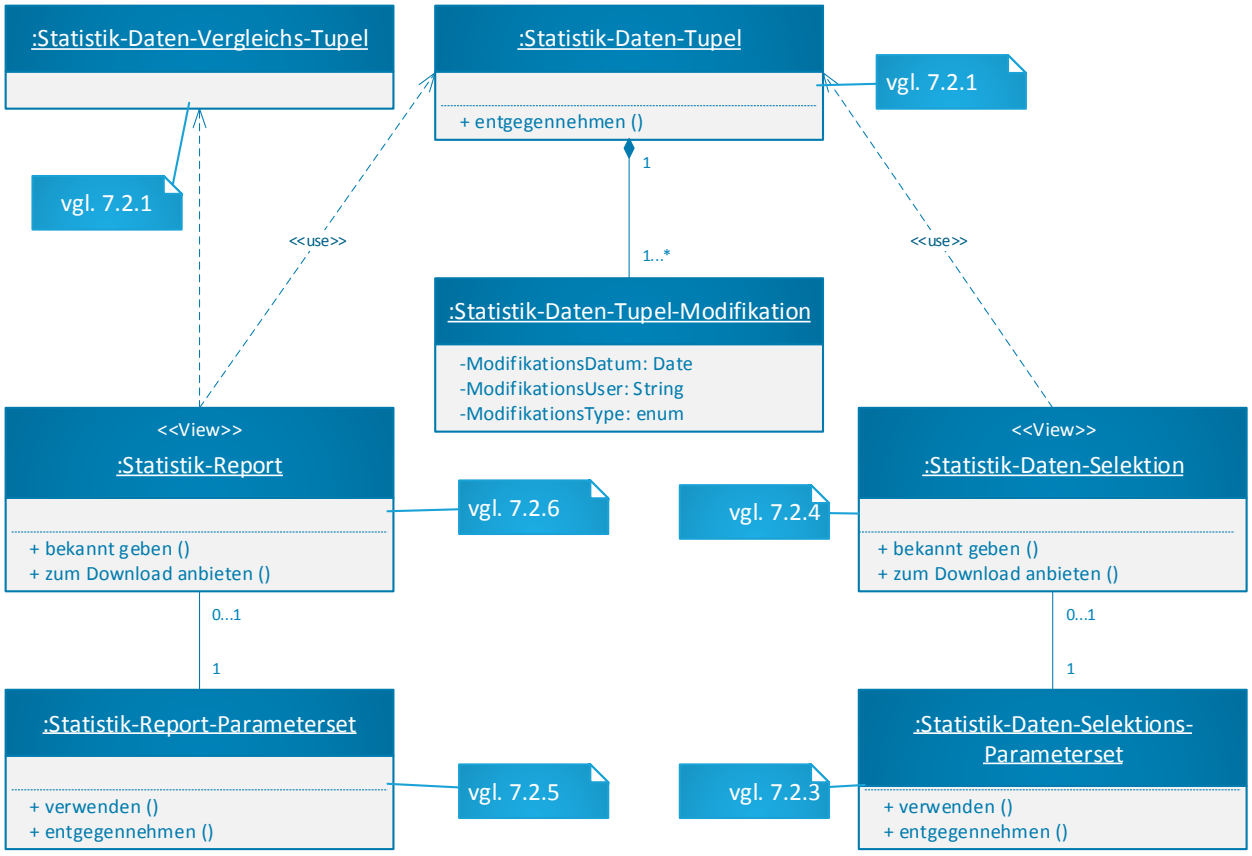


Abbildung 14: Essenzielles Klassendiagramm

9.3 Klassenbeschreibung

9.3.1 Statistik-Daten-Tupel

Beschreibung	Objekte der Klasse Statistik-Daten-Tupel werden erstellt, wenn neue Statistik-Daten ins System eingegeben werden.
Zustände	keine relevanten Zustände
Attribute	vgl. 7.2.1
entgegennehmen()	Argument: DTO gemäss 7.2.1 Antwort: Erfolgsmeldung Ablauf: 1) Übergebene Statistik-Daten werden überprüft. 2a) Bei Erfolg wird ein Statistik-Daten-Tupel-Modifikations Objekt erstellt und sämtliche Informationen persistiert. Zudem wird eine positive Erfolgsmeldung abgesetzt. 2b) Im Fehlerfall, wird eine negative Erfolgsmeldung abgesetzt.

9.3.2 Statistik-Daten-Tupel-Modifikation

Beschreibung	Die Klasse Statistik-Daten-Tupel-Modifikation erweitert die Klasse Statistik-Daten-Tupel dahingehen, dass eine Modifikationen der Daten nachverfolgbar ist.
Zustände	keine relevanten Zustände
Attribute	ModifikationsDatum: Date ModifikationsUser: String ModifikationsType: enum
Methode:	-

9.3.3 Statistik-Daten-Vergleichs-Tupel

Beschreibung	Objekte der Klasse Statistik-Daten-Vergleichs-Tupel werden aufgrund des Statistik-Daten-Report-Parametersets erstellt. Sie werden für den Vergleich berechnet.
Zustände	keine relevanten Zustände
Attribute	vgl. 7.2.1
Methode:	-

9.3.4 Statistik-Daten-Selektion

Beschreibung	Objekte der Klasse Statistik-Daten-Selektion werden erstellt, wenn eine neue Statistik-Daten-Selektion angefordert wird. Sie basieren auf einem entsprechenden Statistik-Daten-Selektions-Parameterset und benützten eine Untermenge von Statistik-Daten-Tupel Objekten.
Zustände	keine relevanten Zustände
Attribute	vgl. 7.2.4
bekannt geben()	Argument: Referenz auf ein Statistik-Daten-Selektions-Parameterset Antwort: DTO gemäss 7.2.4 Ablauf: 1) DTO wird gemäss dem referenzierten Statistik-Daten-Selektions-Parameterset erstellt. 2) Das DTO wird dargestellt.
zum Download anbieten()	Argument: Referenz auf ein Statistik-Daten-Selektions Objekt Antwort: Download-Link Ablauf: 1) DTO wird serialisiert und in eine Datei ausgegeben 2) Download Link wird bekanntgegeben

9.3.5 Statistik-Daten-Selektions-Parameterset

Beschreibung	Objekte der Klasse Statistik-Daten-Selektions-Parameterset werden erstellt, wenn eine neue Statistik-Daten-Auswahl entgegengenommen wird.
Zustände	keine relevanten Zustände
Attribute	vgl. 7.2.3
entgegennehmen()	Argument: DTO gemäss 7.2.3 Antwort: Erfolgsmeldung Ablauf: 1) Übergebenes Statistik-Daten-Selektions-Parameterset wird überprüft. 2a) Bei Erfolg wird ein Statistik-Daten-Selektions-Parameterset Objekt erstellt und sämtliche Informationen persistiert. Zudem wird eine positive Erfolgsmeldung abgesetzt. 2b) Im Fehlerfall, wird eine negative Erfolgsmeldung abgesetzt.
verwenden()	Argument: Referenz auf ein Statistik-Daten-Selektions-Parameterset Antwort: DTO gemäss 7.2.3 Ablauf: einfache getter-Funktion

9.3.6 Statistik-Daten-Report

Beschreibung	Objekte der Klasse Statistik-Daten-Report werden erstellt, wenn ein Report angefordert wird. Sie basieren auf einem entsprechenden Statistik-Daten-Report-Parameterset und benützten eine Untermenge von Statistik-Daten-Tupel und Statistik-Daten-Vergleichs-Tupel Objekten.
Zustände	keine relevanten Zustände
Attribute	vgl. 7.2.8
bekannt geben()	Argument: Referenz auf ein Statistik-Daten-Report-Parameterset Antwort: DTO gemäss 7.2.8 Ablauf: 1) Gemäss dem referenzierten Statistik-Daten-Report-Parameterset werden Statistik-Daten-Tupel ausgewählt und entsprechende Statistik-Daten-Vergleichs-Tupel berechnet. Diese Tupels werden dann zu einem DTO zusammengefasst. 2) Das DTO wird dargestellt.
zum Download anbieten()	Argument: Referenz auf ein Statistik-Daten-Report Objekt Antwort: Download-Link Ablauf: 1) DTO wird serialisiert und in eine Datei ausgegeben 2) Download Link wird bekanntgegeben

9.3.7 Statistik-Daten-Report-Parameterset

Beschreibung	Objekte der Klasse Statistik-Daten-Report-Parameterset werden erstellt, wenn eine neue Report-Anfrage entgegengenommen wird.
Zustände	keine relevanten Zustände
Attribute	vgl. 7.2.7
entgegennehmen()	Argument: DTO gemäss 7.2.7 Antwort: Erfolgsmeldung Ablauf: 1) Übergebenes Statistik-Daten-Report-Parameterset wird überprüft. 2a) Bei Erfolg wird ein Statistik-Daten-Report-Parameterset Objekt erstellt und sämtliche Informationen persistiert. Zudem wird eine positive Erfolgsmeldung abgesetzt. 2b) Im Fehlerfall, wird eine negative Erfolgsmeldung abgesetzt.
verwenden()	Argument: Referenz auf ein Statistik-Daten-Report-Parameterset Antwort: DTO gemäss 7.2.7 Ablauf: einfache getter-Funktion

10 Randbedingungen an die Realisierung

10.1 Benutzbarkeit

RB01: (nicht öffentlich)

10.2 Verfügbarkeit

RB02: (nicht öffentlich)

10.3 Robustheit

(keine)

10.4 Zuverlässigkeit

RB03: Sämtlich Software- und Hardwaresysteme die zum Betrieb des «Statistik-Daten Auswertungs» Systems benötigte werden müssen gegen Betriebsausfall gesichert sein.

10.5 Effizienz

RB04: Die maximale Latenzzeit und die minimale Bandbreite, die gegeben sein müssen, sind noch zu bestimmen.

10.6 Systemsicherheit

RB05: Das System darf nur von Mitarbeitern bedient und benutzt werden.

RB06: Das System darf nur in einem Netzwerk betrieben werden, welches vor Fremdzugriffen geschützt ist. (z.B. Firewall)

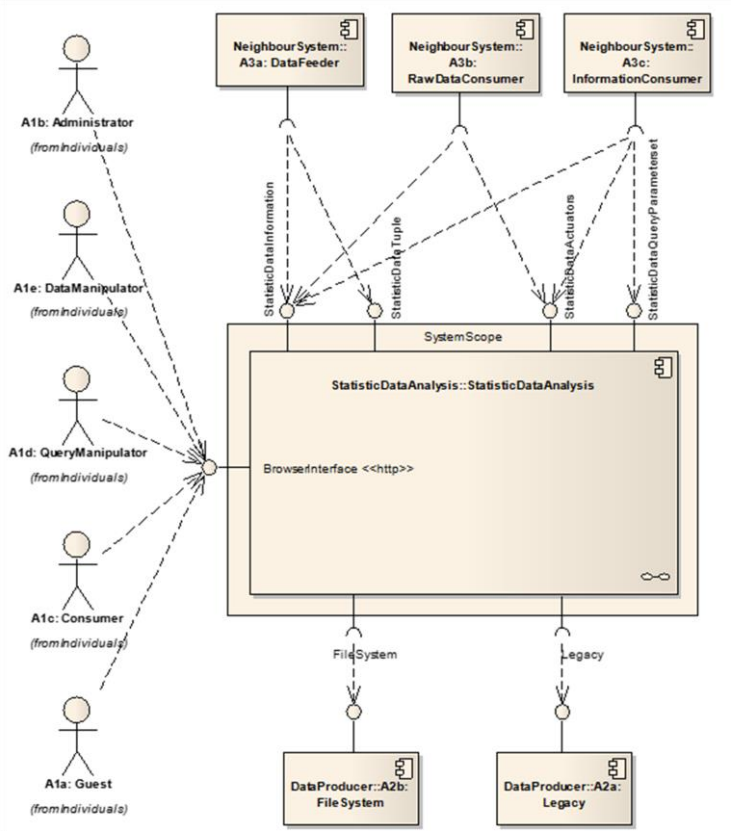
10.7 Änderbarkeit

RB07: (nicht öffentlich)

10.8 Betriebliche Anforderungen

RB08: Virtueller Microsoft Windows Server

RB09: Datenbanksystem (DBS) muss noch definiert werden.



Statistik-Daten Auswertung

Masterarbeit HSR-MAS-SE 2016

Architekturdokumentation

Version:	006
Status:	RELEASE
Release Date:	26.08.2018
Classification:	PUBLIC

Document-ID: n/a
architekturdokumentation_public.docx

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Zu diesem Dokument

1.1 Zweck

Die Architekturdokumentation hilft Stakeholdern einerseits die Architektur des Systems zu verstehen und andererseits dokumentiert sie die zugrunde liegenden Entscheidungen und Konzepte. Zum Schluss listet sie offene Punkte und Risiken, die in Zukunft weiterverfolgt werden müssen.

Die Form und Gliederung der Architekturdokumentation folgt dem arc42-Template: https://arc42.org/

1.2 Gültigkeit

Dieses Dokument ist über die ganze Projektdauer gültig.

1.3 Verfügbarkeit des Dokumentes

Dieses Dokument befindet sich im Projektverzeichnis: 11_Architekturdokumentation

1.4 Graphische Darstellungen

Die graphische Dokumentation wurde mit «Enterprise Architect» (vgl. 13.1) gemacht. Die meisten manuell bearbeiteten Diagramme sind in dieser Architekturdokumentation enthalten. Weitere Diagramme wurden zudem automatisch aus dem Quellcode erzeugt.

Die Datei (StatisticDataAnalysis.eap) befindet sich im Projektverzeichnis:
11_Architekturdokumentation

Eine html-Version des EA-Projekts befindet sich im Projektverzeichnis:
11_Architekturdokumentation\EA_Doc

1.5 Dokumentation der Programmierschnittstellen (Swagger)

Eine interaktive Dokumentation der Programmierschnittstellen befindet sich im Projektverzeichnis:

11_Architekturdokumentation\Swagger_Doc\AccessCoordination
11_Architekturdokumentation\Swagger_Doc\DataPersistence

2 Weiterführende Informationen

2.1 Entscheidungen

Entscheidungen (z.B. Änderungen betreffend Projektantrag oder Anforderungsspezifikation) werden wie folgt im Text markiert und sind im Kapitel 14 erläutert:

EAxx	Abgrenzende Entscheidungen
EFxx	Funktionale Entscheidungen
ETxx	Technologische Entscheidungen

2.2 Definitionen, Akronyme und Abkürzungen

Sämtliche Definitionen, Akronyme und Abkürzungen werden zentral in einem Projektglossar dokumentiert. Das Projektglossar wird kontinuierlich angepasst.

Das Projektglossar befindet sich im Projektverzeichnis: 09_Glossar

2.3 Ergänzende Dokumente

Ref.	Beschreibung
[1]	Projektantrag: «Statistik und Event-Log Auswertung», VER011
[2]	Projektleitdokumentation: «Statistik-Daten Auswertung, Projektleitdokumenation», VER005
[3]	Anforderungsspezifikation: «Statistik-Daten Auswertung, Anforderungsspezifikation», VER006
[4]	Testkonzept: «Statistik-Daten Auswertung, Testkonzept», VER003
[5]	Enterprise Architect, Sparx Systems: StatisticDataAnalysis.eap

2.4 Prozessbezogene Dokumente

Ref.	Beschreibung
[6]	Masterarbeit MAS-SE 2016, Richtlinie MAS-SE
[7]	Kursunterlagen MAS-SE «Kommunikation für verteilte Systeme»
[8]	Kursunterlagen MAS-SE «Architekturen verteilter Systeme»
[9]	Kursunterlagen MAS-SE «Web Frameworks und Internettechnologien Advanced»
[10]	Kursunterlagen MAS-SE «Cloud Computing»
[11]	Verteilte Systeme, 2. Auflage, Alexander Schill, Thomas Springer ISBN: 978-3-642-25795-7
[12]	Effektive Software-Architekturen, 7. Auflage, Gernot Starke ISBN: 978-3-446-44361-7
[13]	Software-Architektur kompakt, 2. Auflage, Gernot Starke, Peter Hruschka ISBN: 978-3-8274-2834-9
[14]	Patterns kompakt, 4. Auflage, Karl Eilebrecht, Gernot Starke ISBN: 978-3-642-34717-7

[15]	arc42 in Aktion, 1. Auflage, Gernot Starke, Peter Hruschka ISBN: 978-3-446-44801-8
[16]	REST und HTTP, 3. Auflage, Tilkov, Eigenbrodt, Schreier, Wolf

2.5 Versionshistorie

(nicht öffentlich)

2.6 Verteilung

Ver.	Name	Rolle / Titel / Link
006	MAS-Arbeit	MAS-Arbeit

3 Einführung und Ziele

Dieses Kapitel behandelt nochmals die Themen der Anforderungsspezifikation und erweitert oder korrigiert Punkte, die bei der Freigabe der Anforderungsspezifikation noch nicht geklärt waren, oder aber aufgrund von Architekturentscheidungen geändert werden mussten.

3.1 Aufgabenstellung

Die bestehende Client-Datenbank-Architektur zur Speicherung von Aktuatorkenngrössen (Statistik-Daten) soll durch eine flexiblere Architektur ersetzt werden. Für die Datenanalyse sollen die Statistik-Daten aufbereitet und rollenspezifisch zur Verfügung gestellt werden. Das Abfragen und Analysieren der Daten soll für jeden Akteur ohne fremdes Zutun möglich sein. Bisher waren Akteure oft auf die Hilfe von Personen mit Datenbank-Fachwissen angewiesen.

3.1.1 Abgrenzung gegenüber dem ursprünglichen Projektantrag

EA01	Keine direkte Anbindung von Hardware
EA02	Event-Log Daten werden nicht in das System übernommen

3.2 Stakeholder

(nicht öffentlich)

3.3 Akteure

Bei der Implementierung hat sich gezeigt, dass die Aufteilung der Akteure nicht ideal gewählte wurde. Die nachfolgende Aufteilung unterscheidet sich daher von der Anforderungsspezifikation.

EF03	Für die Implementierung der Rollen musste eine Benutzerverwaltung mit Authentifizierung und Autorisierung eingeführt werden. Dadurch entstand die Rolle «Guest»

3.3.1 A1: Natürliche Person

Akteure können mehr als eine Rolle besitzen.

A1a: Guest

«Guest» ist ein nicht registrierter Anwender. Er hat keine Rollen und kann (ausser der Registrierung) keine Anwendungsfälle ausführen.

A1b: Administrator

Der «Administrator» verwaltet und konfiguriert das System. Er teilt anderen Akteuren ihre Rollen zu.

A1c: Consumer

Nach erfolgreicher Registrierung wird ein «Guest» automatisch zum «Consumer». Der «Consumer» kann auf Statistik-Daten-Abfrage-Parametersets lesend zugreifen und diese verwenden um Statistik-Daten-Selektionen, -Reports und -Alarmierungen zu konsumieren. Zusätzlich hat er die Möglichkeit die Daten einzelner Aktuatoren direkt zu betrachten.

A1d: QueryManipulator

Der «QueryManipulator» erstellt, editiert oder löscht Statistik-Daten-Abfrage-Parametersets.

A1e: DataManipulator

Der «DataManipulator» erstellt, importiert und verändert einzelne Statistik-Daten-Tupel.

Rollenzuteilung

Stakeholder	Guest	Administrator	Consumer	Query-Manipulator	Data-Manipulator
Entwickler (S3, S4)	X	X	X	X	X
Supporter (S5)	X		X	X	X
Test-Verifikation-Validation-Team (S6)	X		X	X	X
Feldtest-Manager (S6)	X		X	X	
Projektleiter (S7)	X		X		

3.3.2 A2: Datenquellen

Betriebskenngrössen werden während der Laufzeit eines Aktuators erzeugt und intern abgespeichert. Diese Daten können auf unterschiedliche Weise ausgelesen werden. Nach einer automatischen oder manuellen Aufbereitung, können die Daten als Statistik-Daten-Tupel dem «Statistik-Daten Auswertung»-System übergeben werden.

A2a: DataProducer: Legacy

Das Legacy-System liefert sämtliche erfassten Statistik-Daten-Tupels. Es funktioniert als Pipeline zwischen Hardware und «Statistik-Daten Auswertung»-System.

A2b: DataProducer: FileSystem

Für Testzwecke sollen Statistik-Daten-Tupel via Dateisystem importiert werden können.

3.3.3 A3: Verwaltungs-/Hostsysteme

(nicht öffentlich)

A3a: DataFeeder

Ein «DataFeeder»-System erzeugt einzelne Statistik-Daten-Tupel. Das «Statistik-Daten Auswertung»-System kennt ein allfälliges «DataFeeder»-System nicht. Es wird eine Schnittstelle angeboten, die durch ein allfälliges «DataFeeder»-System angesprochen werden kann.

A3b: RawDataConsumer

Ein «RawDataConsumer»-System konsumiert Statistik-Daten einzelner Aktuatoren. Das «Statistik-Daten Auswertung»-System kennt ein «RawDataConsumer»-System nicht. Es wird eine Schnittstelle angeboten, die durch ein allfälliges «RawDataConsumer»-System angesprochen werden kann.

A3c: InformationConsumer

Ein «InformationConsumer»-System konsumiert Statistik-Daten in Form von Selektionen, Reports oder Alarmierungen. Das «Statistik-Daten Auswertung»-System kennt ein allfälliges «InformationConsumer»-System nicht. Es wird eine Schnittstelle angeboten, die durch ein allfälliges «InformationConsumer»-System angesprochen werden kann.

3.4 Erweiterung der Anwendungsfälle gegenüber der Anforderungsspezifikation

3.4.1 UC40: Statistik-Daten-Alarmierung bekannt geben

Der folgende Anwendungsfall unterscheidet sich nur im Ergebnis von den Anwendungsfällen UC10 und UC20. Anstatt einer Statistik-Daten-Selektion oder eines Statistik-Daten-Report ist das Ergebnis eine Statistik-Daten-Alarmierung.

Abschnitt	**Inhalt**
Bezeichner	UC40
Anwendungsfallname	Statistik-Daten-Alarmierung bekannt geben
Autoren	Projektleitung
Prioritäten	Tief
Kritikalität	Mittel
Quelle	Projektantrag [1]
Verantwortlicher	Projektleitung
Kurzbeschreibung	Auf Anfrage gibt das System dem Akteur die gewünschte Statistik-Daten-Alarmierung bekannt.
Auslösendes Ereignis	Akteur setzt mittels Statistik-Daten-Abfrage-Parameterset eine Anfrage ab.
Akteure	A1c: Consumer A3c: InformationConsumer
Datenstruktur (Anfrage): Akteur → System	Statistik-Daten-Abfrage-Parameterset
Datenstruktur (Antwort): System → Akteur	Statistik-Daten-Alarmierung
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt gewünschte Statistik-Daten-Alarmierung entgegen.
Hauptszenario	1) Anfrage zur Bekanntgabe einer Statistik-Daten-Alarmierung durch den Akteur. 2) System verwendet das gesendete Statistik-Daten-Abfrage-Parameterset zur Erstellung der gewünschten Statistik-Daten-Alarmierung. 3) System gibt die Statistik-Daten-Alarmierung dem Akteur bekannt.
Alternativszenarien	-
Ausnahmeszenarien	vgl. UC10 und UC20 der Anforderungsspezifikation [3]
Fehlerfall	vgl. UC10 und UC20 der Anforderungsspezifikation [3]

3.4.2 UC50: Statistik-Daten-Aktuator bekannt geben

Abschnitt	Inhalt
Bezeichner	UC50
Anwendungsfallname	Statistik-Daten-Aktuator bekannt geben
Autoren	Projektleitung
Prioritäten	Mittel
Kritikalität	Mittel
Quelle	Meeting_Minutes_20180514.docx
Verantwortlicher	Projektleitung
Kurzbeschreibung	Auf Anfrage gibt das System dem Akteur sämtliche Daten eines Aktuators bekannt.
Auslösendes Ereignis	Akteur setzt mittels Seriennummer eine Anfrage ab.
Akteure	A1c: Consumer A3b: RawDataConsumer
Datenstruktur (Anfrage): Akteur → System	Seriennummer
Datenstruktur (Antwort): System → Akteur	sämtliche Statistik-Daten eines Aktuators (Statistik-Daten-Aktuator)
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt gewünschte Statistik-Daten des Aktuators entgegen.
Hauptszenario	1) Anfrage zur Bekanntgabe der Statistik-Daten eines Aktuators durch den Akteur. 2) System verwendet die gesendete Seriennummer und gibt die Statistik-Daten des Aktuators bekannt.
Alternativszenarien	-
Ausnahmeszenarien	1) Anfrage zur Bekanntgabe der Statistik-Daten eines Aktuators durch den Akteur. 2) Fehlerfall! 3) System gibt dem Akteur bekannt, dass die Statistik-Daten des Aktuators nicht verarbeitet werden konnte.
Fehlerfall	• Verbindungsfehler können detektiert und dem Benutzer bekannt gegeben werden. • Eine unbekannte Seriennummer kann detektiert und dem Benutzer bekannt gegeben werden. • Eine nicht-verarbeitbare Seriennummer kann detektiert und dem Benutzer bekannt gegeben werden.

3.4.3 UC60: API-Informationen bekannt geben

Abschnitt	Inhalt
Bezeichner	UC60
Anwendungsfallname	API-Information bekannt geben
Autoren	Projektleitung
Prioritäten	Mittel
Kritikalität	Mittel
Quelle	Keine
Verantwortlicher	Projektleitung
Kurzbeschreibung	Auf Anfrage gibt das System dem Akteur die API-Informationen bekannt.
Auslösendes Ereignis	Triviale Anfrage auf Ressource
Akteure	A3a: DataFeeder A3b: RawDataConsumer A3c: InformationConsumer
Datenstruktur (Anfrage): Akteur → System	Keine (einfache GET Anfrage)
Datenstruktur (Antwort): System → Akteur	API-Informationen
Vorbedingungen	-
Nachbedingungen	-
Ergebnis	Akteur nimmt API-Informationen entgegen.
Hauptszenario	1) Anfrage zur Bekanntgabe der API-Informationen durch den Akteur. 2) System gibt API-Informationen bekannt.
Alternativszenarien	-
Ausnahmeszenarien	1) Anfrage zur Bekanntgabe der API-Informationen durch den Akteur. 2) Fehlerfall! 3) System gibt dem Akteur bekannt, dass die Anfrage nicht verarbeitet werden konnte.
Fehlerfall	• Verbindungsfehler können detektiert und dem Benutzer bekannt gegeben werden.

3.4.4 UC100: Benutzerregistrierung entgegennehmen

(siehe unten)

3.4.5 UC101: Benutzer-Rollen-Wechsel entgegennehmen

Diese beiden Anwendungsfälle wurde durch die Implementierung der Benutzerverwaltung mit Authentifizierung und Autorisierung eingeführt. Sie sind der Vollständigkeit halber aufgeführt. Die Architektur beruht auf der Dokumentation von Microsoft. Aus diesem Grund wird hier auf die Dokumentation von Microsoft verwiesen:
Authentifizierung: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/

Autorisierung: https://docs.microsoft.com/en-us/aspnet/core/security/authorization/

3.5 Übersicht Anwendungsfälle

Selektionen, Reports und Alarmierung und deren Handhabung wurden bei der Implementierung vereinheitlicht. Für alle drei Arten wird immer der gleiche Typ Statistik-Daten-Abfrage-Parameterset verwendet. Aus diesem Grund konnte die Anzahl bestehender Anwendungsfälle reduziert werden.

UC21 (Report) wurde in UC11 (Selektion) integriert, welcher zudem auch für die Handhabung von Alarmierungen verwendet werden kann. Das Gleiche gilt für UC12 (Report) und UC22 (Selektion).

Das System muss folgende Anwendungsfälle erfüllen können		**Akteur**
UC10:	Statistik-Daten-Selektion bekannt geben	A1c, A3c
UC11:	Alle Statistik-Daten-Abfrage-Parametersets bekannt geben	A1c, A3c
UC12:	Statistik-Daten-Abfrage-Parameterset entgegennehmen	A1d
UC20:	Statistik-Daten-Report bekannt geben	A1c, A3c
UC40:	Statistik-Daten-Alarmierung bekannt geben	A1c, A3c
UC30:	Statistik-Daten-Tupel entgegennehmen	A1e, A2a, A2b, A3a
UC31:	Modifiziertes Statistik-Daten-Tupel entgegennehmen	A1e
UC50:	Statistik-Daten-Aktuator bekannt geben	A1c, A3b
UC60:	API-Informationen bekannt geben	A3a, A3b, A3c
UC100:	Benutzerregistrierung entgegennehmen	A1a
UC101:	Benutzer-Rollen-Wechsel entgegennehmen	A1b

Das folgende Diagramm (nächste Seite) ist eine Übersicht der Anwendungsfälle:

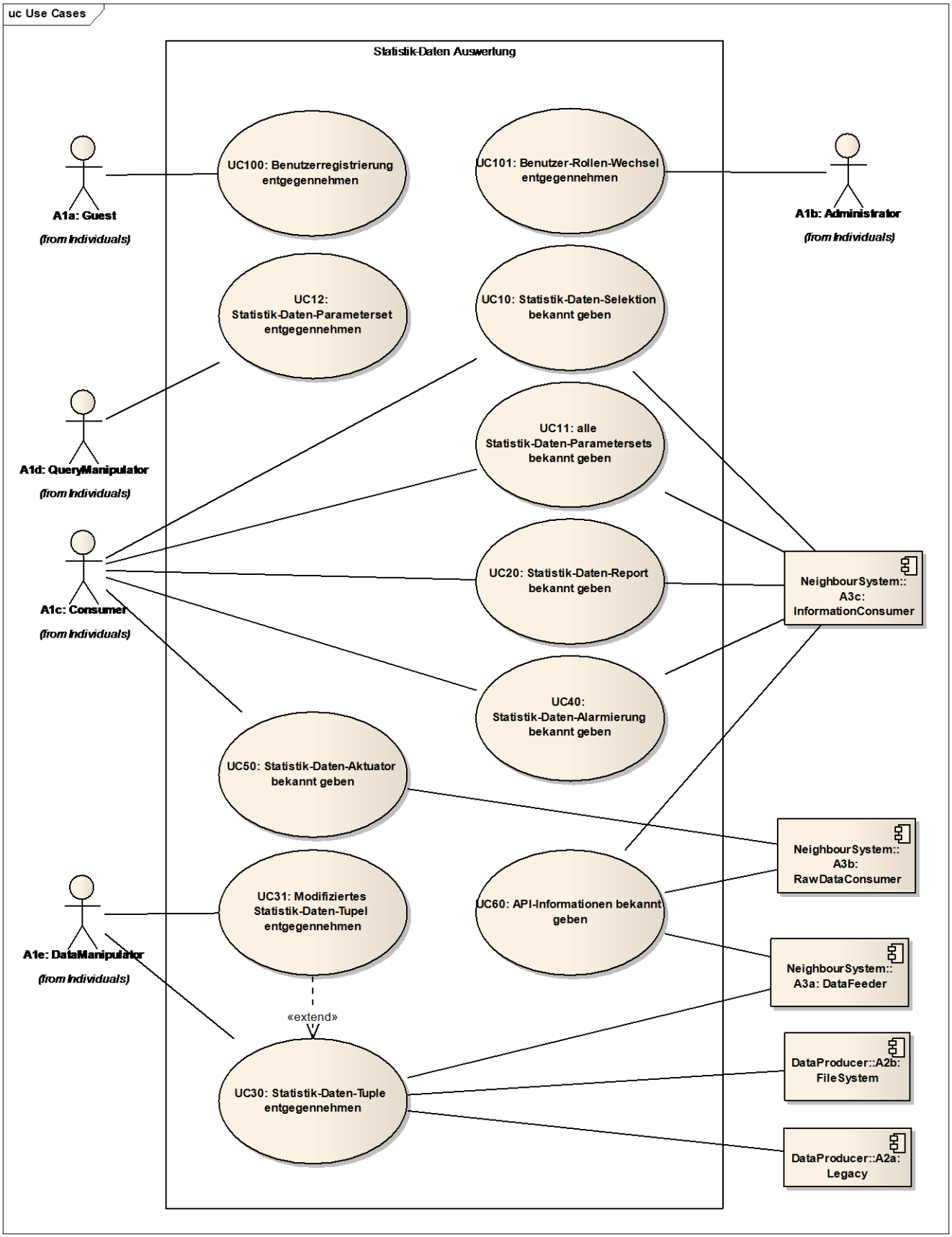


Abbildung 1: Übersicht der Anwendungsfälle

3.6 Qualitätsziele / Qualitätsanforderungen

Die nicht-funktionalen Systemanforderungen der Anforderungsspezifikation (Kapitel 8) sind ein wesentlicher Teil der Qualitätsziele. Nicht-funktionale Systemanforderungen, die einen direkten Einfluss auf die Architektur haben, wurden daher in diese Liste übernommen und mit weiteren Punkten ergänzt.

Nicht alle Qualitätsanforderungen können durch die jetzige Architektur erfüllt werden. Ziele wäre aber, dass die hier gelisteten Qualitäten den Weg für die Weiterentwicklung aufzeigen.

Bemerkung: Einträge, die mit «new» gekennzeichnet sind, konnten bei der Implementierung aus Zeitgründen nicht berücksichtigt werden.

		Qualitätsziel	Erläuterung	Stand
QZ01	**NF10**	**Funktionalität**	Der Zugriff auf die Statistik-Daten ist rollenbasiert und mittels Passwort geschützt.	
QZ01.1			Anbindung an das Active Directory wäre wünschenswert.	new
QZ01.2			Rollen müssen eindeutig zugeteilt werden können. Es wird daher eine Benutzerverwaltung mit Authentifizierung und Autorisierung benötigt.	
QZ01.3			Rollen müssen zugeteilt und entfernt werden können.	
QZ01.4			Neue Rolle sollten erzeugt werden können.	new
QZ02		**Funktionalität**	Gespeicherte Statistik-Daten-Abfrage-Parametersets müssen permanent gespeichert und für alle autorisierten Benutzer sichtbar sein.	
QZ02.1			Statistik-Daten-Abfrage-Parametersets können gruppiert und spezifisch an Benutzergruppen zugewiesen werden.	new
QZ03	**NF04**	**Multisession tauglich**	Das System muss von 5 Benutzern gleichzeitig benutzt werden können.	
QZ03.1			Benutzer können Daten-Importe persönlich verwalten.	
QZ03.2			System muss verhindern, dass Benutzer zur gleichen Zeit die gleichen Daten manipulieren (z.B. Statistik-Daten-Abfrage-Parameterset).	new
QZ04	**NF05**	**Robustheit**	Das System prüft die eingegebenen Daten auf Vollständigkeit. Sind die eingegebenen Daten nicht vollständig, wird der Benutzer darüber informiert.	
QZ04.1			Benutzereingaben werden bereits bei der Eingabe geprüft. Client-Validation	
QZ04.2			Benutzereingaben werden bei der Verarbeitung geprüft. Server-Validation	new

QZ05	**NF07**	**Zuverlässigkeit**	Statistik-Daten, die vom System geprüft und verarbeitet wurden, müssen nach einem Betriebsausfall rekonstruierbar sein, sofern die Randbedingung geben ist, dass ein allfälliges Datenbank- und Betriebssystem gegen einen Betriebsausfall gesichert ist.	
QZ05.1			Die Anwendung verwalten nur minimale Nutzerdaten des Session-States im Speicher des Clients oder Servers. Ziel ist es, dass der Session-State in der Datenbank verwaltet wird.	
QZ06	**NF08**	**Effizienz**	Eine Datenabfrage muss innerhalb von 5 Sekunden eine gültige Antwort zurückgeben.	
QZ07	**NF09**	**Effizienz**	Eine Reportabfrage muss innerhalb von 5 Sekunden eine gültige Antwort zurückgeben.	
QZ08		**Sicherheit**	Der Inhalt der Legacy Datenbank darf durch das «Statistik-Daten Auswertung»-System nicht verändert werden.	
QZ08.1			Die Legacy Datenbank verwaltet weiterhin die Rohdaten, dadurch wird das Gesamtsystem nicht eingeschränkt, sondern erweitert.	
QZ09		**Datenintegrität**	Neue Statistik-Daten-Tupel werden bestehenden Aktuatoren korrekt zugewiesen. Falls die Seriennummer des Statistik-Daten-Tupel noch nicht in der Datenbank existiert, wird ein neuer Aktuator angelegt.	
QZ09.1			Für spezielle Anwendungsfälle soll eine Verknüpfung zwischen zwei Seriennummer möglich sein.	New
QZ10		**Änderbarkeit**	Änderungswünsche zu einem späteren Zeitpunkt (nach Pilotbetrieb) müssen möglich sein. Open-Closed-Prinzip	
QZ11	**NF06**	**Stabilität**	Die Fehlertoleranz des Systems gemessen in MTBF (Mean Time Between Failure) soll mindestens 30 Tage im ersten Betriebsjahr sein	
QZ12		**Benutzbarkeit**	Das System soll im Fehlerfall in einen definierten Zustand übergehen. (z.B. Error-Seite)	

4 Randbedingungen

Die in der Anforderungsspezifikation gelisteten Randbedingungen (Kapitel 10) wurden neu beurteilt. Die dort gelisteten Randbedingungen betreffen grössten Teils den Betrieb und die Umsysteme des Systems und haben daher wenig Einfluss auf die Architektur. Es gibt aber sehr wohl Randbedingungen, die beachtet werden müssen:

Id	Randbedingung	Erläuterung
RB10	Organisatorisch	Die Entwicklung des «Statistik-Daten Auswertung»-Systems muss die Vorgaben der «Masterarbeit MAS-SE 2016, Richtlinie MAS-SE» [6] befolgen.
RB11	Analysierbarkeit	Es muss möglich sein, Fehler die im Betrieb auftreten, zu analysieren. Eine Möglichkeit ist das Loggen in Dateien.
RB12	Technologie	Die Anzahl eingesetzter Technologien soll niedrig gehalten werden.
RB13	Flexibilität	Das «Statistik-Daten Auswertung»-System muss auf einem der bestehenden Systemen lauffähig sein, d.h. Windows Server 2008, 2012 oder Windows 7
RB14	Interoperabilität	Das «Statistik-Daten Auswertung»-System darf andere Webservices nicht beeinflussen, d.h. entweder zu Apache HTTP Server kompatibel sein oder aber Apache nicht an der Ausführung hindern

5 Kontextabgrenzung

5.1 Fachlicher Kontext

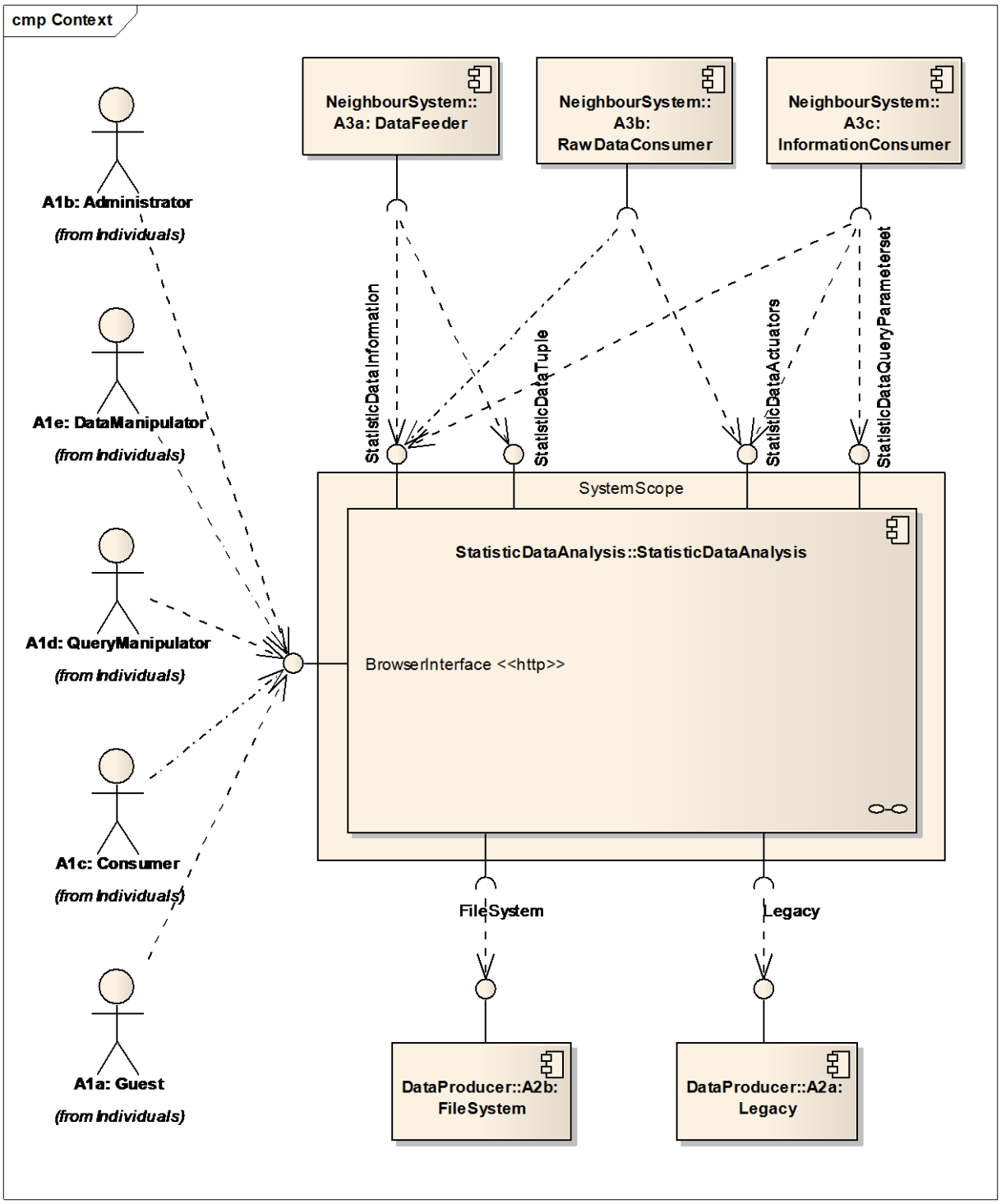


Abbildung 2: Fachlicher Kontext

ID	Name	Beschreibung
P1	BrowserInterface	Typ: Provided Interface, Akteure greifen darauf zu Implementierung: Browser Schnittstelle (Webinterface) Protokoll: http Hauptzweck: Über dieses Interface kann ein Akteur mittels Standard-Webbrowser (z.B. Mozilla Firefox) mit dem System interagieren.
P2	/api/StatisticDataInformation	Typ: Provided Interface, externe Systeme greifen darauf zu Implementierung: REST-API Ressource Protokoll: http Hauptzweck: Über dieses Interface können externe Systeme Information zu weiteren REST-API Ressourcen abfragen.
P3	/api/StatisticDataTuple	Typ: Provided Interface, externe Systeme greifen darauf zu Implementierung: REST-API Ressource Protokoll: http Hauptzweck: Über dieses Interface können externe Systeme neue Daten in das System einspeisen.
P4	/api/StatisticDataActuators	Typ: Provided Interface, externe Systeme greifen darauf zu Implementierung: REST-API Ressource Protokoll: http Hauptzweck: Über dieses Interface können externe Systeme die Aktuatorkenngrössen auslesen.
P5	/api/StatisticDataQueryParameterset	Typ: Provided Interface, externe Systeme greifen darauf zu Implementierung: REST-API Ressource Protokoll: http Hauptzweck: Über diese Schnittstelle können externe Systeme einerseits Query-Parametersets verwalten und andererseits mittels dieser Query-Parametersets Statistik-Daten in Form von Selektionen, Reports oder Alarmierung abfragen.
R1	Legacy	Typ: Required Interface, für Daten-Import benötigt Implementierung: Datenbank-Zugriff Protokoll: TCP/IP Hauptzweck: Über diese Schnittstelle greift das System auf das bestehende Legacy-Data-Repository zu.
R2	FileSystem	Typ: Required Interface, für Entwicklung und Test benötigt Implementierung: .NET-Framework Protokoll: filesystem Hauptzweck: Über diese Schnittstelle werden Daten aus Dateien gelesen.

Eine ausführliche Beschreibung der Programmierschnittstelle folgt in Kapitel 7.

5.2 Technischer Kontext

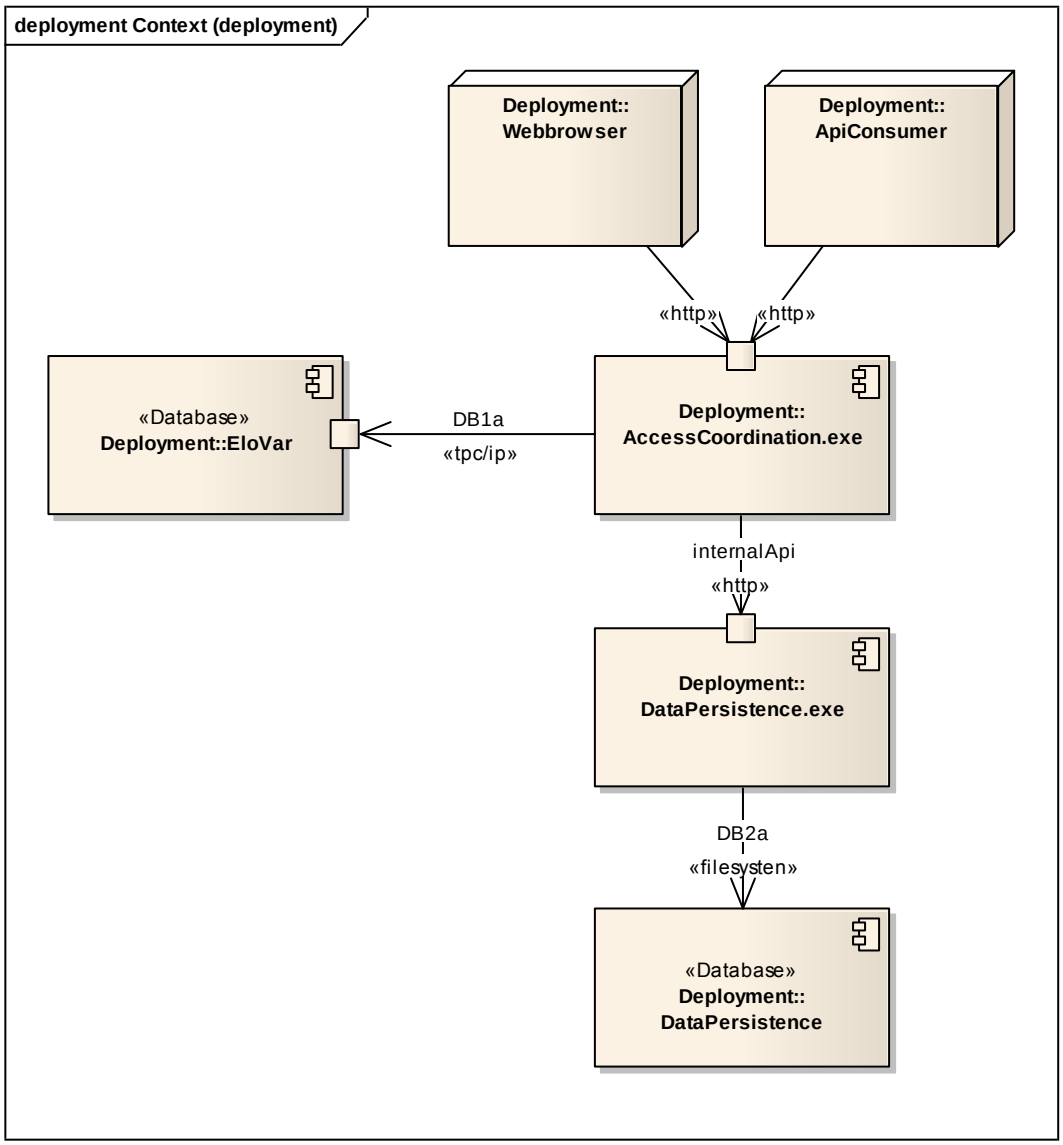


Abbildung 3: Technischer Kontext

Name	Protokoll	Beschreibung
BrowserInterface	http	Verbindung zum Gesamtsystem, die ein Anwender mittels Webbrower (Client) öffnet.
internalApi	http	Verbindung zwischen den beiden unabhängigen Modulen «AccessCoordination» und «DataPersistence». Im Diagramm sind beide «Self-Contained Services» dem gleichen Device zugeteilt. Die Services könnten auch auf eigene Devices verteilt sein.
DB1a	TCP/IP	Verbindung zur externen Legacy-Datenbank, welche auf einem dezidierten Datenbank-Server betrieben wird.
DB1b	Filesystem	Die Daten werden vom «DataPersistence» Modul in einer SQlite Datenbank verwaltet, welche direkt in die Anwendung integriert ist und über Dateisystem-Mechanismen des jeweiligen Betriebssystems verwaltet wird.

Eine ausführliche Beschreibung zur Verteilung der Module folgt in Kapitel 11: Verteilungssicht.

6 Lösungsstrategien

Folgend sind zentrale Strategien und Grundlagen für die Architektur gelistet. Details sind in den nachfolgenden Kapiteln beschrieben und Entscheidungen sind in Kapitel 14 festgehalten.

6.1 Funktionale Lösungen

6.1.1 Aufteilung in «AccessCoordination» und «DataPersistence»-Modul

- Jedes Modul hat einen definierten Aufgabenbereich:
  AccessCoordination: öffentliche Schnittstellen, Zugangsverwaltung, Verwaltung von Benutzerdaten
  DataPersistence: Verwaltung von Statistik-Daten, Generierung von Vergleichsdaten
- Module können unabhängig entwickelt, erweitert, ausgetauscht, betrieben und verteilt werden.

EF01	Lose Kopplung zwischen «AccessCoordination» und «DataPersistence»-Modul

6.1.2 Statistik-Daten-Aktuator als Basis der Datenbank

- Statistik-Daten werden gemäss Businesslogik geordnet, d.h. Betriebskenngrössen sind bereits in der Datenbank dem Aktuator zugeordnet und müssen nicht jeweils wieder bei einer Abfrage nach Seriennummer geordnet werden.
- Ein Statistik-Daten-Aktuator-Modell kann mit Referenz auf neue Daten erweitert werden, wenn in Zukunft weitere Kenngrössen gespeichert werden.

EF02	Änderung des zentralen Datenmodells vom Statistik-Daten-Tupel zum Statistik-Daten-Aktuator

6.1.3 Statistik-Daten-Abfrage-Parameterset können gespeichert und verwaltet werden

In den Anforderungen ist der Ablauf für die Anwendungsfälle UC10, UC11 und UC12 wie folgt beschrieben:

Anwendungsfall	Hauptszenario
UC10	1) Anfrage zur Bekanntgabe einer Statistik-Daten-Selektion durch den Akteur. 2) System verwendet das gesendete Statistik-Daten-Selektions-Parameterset zur Erstellung der gewünschten Statistik-Daten-Selektion. 3) System gibt die Statistik-Daten-Selektion dem Akteur bekannt.
UC11	1) Anfrage zur Bekanntgabe sämtlicher Statistik-Daten-Selektions-Parametersets durch den Akteur. 2) System gibt sämtliche Statistik-Daten-Selektions-Parametersets dem Akteur bekannt.
UC12	1) Eingabe eines Statistik-Daten-Selektions-Parametersets durch den Akteur. 2) System verarbeitet das Statistik-Daten-Selektions-Parameterset. 3) System gibt triviale Empfangsbestätigung dem Akteur bekannt.

Wie im Abschnitt 3.5 erläutert, wurden Statistik-Daten-Selektions-Parameterset, -Report-Parameterset und -Alarm-Parameterset zu Statistik-Daten-Abfrage-Parameterset vereinheitlicht.

Aufgrund der Aufteilung der Anwendung in «DataPersistence» und «AccessCoordination»- Modul werden die Anwendungsfälle UC10, UC11, UC12, UC20 und UC40 wie folgt gelöst:

«DataPersistence»-Modul

Das «DataPersistence»-Modul erfüllt die Anwendungsfälle UC10, UC20 und UC40 wie folgt:

- Es erzeugt aufgrund eines gesendeten Statistik-Daten-Abfrage-Parametersets eine/n passende/n Statistik-Daten-Selektion/Report/Alarm und gibt diese/n zurück.

Das «DataPersistence»-Modul erfüllt die Anwendungsfälle UC10, UC20 und UC40 unter folgenden Bedingungen:

- Es verwaltet keine Statistik-Daten-Abfrage-Parametersets.
- Es hat kein Gedächtnis bezüglich Statistik-Daten-Abfrage-Parametersets.
- Das Statistik-Daten-Abfrage-Parametersets muss bei jeder Anfrage gesendet werden.

«AccessCoordination»-Modul

Das «AccessCoordination»-Modul erfüllt die Anwendungsfälle UC11 und UC12, indem sämtliche Statistik-Daten-Abfrage-Parametersets durch das Modul verwaltet werden.

- Statistik-Daten-Abfrage-Parameterset entgegennehmen (Create)
- Statistik-Daten-Abfrage-Parameterset bekannt geben (Read)
- Änderungen an einem bestehenden Statistik-Daten-Abfrage-Parameterset entgegennehmen (Update)
- Statistik-Daten-Abfrage-Parameterset löschen (Delete)

Das «AccessCoordination»-Modul erfüllt die Anwendungsfälle UC10, UC20 und UC40 wie folgt:

- Es sendet auf Anfrage eines Akteurs ein selektiertes bereits gespeichertes Statistik-Daten-Abfrage-Parameterset an das «DataPersistence»-Modul, nimmt die/den Statistik-Daten-Selektion/Report/Alarm entgegen, und gibt diese/n als Antwort auf die Anfrage an den Akteur zurück.

Um die Statistik-Daten-Abfrage-Parametersets verwalten zu können, wurden die Struktur gegenüber der Anforderungsspezifikation um eine «Identifikation» und eine «Beschreibung» erweitert.

StatisticDataQueryParameterset (Erweiterung)		
Member Name	Typ	Notwendig
StatisticDataQueryParametersetId	numerisch	Nein
Bemerkung: - wird als Identifikation (PK) in der Datenbank verwendet - bei PUT Methoden zwingend - muss bei POST Methoden auf «null» gesetzt werden.		
Description	Text (string)	Ja
Bemerkung: - textuelle Identifikation für Benutzer		

6.1.4 Statistik-Daten-Selektionen für mehrere Aktuatoren

Die Statistik-Daten-Selektion wurde dahingehend erweitert, dass nicht nur eine Statistik-Daten-Selektion auf eine Anfrage zurückgesendet werden kann, sondern eine Liste von Statistik-Daten-Selektionen. Dies gilt auch für Statistik-Daten-Report und Statistik-Daten-Alarmierung.

data (unbenanntes Objekt, dass die Liste kapselt)		
Member Name	Typ	Notwendig
List<StatisticDataSelection>	zusammengesetzt	Ja
Bemerkung: Liste vom Typ «**StatisticDataSelection**»		

6.2 Schnittstellen

6.2.1 http als Kommunikationsprotokoll

- Webinterface und Programmierschnittstelle können das selbe Protokoll verwenden.
- http-Kommunikation ist meist ohne Anpassung von Firewall-Regel möglich.
- Gute Unterstützung für Entwickler (es existieren für alle gängigen Programmiersprachen Bibliotheken).

6.2.2 REST

- Implementation von Ressourcen strukturiert die Schnittstelle.
- Ressourcen können eindeutig angesprochen werden und sind zustandslos:
  z.B. /api/StatisticDataActuators/12345678 ist unveränderlich und liefert immer den Aktuator mit der S/N: 12345678
- Verwendung der http-Verben: GET, POST, PUT, DELETE, Auswertung des http-Status-Codes

6.3 Technologie

6.3.1 .Net, C#, ASP.NET Core 2.0 und Entity Framework Core

Diese Toolchain von Microsoft bietet für sämtliche Problemstellungen eine Lösung und ist sehr gut dokumentiert, zudem ist sie verfügbar.

Informationen: https://docs.microsoft.com/en-us/aspnet/core/

Eine Offline-Version der Dokumentation befindet sich im Projektverzeichnis: 99_Anhang

ET01	Verwendung: ASP.NET Core 2.0 Framework
ET02	Verwendung: Entity Framework Core als OR-Mapper

6.3.2 SQlite als Datenbanksystem

SQlite ist ein relationales Datenbanksystem. Die gesamte Datenbank befindet sich in einer einzigen Datei und wird zusammen mit der Anwendung verteilt.

Informationen: https://sqlite.org/index.html

ET03	SQlite-Datenbanksystem

6.3.3 Verteilung als «Self-Contained Windows Service»

Für die Verteilung wird eine einfache Lösung benötigt, die möglichst wenig Abhängigkeiten zu Drittsystemen hat.

ET04	Verteilung als «Self-Contained Windows Service»

7 Programmierschnittstelle des Gesamtsystems

Es wird nur die Programmierschnittstelle ausführlich dokumentiert. Die Schnittstelle des «BrowserInterface» ist Teil des Webinterface, und es ist nicht Sinn und Zweck diese Schnittstelle explizit anzusprechen. Die Funktionalitäten der Programmierschnittstelle und des Webinterface sind bis auf die Anwendungsfälle UC100 und UC101 deckungsgleich. Für die Anwendungsfälle UC100 und UC101 (Benutzerverwaltung mit Authentifizierung und Autorisierung) gibt es keine äquivalente Programmierschnittstelle.

EF04	Programmierschnittstellen implementiert keine Authentifizierung und Autorisierung

Die aktuellste und akkurateste Beschreibung liefert das System selbst. Die Programmierschnittstelle wird mittels Swagger (https://swagger.io) jeweils bei der Kompilierung des Moduls aus dem Quellcode erzeugt. Man kann die Online-Dokumentation unter folgender URI abrufen: https://[authority]/swagger

7.1 REST-Ressourcen

Es gibt Ressourcen, die implementiert wurden, die aber in keinem Anwendungsfall explizit gefordert wurden. So ist es z.B. möglich ein Statistik-Daten-Abfrage-Parameterset zu löschen, oder sämtliche Statistik-Daten-Tupel abzurufen, die nicht synchronisiert wurden.

Nachfolgende sind sämtliche REST-Ressourcen und der dazugehörige Anwendungsfall gelistet.

7.1.1 StatisticDataActuators

Methode	Ressource	Anwendungsfall	Akteur
GET	/api/StatisticDataActuators	UC50	A1c, A3b
GET	/api/StatisticDataActuators/{id}	UC50	A1c, A3b

7.1.2 StatisticDataInformation

Methode	Ressource	Anwendungsfall	Akteur
GET	/api	UC60	A3a, A3b, A3c

7.1.3 StatisticDataQueryParameterset

Methode	Ressource	Anwendungsfall	Akteur
GET	/api/StatisticDataQueryParametersets	UC11	A1c, A3c
POST	/api/StatisticDataQueryParametersets	UC12	A1d
GET	/api/StatisticDataQueryParametersets/{id}	UC11	A1c, A3c
PUT	/api/StatisticDataQueryParametersets/{id}	UC12	A1d
DELETE	/api/StatisticDataQueryParametersets/{id}		
GET	/api/StatisticDataQueryParametersets/Selection/{id}	UC10	A1c, A3c

GET	/api/StatisticDataQueryParametersets/Report/{id}	UC20	A1c, A3c
GET	/api/StatisticDataQueryParametersets/Alarm/{id}	UC40	A1c, A3c

7.1.4 StatisticDataTuple

Methode	Ressource	Anwendungsfall	Akteur
GET	/api/StatisticDataTuple		
POST	/api/StatisticDataTuple	UC30	A1e, A3a
GET	/api/StatisticDataTuple/{id}	UC31	A1e

7.2 Modelle

Es gibt folgende zwei Modelle, die mit einer POST, oder PUT Methode übermittelt werden können:

- Statistik-Daten-Tupel: «StatisticDataTuple»
- Statistik-Daten-Abfrage-Parameterset: «StatisticDataQueryParameterset»

Auf eine Daten-Anfrage antwortet das System entweder mit einem Statistik-Daten-Aktuator oder aber einer Statistik-Daten-Selektion / -Report oder -Alarmierung:

- Statistik-Daten-Aktuator: «StatisticDataActuator»
- Statistik-Daten-Selektion: «StatisticDataSelection»
- Statistik-Daten-Report: «StatisticDataReport»
- Statistik-Daten-Alarmierung: «StatisticDataAlarm»

7.2.1 Benutzte Modelle (für Anfrage) bei Anwendungsfällen

Anwendungsfall	Modell	
	Statistik-Daten-Abfrage-Parameterset	Statistik-Daten-Tupel
UC12: Statistik-Daten-Abfrage-Parameterset entgegennehmen		
UC30: Statistik-Daten-Tupel entgegennehmen		
UC31: Modifiziertes Statistik-Daten-Tupel entgegennehmen		

7.2.2 Benutzte Modelle (für Antworten) bei Anwendungsfällen

Anwendungsfall	**Modell**			
	Statistik-Daten-Aktuator	Statistik-Daten-Selektion	Statistik-Daten-Report	Statistik-Daten-Alarmierung
UC50: Statistik-Daten-Aktuator bekannt geben				
UC10: Statistik-Daten-Selektion bekannt geben				
UC20: Statistik-Daten-Report bekannt geben				
UC40: Statistik-Daten-Alarmierung bekannt geben				

7.2.3 Aufbau: Statistik-Daten-Tupel / «StatisticDataTuple»

StatisticDataTuple		
Member Name	Typ	Notwendig
pKey	UInt32	Ja
Klasse mit primitiven Attributen vom Typ: Integer, String, Datum Die Anzahl und der jeweilige Typ der Attribute kann dem Anhang 16.1 entnommen werden.		

7.2.4 Aufbau: Statistik-Daten-Abfrage-Parameterset / «StatisticDataQueryParameterset»

StatisticDataQueryParameterset		
Member Name	Typ	Notwendig
StatisticDataQueryParametersetId	numerisch	Nein
Bemerkung: - wird als Identifikation (PK) in der Datenbank verwendet - bei PUT Methoden zwingend - muss bei POST Methoden auf «null» gesetzt werden.		
Description	Text (string)	Ja
Bemerkung: - textuelle Identifikation für Benutzer		
SelectedStatisticDataTupleNumericAttributes	zusammengesetzt	Ja
Bemerkung: Liste vom Typ «SelectedStatisticDataTupleNumericAttribute» (siehe unten)		
ConditionSetOperator	Enumeration	Ja
Bemerkung: - gibt an, ob die «Conditions» (Bedingungen) als Schnitt- (AND) oder Vereinigungsmenge (OR) behandelt werden		

Conditions	zusammengesetzt	Ja
Bemerkung: - Liste vom Typ «Condition» (siehe unten)		
StartDate	Datum	Nein
Bemerkung: - Startdatum eines Auswahlbereichs		
EndDate	Datum	Nein
Bemerkung: - Enddatum eines Auswahlbereichs		

SelectedStatisticDataTupleNumericAttribute		
Member Name	Typ	Notwendig
StatisticDataTupleNumericAttribute	Enumeration	Ja
Bemerkung: - eines von 23 möglichen numerischen Attributen eines Statistik-Daten-Tupels		
IncreaseByPeriodType	Enumeration	Nein
Bemerkung: - gibt an, ob der folgende Wert das Wachstum pro Tag, Woche, Monat oder Jahr angibt - mögliche Werte sind: DAY, WEEK, MONTH, YEAR (vgl. «IncreaseByPeriod»)		
IncreaseByPeriod	numerisch	Nein
Bemerkung: - Wachstum pro Tag, Woche, Monat oder Jahr (vgl. «IncreaseByPeriodType»)		
InitialValue	numerisch	Nein
Bemerkung: - Startwert der Wachstumskurve - falls keine Angaben, dann 0		
AlarmLimit	numerisch	Nein
Bemerkung: - konstante Limite für Alarmierung		
AlarmCompareOperator	Enumeration	Nein
Bemerkung: - gibt an, ob der Alarm bei Über- oder Unterschreitung der Limite ausgelöst wird (vgl. AlarmLimit)		
AlarmNotificationEmail	Text (string)	Nein
Bemerkung: - E-Mail-Adresse, die alarmiert wird bei Über- oder Unterschreitung der Limite		
AlarmNotificationEnabled	Boolean	Nein
Bemerkung: - Alarmierung kann ein- oder ausgeschaltet sein		

Condition		
Member Name	Typ	Notwendig
StatisticDataTupleQueryAttribute	Enumeration	Ja
Bemerkung: - eines von 9 möglichen Abfrage-Attributen eines Statistik-Daten-Tupels - mittels dieser Attribute werden die Seriennummern bestimmt, die für die Daten-Auswahl in Frage kommen. - mehrere «Condition» können mittels «ConditionSetOperator» eingeschränkt oder vereinigt werden. (vgl. «ConditionSetOperator»)		

CompareOperator	Enumeration	Ja
Bemerkung: - gibt an, wie der gespeicherte Wert des «StatisticDataTupleQueryAttribute» im Verhältnis zum «CompareValue» steht (vgl. «StatisticDataTupleQueryAttribute», «CompareValue») - mögliche Werte: ==, !=, >=, <=, >, <		
CompareValue	Text (string)	Ja
Bemerkung: - Vergleichswert zum mittels «StatisticDataTupleQueryAttribute» abgefragten Wertes - ist vom Typ «Text», wird allenfalls vom System in einen numerischen Wert gewandelt		

7.2.5 Statistik-Daten-Aktuator: «StatisticDataActuator»

Das Statistik-Daten-Aktuator-Modell hat als zentrales Identifikationsmerkmal die Seriennummer «Serial», welche auch als Schlüssel dient und unveränderlich ist. Weitere unveränderliche Daten sind in der Struktur «ProductionData» zusammengefasst.

Sämtliche weiteren Werte sind primitive Datentypen, welche zeitlich und thematisch zu Strukturen zusammengefasst sind und mit einem Zeitstempel versehen werden. Solche Strukturen werden als Listen unterschiedlicher Länge zusammengefasst.

(nicht öffentlich)

7.2.6 Statistik-Daten-Selektion: «StatisticDataSelection»

data (unbenanntes Objekt, dass die Liste kapselt)		
Member Name	Typ	Notwendig
List<StatisticDataSelection>	zusammengesetzt	Ja
Bemerkung: Liste vom Typ «**StatisticDataSelection**» (siehe unten)		

StatisticDataSelection		
Member Name	Typ	Notwendig
Serial	numerisch	Ja
Bemerkung: Seriennummer des Aktuators		
SingleAttributes	zusammengesetzt	Ja
Bemerkung: - Liste vom Typ «SingleAttribute» (siehe unten)		

SingleAttribute		
Member Name	Typ	Notwendig
NumericAttribute	Enumeration	Ja
Bemerkung: - eines von 23 möglichen numerischen Attributen eines Statistik-Daten-Tupels		
ResultTuples	zusammengesetzt	Ja
Bemerkung: - Liste vom Typ «ResultTuple» (siehe unten)		

ResultTuple		
Member Name	Typ	Notwendig
DatapointDate	Datum	Ja
Bemerkung: - Zeitstempel		
ValueDevice	numerisch	Ja
Bemerkung: - Zahlenwert vom Typ «NumericAttribute» mit Zeitstempel		

7.2.7 Statistik-Daten-Report: «StatisticDataReport»

Ein Statistik-Daten-Report ist analog zu einer Statistik-Daten-Selektion (vgl. 0), der Unterschied besteht im Typ «ResultTuple».

ResultTuple		
Member Name	Typ	Notwendig
DatapointDate	Datum	Ja
Bemerkung: - Zeitstempel		
ValueDevice	numerisch	Ja
Bemerkung: - Zahlenwert vom Typ «NumericAttribute» zum Zeitpunkt des «DatapointDate»		
ValueCompare		
Bemerkung: - berechneter Vergleichswert zum Zeitpunkt des «DatapointDate»		

7.2.8 Statistik-Daten-Alarmierung: «StatisticDataAlarm»

Eine Statistik-Daten-Alarmierung ist analog zu einer Statistik-Daten-Selektion (vgl. 0). Der Unterschied ist nicht das Format, sondern dass nur Werte zurückgeliefert werden, die über-/unter dem Limit sind.

8 Typische Strukturen und übergreifende Konzepte

8.1 REST

Ziel: Die Schnittstellen sollen REST-konform sein

Es wird an dieser Stelle nicht beschrieben was REST-konform bedeutet. Information dazu findet man unter anderem in folgenden Quellen:

REST und HTTP, 3. Auflage, Tilkov, Eigenbrodt, Schreier, Wolf [16]

https://de.wikipedia.org/wiki/Representational_State_Transfer

https://www.restapitutorial.com

Um das Ziel zu erreichen, wurde versucht die Schnittstellen so zu implementieren, dass sie folgende Stufen erfüllen.

Quelle: https://www.martinfowler.com/articles/richardsonMaturityModel.html

	Bezeichnung	**Bemerkung**
Stufe 0	http als Protokoll	• Ein wesentliches Merkmal von REST ist es, dass Technologien des World Wide Webs verwendet werden.
Stufe 1	Ressourcen	• Informationen und Informationsgruppen haben eindeutige Adressen. • Anfragen eines Clients an den Server sollen sämtliche Information über den Zustand beinhalten. • Zugriff auf eine Ressource sollte immer das gleiche Ergebnis liefern: → Zustandslosigkeit
Stufe 2	http Methoden	• http Methoden und http Status Codes sind standardisiert. • Die Wahl der Methode hat Einfluss auf die Behandlung der Anfrage, z.B. gilt für GET, dass keine Veränderung vorgenommen wird: → Zustandslosigkeit
Stufe 3	Hypermedia Controls	HATEOAS (Hypertext As The Engine Of Application State) • Antworten werden mit Links erweitert, die aufzeigen, welche möglichen weiteren Anfragen die Anwendung im momentanen Zustand entgegennehmen kann.

8.2 HATEOAS (Hypertext As The Engine Of Application State)

Bei HATEOAS geht es darum, dass die Anwendung als Zustandsmaschine modelliert wird, und man mittels Links die Zustände wechselt. Eine Antwort enthält darum Informationen, in welche Zustände man als Nächstes übergehen kann. Ein Konsument benötigt daher in der Theorie kein Wissen über die Schnittstelle, sondern kann aufgrund der jeweils erhaltenen Informationen von einem in den nächsten möglichen Zustand übergehen.

Eine Maschine kann so, ähnlich wie ein Mensch im World Wide Web, durch eine Anwendung surfen, ohne dabei auf Vorwissen angewiesen zu sein.

EF05	Programmierschnittstellen um HATEOAS Links erweitern

8.2.1 Vergleich der System-Schnittstellen

Nachfolgend sind die Zustände des Webinterfaces der «Statistik-Daten Auswertung» dargestellt. Öffnet man die Anwendung, startet man, wie bei den meisten anderen Webanwendungen, auf der Startseite. Der Benutzer bekommt dort die Möglichkeit zur gewünschten Funktion zu navigieren und weitere Zustände zu durchlaufen.

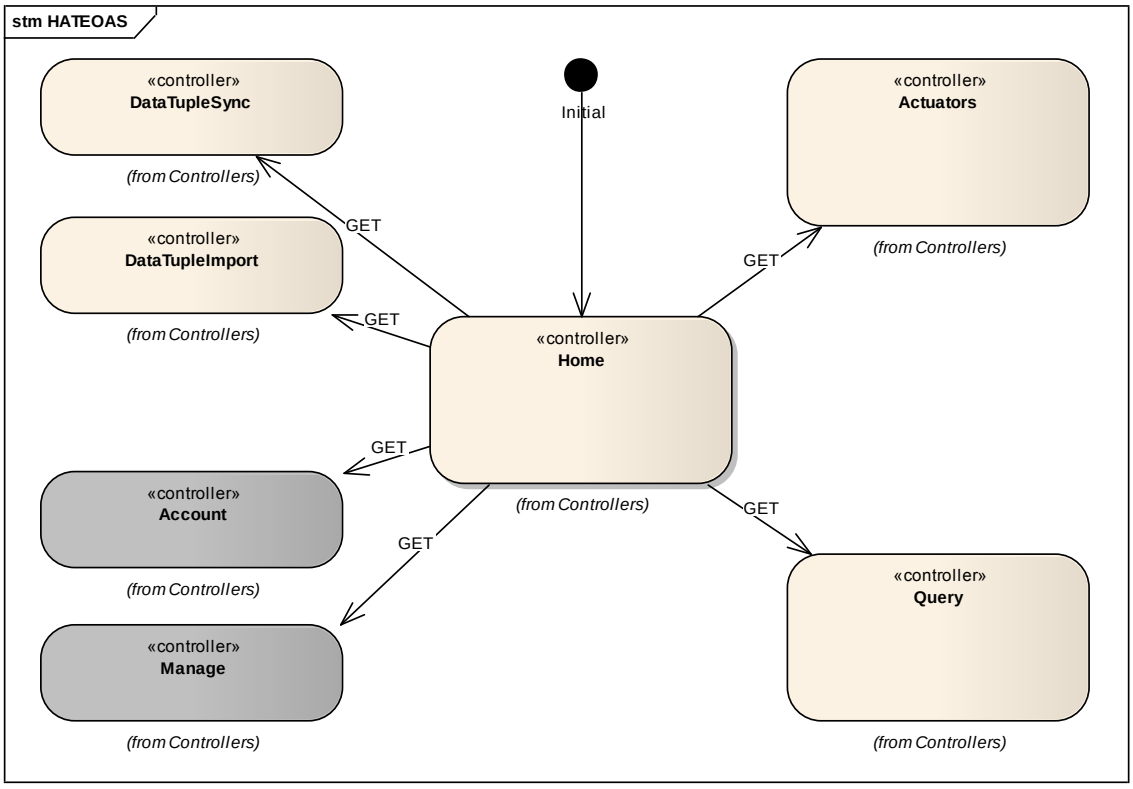


Abbildung 4: Zustandsmaschine des Webinterfaces

Dasselbe gilt auch für die Programmierschnittstelle der «Statistik-Daten Auswertung». Eine Übersicht der Möglichkeiten in Form von Links erhält man, wenn man eine GET Anfrage auf die Ressource «/api» macht.

Da es bei der «Statistik-Daten Auswertung» Anwendung ein Hauptanliegen ist, dass Menschen und Maschinen als Akteure auftreten können, ist die Zustandsmaschine für die Programmierschnittstelle der Zustandsmaschine des Webinterfaces sehr ähnlich.

In der momentanen Implementation wird aber bei der Programmierschnittstelle auf Authentifizierung und Autorisierung verzichtet, daher fehlen Pendants für «Account» und «Manage».

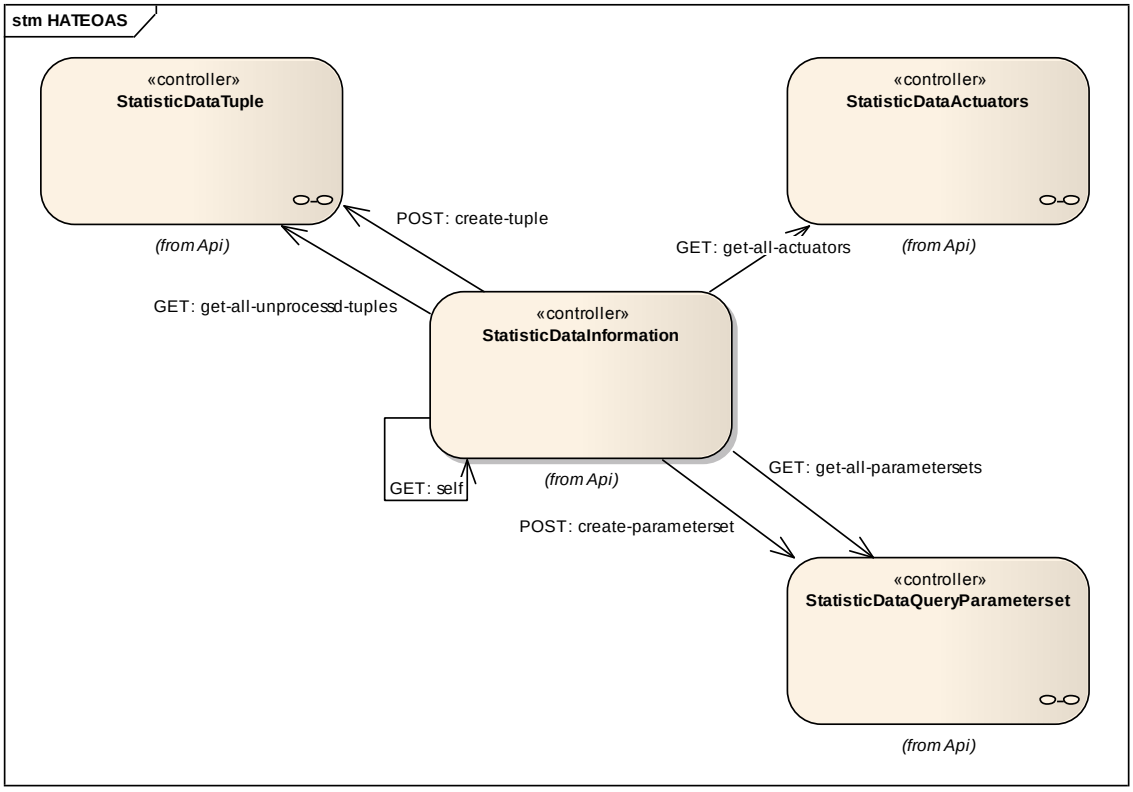


Abbildung 5: Zustandsmaschine der Programmierschnittstelle

Das folgende Diagramm zeigt beispielhaft die Zustandsmaschine einer einzelnen Ressource. Das Navigieren zurück in die Hauptebene wird nicht dargestellt, da es implizit gegeben ist. Sämtliche Diagramme der Zustandsmaschine sind einerseits im Kapitel 10.2 zu finden oder aber im Package «10. Runtime View» des «Enterprise Architect Projekts» [5].

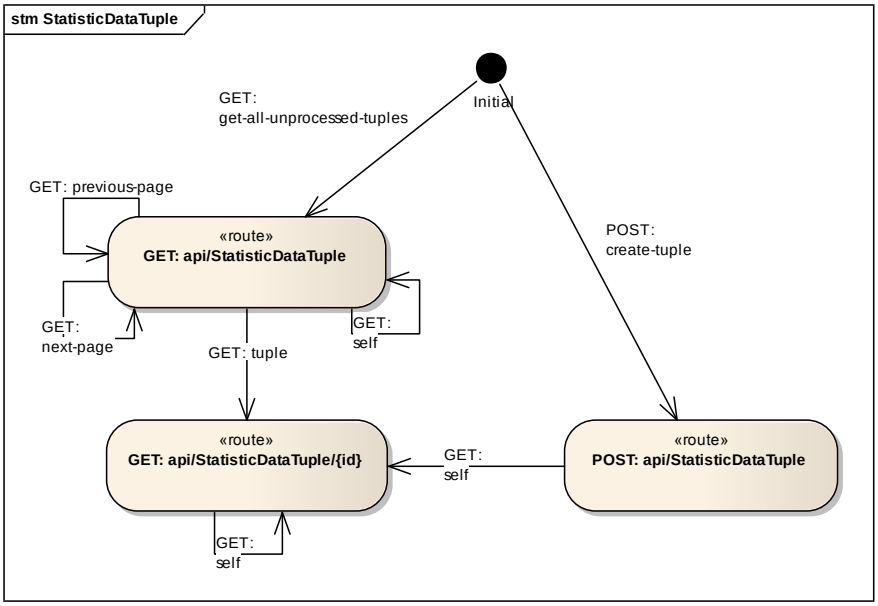


Abbildung 6: Zustandsmaschine der Ressource «/api/StatisticDataTuple»

8.2.2 Implementierung

Mittels «HateoasLinkHelper»-Klasse können jegliche Arten von Objekten gekapselt und mit Links versehen werden, bevor sie an den Konsumenten versendet werden. Der Konsument entpackt die Nutzdaten und verwendet die Link-Information für seine nächste Aktion.

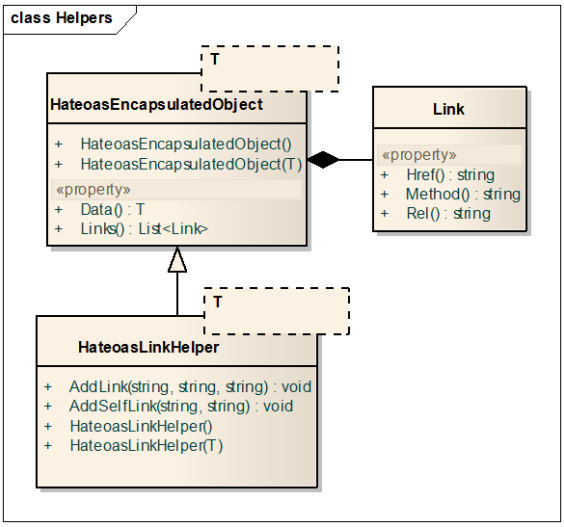


Abbildung 7: Klassendiagramm des «HateoasLinkHelpers»

8.2.3 Interne Schnittstelle

Auch die interne Schnittstelle zwischen «AccessCoordination»- und «DataPersistence»-Modul ist nach demselben Prinzip entwickelt worden. Auch hier werden aus Design-Gründen die Objekte gekapselt, allenfalls müsste dieser Daten-Overhead aber zu einem späteren Zeitpunkt nochmals neu beurteilt werden.

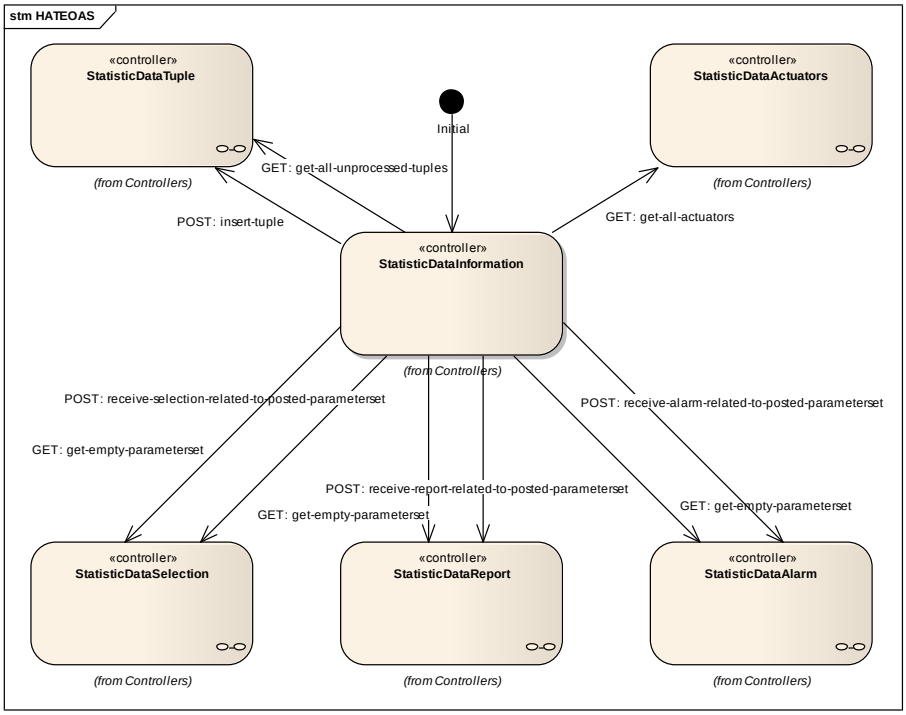


Abbildung 8: Zustandsmaschine der internen Programmierschnittstelle

Bei der Implementierung informiert sich das «AccessCoordination»-Modul nach einem Neustart als Erstes bei der «/internalApi» Ressource über die weiteren Möglichkeiten. Dabei versucht das System die Ressourcen dynamisch zuzuordnen. Einer Änderung der Schnittstellen-Benennung auf Seiten des «DataPersistence»-Moduls könnte also durch einen einfachen Neustart des «AccessCoordination»-Moduls beigekommen werden. Das System kommt aber an seine Grenzen, sobald Ressourcen und Funktionalitäten nicht mehr gleichbehandelt werden. Allgemein gilt, dass es für HATEOAS nicht genügt, Links hinzuzufügen, sondern ein Konsument muss auch in der Lage sein, diese zu interpretieren und sich dann zum nächsten Zustand zu begeben.

8.3 Datenpersistenz in der Datenbank

Für die Anwendung gilt, dass Zustände in Datenbanken gespeichert werden (Database Session State [10]). Zum Beispiel gibt es in der Datenbank «AccessCoordinationDb» die Tabelle «TemporaryStatisticDataTuples», in welcher sämtliche Statistik-Daten-Tupel verwaltet werden, die importiert aber noch nicht akzeptiert wurden. Temporäre Daten könnten auch im Memory des Webservers (Server Session State [10]) oder aber als Cookies im Browser (Client Session State [10]) gespeichert werden. Der Vorteil von Datenbanken liegt aber darin, dass der Session State nicht an den Client gebunden ist. So können temporäre Daten in einer ersten Session importiert und bei einem nächsten Login bearbeitet werden.

8.4 Vorgaben von ASP.NET Core 2.0

Aufgrund der Wahl des Frameworks werden viele typische Strukturen und Konzepte durch das Framework vorgegeben. Diese Vorgaben sollten auch so angewendet werden. Nachfolgende Aufzählung weist auf Vorgaben hin, die bei der Architektur berücksichtig wurden, hat aber keinen Anspruch auf Vollständigkeit. Es sei hier auf die Dokumentation von Microsoft verwiesen:

https://docs.microsoft.com/en-us/aspnet/core/

Eine Offline-Version der Dokumentation befindet sich im Projektverzeichnis: 99_Anhang

ET01	Verwendung: ASP.NET Core 2.0 Framework

8.4.1 «Konventionen vor Konfiguration»

Dieses Architektur-Paradigma ist fester Bestandteil von ASP.NET Core 2.0. Dieses Paradigma ist vor allem massgebend beim Routing von ASP.NET MVC Anwendungen.

Zum Beispiel der Klassen-Name «DataTupleImportController» wird zum Routing-Endpunkt «DataTupleImport», die Methoden-Namen der Klasse erweitern den Pfad.

Als Beispiel folgt die Klasse «DataTupleImportController» und ihre Methoden:

Methode	**http**	**Routing-Endpunkt**
Index()	GET	/DataTupleImport/
Details(uint? id)	GET	/DataTupleImport/Details/5
Create()	GET	/DataTupleImport/Create
Create(StatisticDataTuple statisticDataTuple)	POST	/DataTupleImport/Create
Edit(uint? id)	GET	/DataTupleImport/Edit/5
Edit(uint? id, StatisticDataTuple statisticDataTuple)	POST	/DataTupleImport/Edit/5
AcceptAll(string currentFilter)	GET	/DataTupleImport/AcceptAll

Accept(uint? id)	GET	/DataTupleImport/Accept/5
Accept(uint id, StatisticDataTupleDbModel statisticDataTupleDbModel)	POST	/DataTupleImport/Accept/5
RejectAll(string currentFilter)	GET	/DataTupleImport/RejectAll
Reject(uint? id)	GET	/DataTupleImport/Reject/5
Reject(uint id, StatisticDataTupleDbModel statisticDataTupleDbModel)	POST	/DataTupleImport/Reject/5
Import()	GET	/DataTupleImport/Import
Import(List<IFormFile> files)	POST	/DataTupleImport/Import

8.4.2 «Model View Controller»-Muster

Beim typischen MVC Muster wird die «View» mittels «Observer»-Pattern benachrichtigt, wenn das «Model» sich ändert. Dieses Konzept funktioniert bei Web-Anwendungen nicht, da aufgrund der Client-Server-Architektur das «Model» die «View» nicht notifizieren kann. (vgl. Dokumentation Microsoft)

https://docs.microsoft.com/en-us/aspnet/core/mvc/overview

Bei Web-Anwendung empfängt der «Controller» eine Anfrage «Request» und stösst damit die Verarbeitung der Business-Logik an. Die Business-Logik setzt sich zusammen aus «Model» (Daten) und «Service» (Logik). Falls für die Bearbeitung der Anfrage auf persistente Daten zugegriffen werden muss, greift die Business-Logik auf eine Datenbank «Data Source» zu.

Das Resultat der Anfrage («Request) wird vom «Controller» in Form eines «ViewModel» an die «View» übergeben, welche ihrerseits den Inhalt des «ViewModel» als Antwort («Response») dem Benutzer anzeigt.

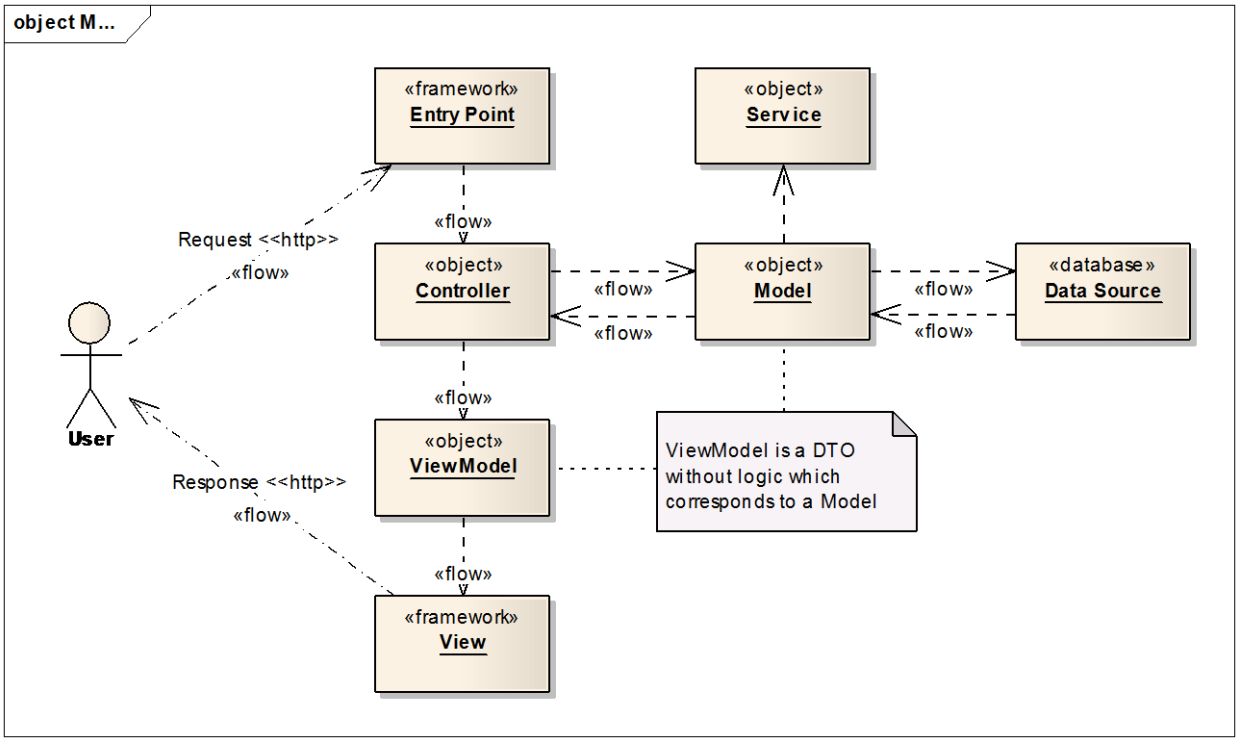


Abbildung 9: MVC Muster in Web-Anwendungen

Die Objekte «Controller», «Model», «Service», «Database» und «ViewModel» sind nach diesem Muster implementiert und auch als solches als «Namespaces» im Quellcode zusammengefasst.

Ein «Entry Point» (oder auch Front Controller) wird bei ASP.NET als “Middleware” durch das Framework implementiert und bedient mehrere Kontroller. Verschiedene «Middleware» Stufen funktionieren wie eine Pipeline.

Eine Anfrage durchläuft mehrere Stufen, bevor sie an einen «Controller» übergeben wird. Es können auch benutzerspezifische «Middleware»-Elemente selber hinzugefügt werden.

Das Konzept ist ausführlich in der Dokumentation von Microsoft beschrieben:

https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/

Die «View» wird wiederum dynamisch durch das Framework erzeugt. Das Template der Darstellung wird als cshtml-Datei im Ordner «Views» gespeichert. Auch hier gilt «Konventionen vor Konfiguration».

Als Beispiel folgt die Klasse «DataTupleImportController» und ihre Methoden mit passender View:

Methode	http	Routing-Endpunkt	View-Pfad
Index()	GET	DataTupleImport/	\Views\DataTuple\Import\Index.cshtml
Details(uint? id)	GET	DataTupleImport/Details/5	\Views\DataTuple\Import\Details.cshtml
Create()	GET	DataTupleImport/Create	\Views\DataTuple\Import\Create.cshtml
Edit(uint? id)	GET	DataTupleImport/Edit/5	\Views\DataTuple\Import\Edit.cshtml
Accept(uint? id)	GET	DataTupleImport/Accept/5	\Views\DataTuple\Import\Accept.cshtml
Reject(uint? id)	GET	DataTupleImport/Reject/5	\Views\DataTuple\Import\Reject.cshtml
Import()	GET	DataTupleImport/Import	\Views\DataTuple\Import\Import.cshtml

8.4.3 «Model View Controller» Muster für Programmierschnittstellen

Das Muster für Programmierschnittstelle entspricht weitgehendst dem MVC Muster. Der einzige Unterschied ist, dass es ein «ApiModel» anstatt des «ViewModel» gibt, und die «View» entfällt.

Im Fall einer Anfrage an eine API Ressource liefert der Kontroller ein «ApiModel» an die Middleware. Die Middleware wiederum erzeugt keine «View», sondern sendet die Daten in Form eines serialisierten Objects an den Benutzer.

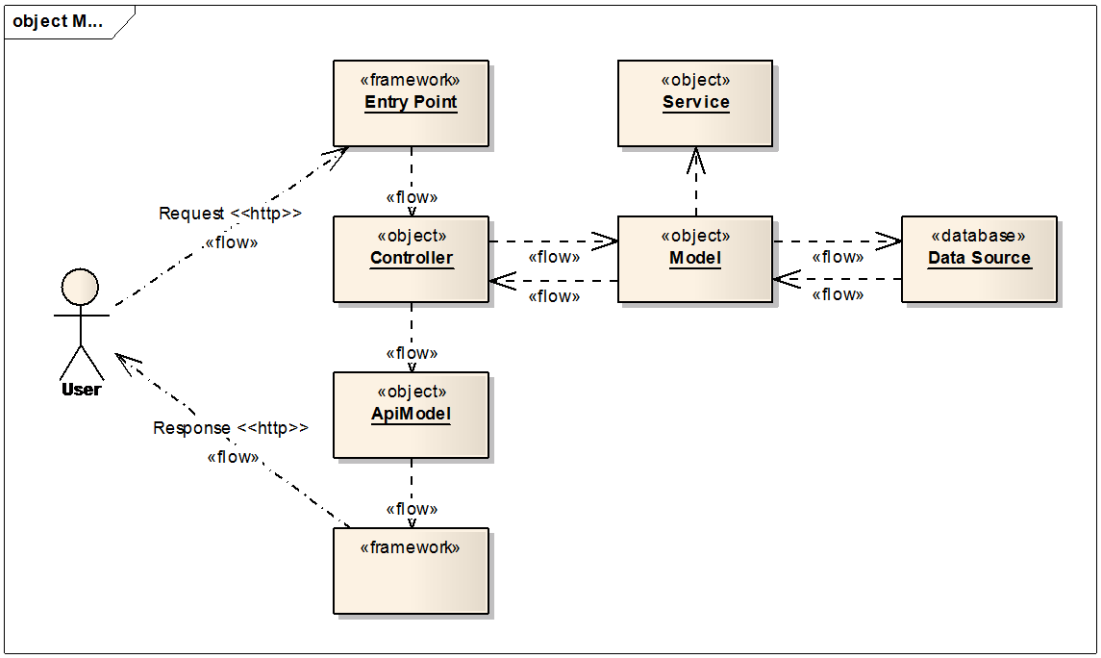


Abbildung 10: Muster für eine Programmierschnittstelle

8.4.4 Dependency Injection

Ein wichtiges Konzept, dass vom ASP.NET Framework vorgegeben wird, ist die «Dependency Injection». Dieses Muster wird vor allem bei der Implementierung von «Controller» verwendet. Dabei werden benötigte Abhängigkeiten z.B. «Service» nicht durch den «Controller» selbst initialisiert, sondern mittels «Dependency Injection» übergeben.

https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection

«Services» die mittels «Dependency Injection» zur Verfügung gestellt werden, müssen dabei bei der Programminitialisierung beim Framework («ServiceProvider») registriert werden, zudem muss angegeben werden, ob ein «Service» permanent und einmalig existiert «Singleton» oder aber zeitlich begrenzt ist «Transient». Meistens werden die Abhängigkeiten direkt in den Konstruktor des «Controller» injiziert.

Auszug Registrierung eines «Service» in der Datei «Startup.cs»:

```
services.AddTransient<StatisticDataTupleService, StatisticDataTupleService>();
```

Vorteil der «Dependency Injection» ist, dass Abhängigkeiten zentral verwaltet werden, und die Laufzeit sowie das Starten und Beenden des Threads durch das Framework verwaltet werden. Dies erleichtert die Entwicklung einer Anwendung, bei der mehrere Anfragen quasi-gleichzeitig beantwortet werden müssen. Zudem können «Controller» für Unittests besser isoliert werden, indem bei Tests ein Mock-Objekt injiziert wird.

Folgendes Sequenzdiagramm zeigt eine Anfrage an das «DataPersistence»-Modul, das sämtliche Aktuatoren zurückliefert. Die Abhängigkeiten werden dabei vom «Dependency Injection» Framework erzeugt.

REST-Ressource: GET /internalApi/StatisticDataActuators

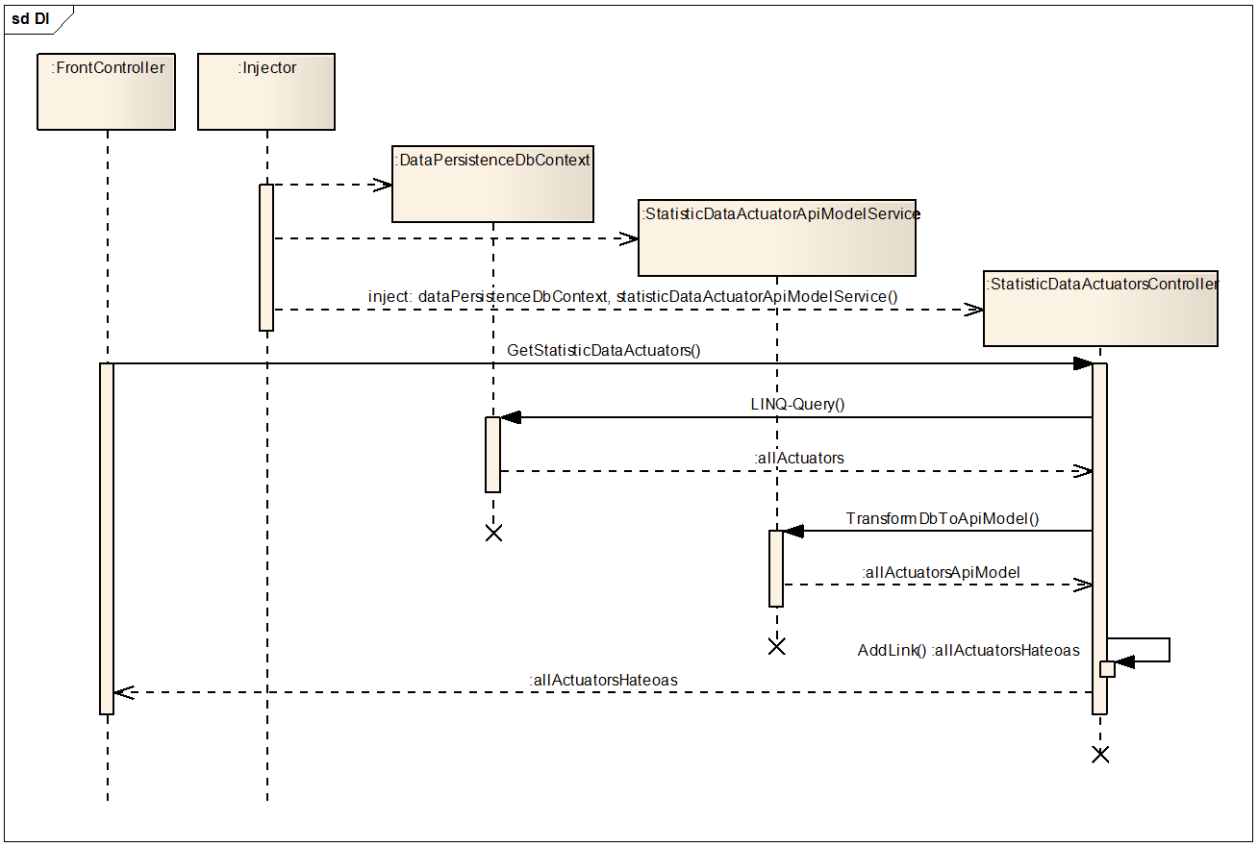


Abbildung 11: Aufruf der Methode «GetStatisticDataActuators()»

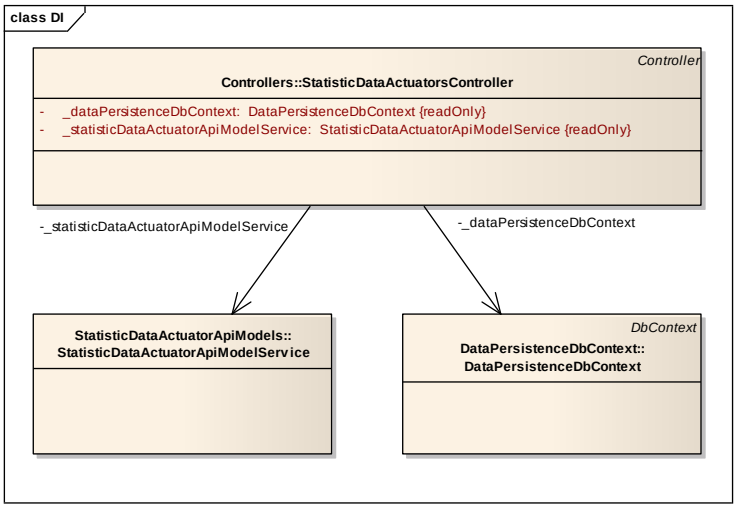


Abbildung 12: Klassen-Diagramm des «StatisticDataActuatorsControllers»

8.4.5 LINQ / Entity Framework Core

ET02	Verwendung: Entity Framework Core als OR-Mapper

Für die Benutzung und Anbindung einer Datenbank an ASP.NET Core 2.0 eignet sich das passende Entity Framework Core (EF Core). Dieser OR-Mapper erlaubt es dem Entwickler direkt mittels LINQ auf die Datenbank zuzugreifen. Für den Entwickler hat das den Vorteil, dass die Datenbank weitgehendst abstrahiert ist, und er sich nicht weiter darum kümmern muss. Selbst die Erzeugung des Datenbankschemas entfällt bei einem «Code-First»-Ansatz. Dafür müssen die Objekt-Modelle aber mit Referenzen versehen werden. Aus diesem Grund gibt es im Quellcode neben «ViewModel» und «ApiModel» auch das «DbModel».

Die Spracherweiterung LINQ für den einfachen Datenzugriff ist ein Bestandteil von .NET und versucht eine einheitliche Methode für jeglichen Datenzugriff zur Verfügung zu stellen.

Für die Architektur hat es den Vorteil, dass sämtliche Datenbanken Typen verwendet werden können, für welche ein Provider existiert. Es müssen nicht zwingend SQL-Datenbanken sein. So ist z.B. die .NET In-Memory-Datenbank eigentlich nicht vollständig SQL kompatibel, wird dank ihrer Geschwindigkeit aber trotzdem für Unittests verwendet, da keine direkten SQL-Abfragen im Projekt verwendet werden.

Folgende Tabelle zeigt eine Übersicht der verwendeten Datenbanken und in welcher Umgebung sie verwendet werden. Die Wahl der verwendeten Datenbank muss dabei nicht zur Kompilationszeit erfolgen, sondern kann per Konfigurationsfile bestimmt werden.

Eingesetzte Datenbanken

Datenbank	Produktiv	Systemtests	Unittests
AccessCoordinationDb	sqlite	inmemory-sqlite	inmemory
EloVarDb	mysql	inmemory	inmemory
DataPersistenceDb	sqlite	inmemory-sqlite	inmemory

Typ der Datenbank

config-switch	Typ	Bemerkung
sqlite	SQlite Filesystem Datenbase	DBMS mit minimalen externen Abhängigkeiten wird für Neuentwicklungen produktiv eingesetzt
mysql	Oracle MySQL Database	DBMS von Oracle Legacy-System beruht auf diesem DBMS
inmemory-sqlite	In-Memory SQlite Database	DBMS Simulation mit dem gleichen SQL-Umfang wie SQlite schneller zum Auf-/Absetzen als reales SQlite-DBMS
inmemory	.NET Core In-Memory Database	DBMS Simulation mit eingeschränktem SQL-Umfang sehr schnell zum Auf-/Absetzen

8.4.6 Konfiguration in Dateien

Die Konfiguration der Module erfolgt mittels jeweiliger Konfigurations-Datei. Dabei handelt es sich um eine Textdatei im JSON-Format, welche im selben Verzeichnis wie die ausführbare Datei abgelegt sein muss.

Dabei gelten folgende Namens-Konventionen:

Umgebung	Konfigurationsdatei
Produktiv	appsettings.json
Entwicklung	appsettings.Development.json
Integrationstests	appsettings.IntegrationTest.json
Systemtests	appsettings.SystemTest.json

Die Dateien für Entwicklung und Tests erweitern dabei jeweils die «appsettings.json» Datei inkrementell. Die Konfiguration kann dabei mit Werten ergänzt werden, oder aber Werte ersetzen bestehende Werte. Besteht in der «appsettings.{environment}.json» Datei kein Eintrag, wird der Wert aus «appsettings.json» unverändert übernommen. Achtung: Konfigurationsänderungen benötigen einen Neustart der Anwendung!

8.4.7 Log-Daten

Um die Anwendung im Betrieb zu überwachen wird der Logging-Mechanismus des ASP.NET Framework verwendet. Das Framework bietet keinen Mechanismus um die Daten direkt in eine Datei zu schreiben, darum wurde das externe Paket «Serilog» verwendet um den Logging-Mechanismus zu erweitern.

https://github.com/serilog/serilog-aspnetcore

Die Logging Einstellungen sind in den «appsettings.json» Dateien festgelegt. Pro Tag wird eine Datei erstellt, die sich in den jeweiligen Unterordner «DataPersistenceLogs» und «AccessCoordinationLogs» befinden.

9 Bausteinsicht

9.1 Übersicht (Level 1)

Die wesentlichen drei Bestandteile des Systems «Statistik-Daten Auswertung» sind «AccessCoordination», «DataPersistence» und «EloVar Datenbank». Das «AccessCoordination» und «DataPersistence»-Modul bilden zusammen die neuentwickelte Anwendung. Wie in der Kontextabgrenzung (vgl. Kapitel 5) erläutert, wird die «EloVar Datenbank» über die Legacy-Schnittstelle angebunden.

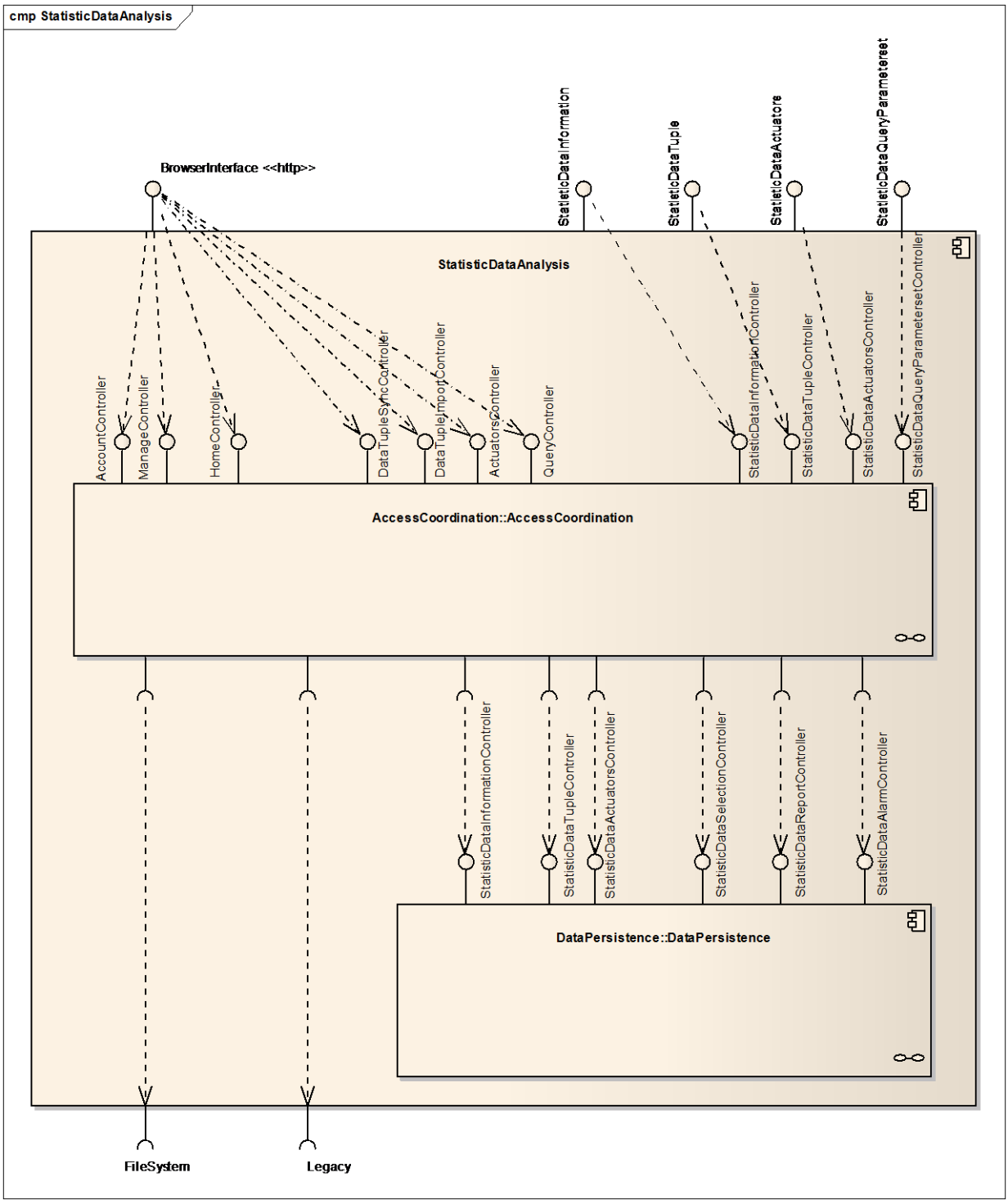


Abbildung 13: Komponenten-Diagramm des Gesamtsystems / Level 1

9.2 «AccessCoordination» (Level 2)

Folgendes Diagramm zeigt die Aufteilung des «AccessCoordination» Moduls in dessen Pakete und Schnittstellen. Die Darstellung soll einen Einblick geben, von welchen Paketen Anfragen bearbeitet werden, und wie die Abhängigkeiten zwischen den Paketen aufgebaut sind. Im Diagramm ist das Paket für die Implementierung der reinen Datenmodelle (vgl. Kapitel 7.2) bewusst nicht dargestellt. Datenmodelle besitzen hier keine Logik. Die «Business-Logik» gemäss «Model View Controller»-Muster (vgl. Kapitel 8.4.2) befindet sich in den Services. Die Implementierung der Datenmodelle ist in den Kapiteln 9.6 und 9.7 dargestellt.

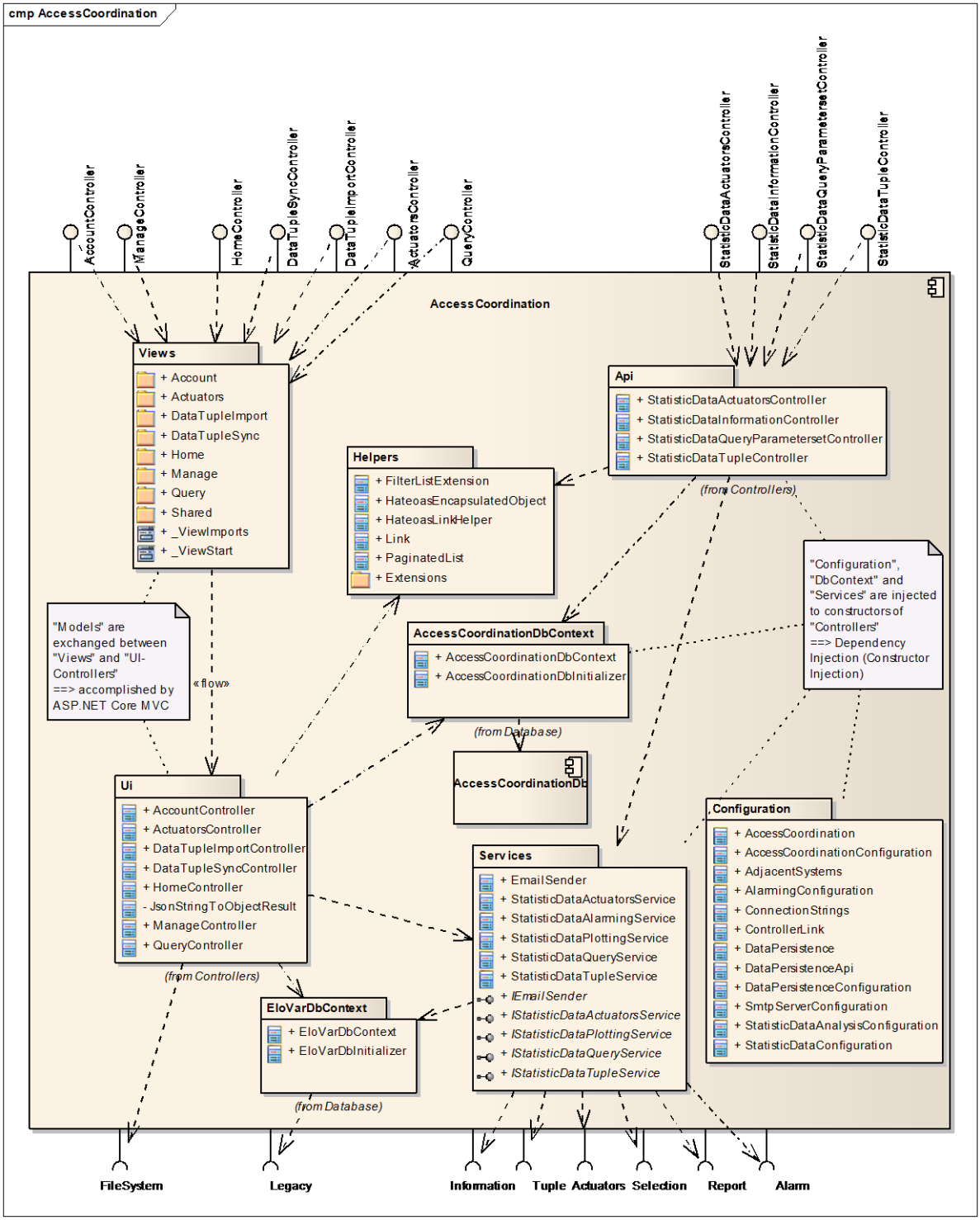


Abbildung 14: Paket-Diagramm «AccessCoordination»

Paket	Beschreibung
Configuration	Eine einfache Objekte Struktur, die es erlaubt, die Konfigurationsdatei «appsettings.json» in die Anwendung zu binden.
Controllers/Api	Diese Gruppe von Kontroller implementieren die REST-Ressourcen der Systemschnittstelle.
Controllers/Ui	Diese Gruppe von «MVC»-Kontrollern implementieren zusammen mit den «Views» das Webinterface.
Database/AccessCoordinationDbContext	Diese abgeleitete Klasse von «Microsoft.EntityFrameworkCore.DbContext» ist die Schnittstelle zum «Entity Framework Core»-OR-Mapper für die SQlite-Datenbank.
Database/EloVarDbContext	Diese abgeleitete Klasse von «Microsoft.EntityFrameworkCore.DbContext» ist die Schnittstelle zum «Entity Framework Core»-OR-Mapper für die MySQL-Datenbank.
Helpers	Helper-Klassen (z.B. Kapselung für HATEOAS-Links)
Services	Business-Logik des «AccessCoordination»-Moduls
Views	Die «Views» implementieren zusammen mit den Kontrollern das Webinterface.

9.2.1 Abhängigkeiten gegenüber dem ASP.NET Framework

«NuGet» ist ein Paket-Manager, der bei der Verwaltung von Bibliotheken hilft. Im nachfolgenden Diagramm sind die Abhängigkeiten zu «NuGet»-Paketen dargestellt.

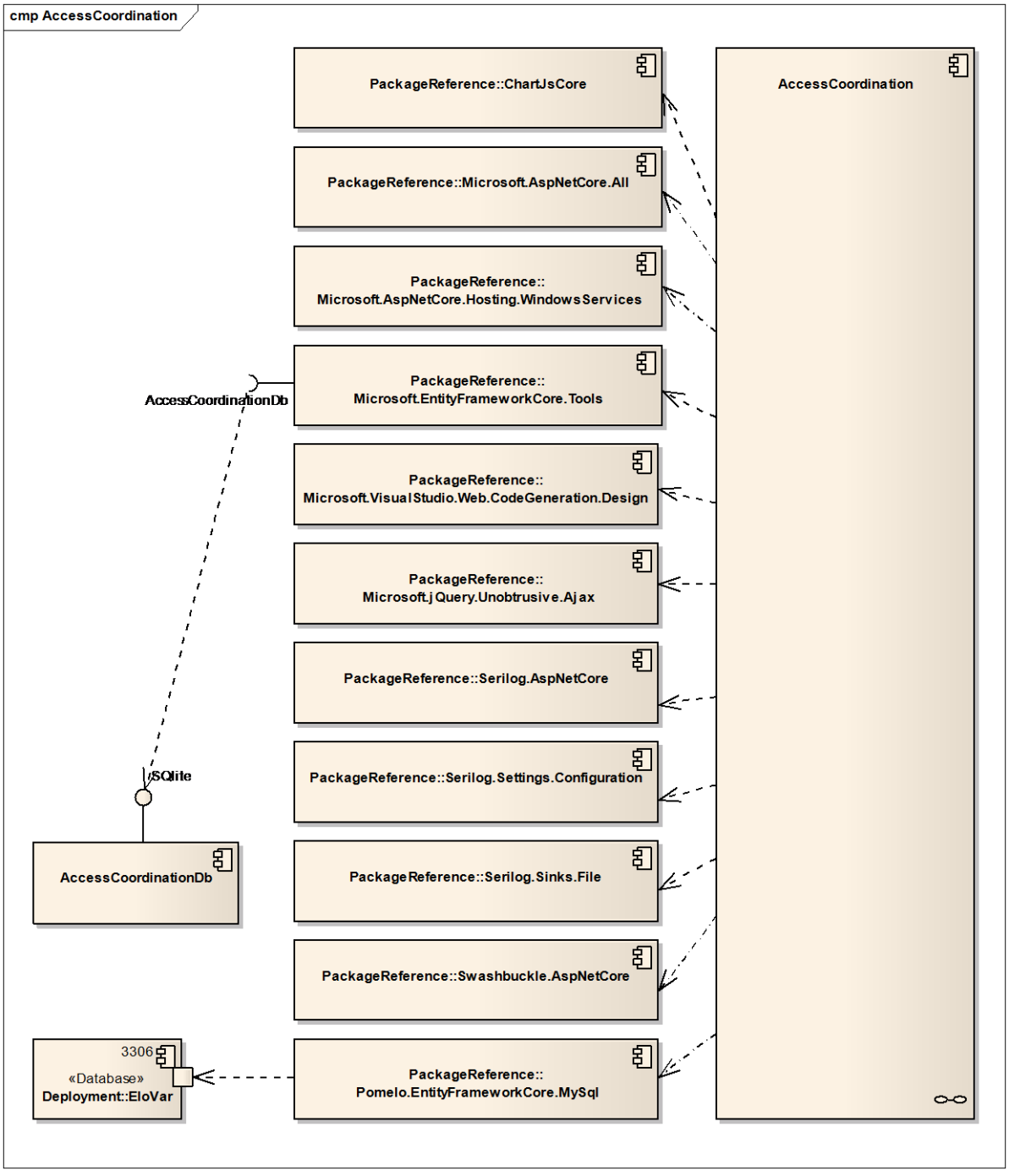


Abbildung 15: Abhängigkeiten gegenüber «NuGet-Packages» des ASP.NET Frameworks

NuGet-Package	Beschreibung
ChartJSCore Version="1.3.0"	Erleichtert die Darstellung von Diagrammen. Kapselt das «Chart.js»-Framework für Diagramme.
Microsoft.AspNetCore.All Version="2.0.9"	Bibliotheken des ASP-NET Frameworks (Grundfunktionen)
Microsoft.AspNetCore.Hosting .WindowsServices Version="2.1.1"	Ermöglicht, dass die Anwendung als «Windows Service» gestartet werden kann.
Microsoft.EntityFrameworkCore.Tools Version="2.1.1"	OR-Mapper (vgl. 8.4.5)
Microsoft.jQuery.Unobtrusive.Ajax Version="3.2.5"	Erleichtert die Verwendung von Ajax.
Microsoft.VisualStudio.Web .CodeGeneration.Design Version="2.0.4"	Bibliotheken des ASP-NET Frameworks (Grundfunktionen)
Pomelo.EntityFrameworkCore.MySql Version="2.1.1"	MySQL-Connector für Entity Framework Core
Serilog.AspNetCore Version="2.1.1"	Logging-Mechanismus
Serilog.Settings.Configuration Version="2.6.1"	Logging-Mechanismus
Serilog.Sinks.File Version="4.0.0"	Logging-Mechanismus-Erweiterung für Log-Dateien
Swashbuckle.AspNetCore Version="3.0.0"	Swagger-Dokumentation

9.3 DataPersistence (Level 2)

Folgendes Diagramm zeigt die Aufteilung des «DataPersistence»-Moduls in dessen Pakete und Schnittstellen. Die Darstellung soll einen Einblick geben, von welchen Paketen Anfragen bearbeitet werden, und wie die Abhängigkeiten zwischen den Paketen aufgebaut sind. Im Diagramm ist das Paket für die Implementierung der reinen Datenmodelle wiederum bewusst nicht dargestellt.

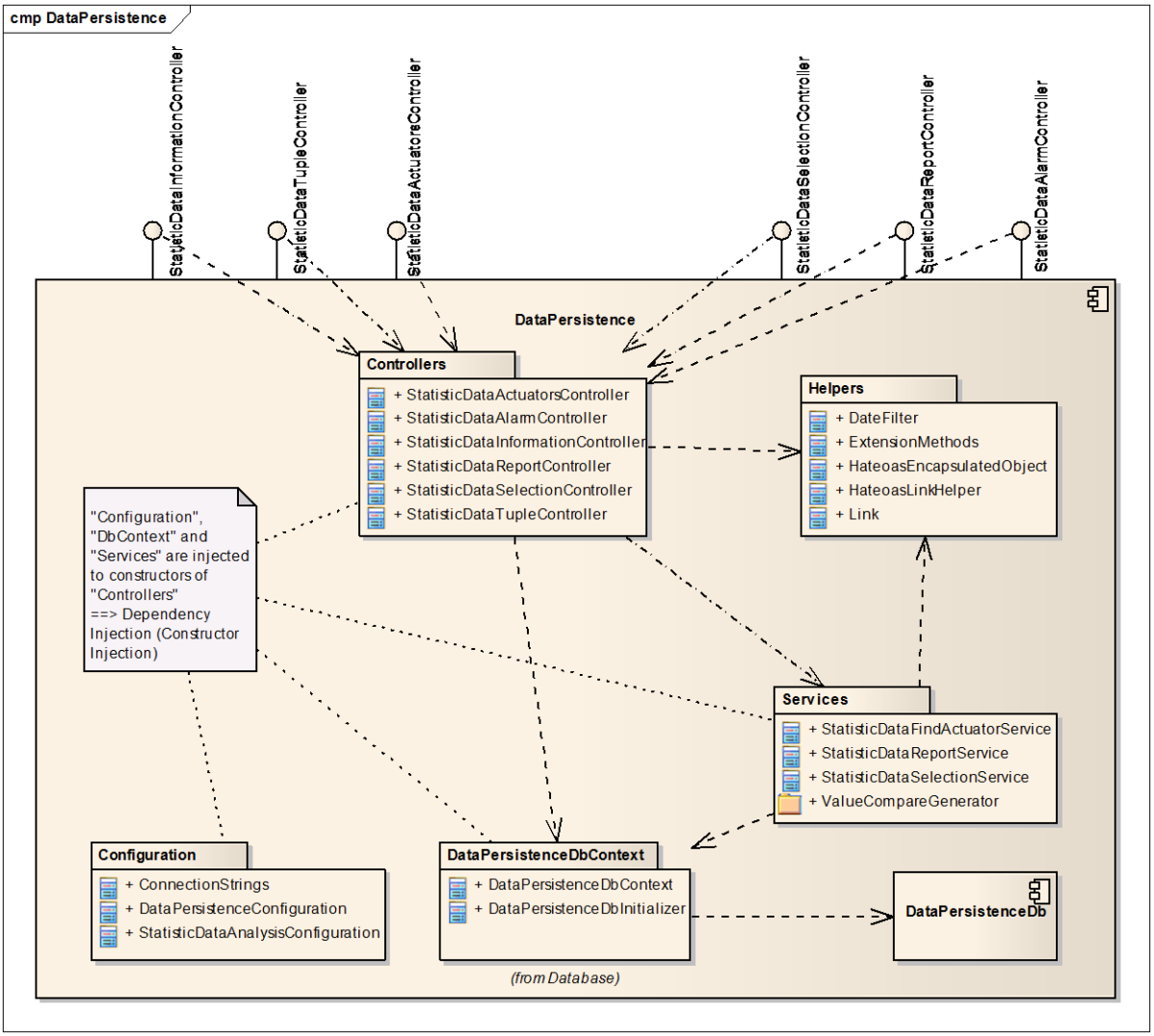


Abbildung 16: Paket-Diagramm “DataPersistence”

Paket	Beschreibung
Configuration	Eine einfache Objekte Struktur, die es erlaubt, die Konfigurationsdatei «appsettings.json» in die Anwendung zu binden.
Controllers	Diese Gruppe von Kontrollern implementieren die REST-Ressourcen der Systemschnittstelle.
DataPersistenceDbContext	Diese abgeleitete Klasse von «Microsoft.EntityFrameworkCore.DbContext» ist die Schnittstelle zum «Entity Framework Core»-OR-Mapper für die SQlite-Datenbank.
Helpers	Helper-Klassen (z.B. Kapselung für HATEOAS-Links)
Services	Business-Logik des «DataPersistence»-Moduls

9.3.1 Abhängigkeiten gegenüber dem ASP.NET Framework

«NuGet» ist ein Paket-Manager, der bei der Verwaltung von Bibliotheken hilft. Im nachfolgenden Diagramm sind die Abhängigkeiten zu «NuGet»-Paketen dargestellt.

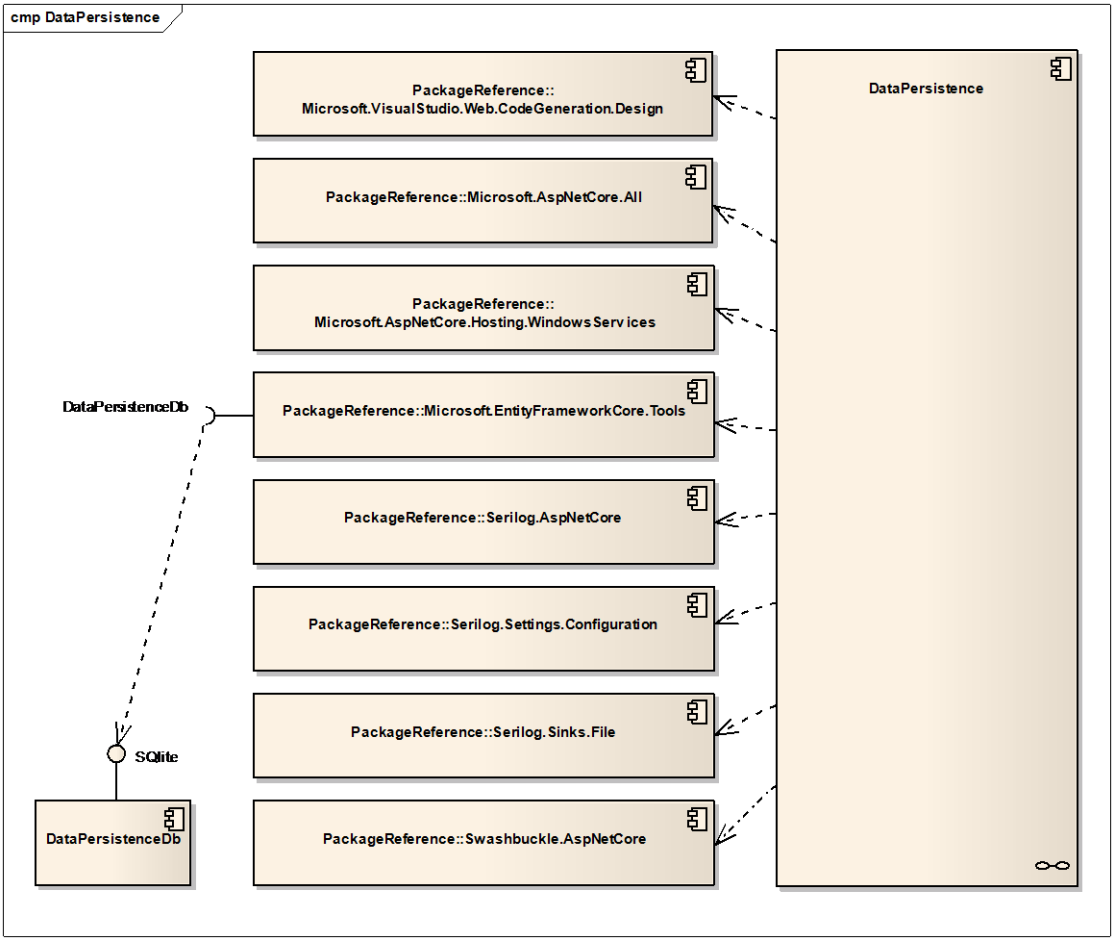


Abbildung 17: Abhängigkeit gegenüber Bibliotheken des ASP.NET Frameworks

NuGet-Package	Beschreibung
Microsoft.AspNetCore.All Version="2.0.9"	Bibliotheken des ASP-NET Frameworks (Grundfunktionen)
Microsoft.AspNetCore.Hosting .WindowsServices Version="2.1.1"	Ermöglicht, dass die Anwendung als «Windows Service» gestartet werden kann.
Microsoft.EntityFrameworkCore.Tools Version="2.1.1"	OR-Mapper (vgl. 8.4.5)
Microsoft.VisualStudio.Web .CodeGeneration.Design Version="2.0.4"	Bibliotheken des ASP-NET Frameworks (Grundfunktionen)
Serilog.AspNetCore Version="2.1.1"	Logging-Mechanismus
Serilog.Settings.Configuration Version="2.6.1"	Logging-Mechanismus
Serilog.Sinks.File Version="4.0.0"	Logging-Mechanismus-Erweiterung für Log-Dateien
Swashbuckle.AspNetCore Version="3.0.0"	Swagger-Dokumentation

9.4 AccessCoordination (Level 3)

Am Beispiel des «StatisticDataActuatorsController» soll nochmals gezeigt werden, wie mittels «Dependency Injection» (vgl. 8.4.4) die Abhängigkeiten zwischen einem Kontroller und der Business-Logik aufgebaut ist.

Der «StatisticDataActuatorsController» verwaltet die Ressource «/api/StatisticDataActuators». Diese Ressource hat folgende Beschreibung:

Methode	Ressource	Anwendungsfall
GET	/api/StatisticDataActuators	UC50: Statistik-Daten-Aktuator bekannt geben
GET	/api/StatisticDataActuators/{id}	UC50: Statistik-Daten-Aktuator bekannt geben

Um einen Statistik-Daten-Aktuator bekannt zu geben, muss er gefunden werden. Diese Aufgabe übernimmt der «StatisticDataActuatorsService», welcher das passende Interface realisiert. Auch der «StatisticDataActuatorsMockService» realisiert dieses Interface. Zu beiden Services kann also zur Laufzeit eine Abhängigkeit aufgebaut werden. Je nach Konfiguration ist das der Produktivbetrieb oder aber die Testumgebung. (vgl. 12). Das Interface «Ilogger» wird durch das Logging-Framework realisiert.

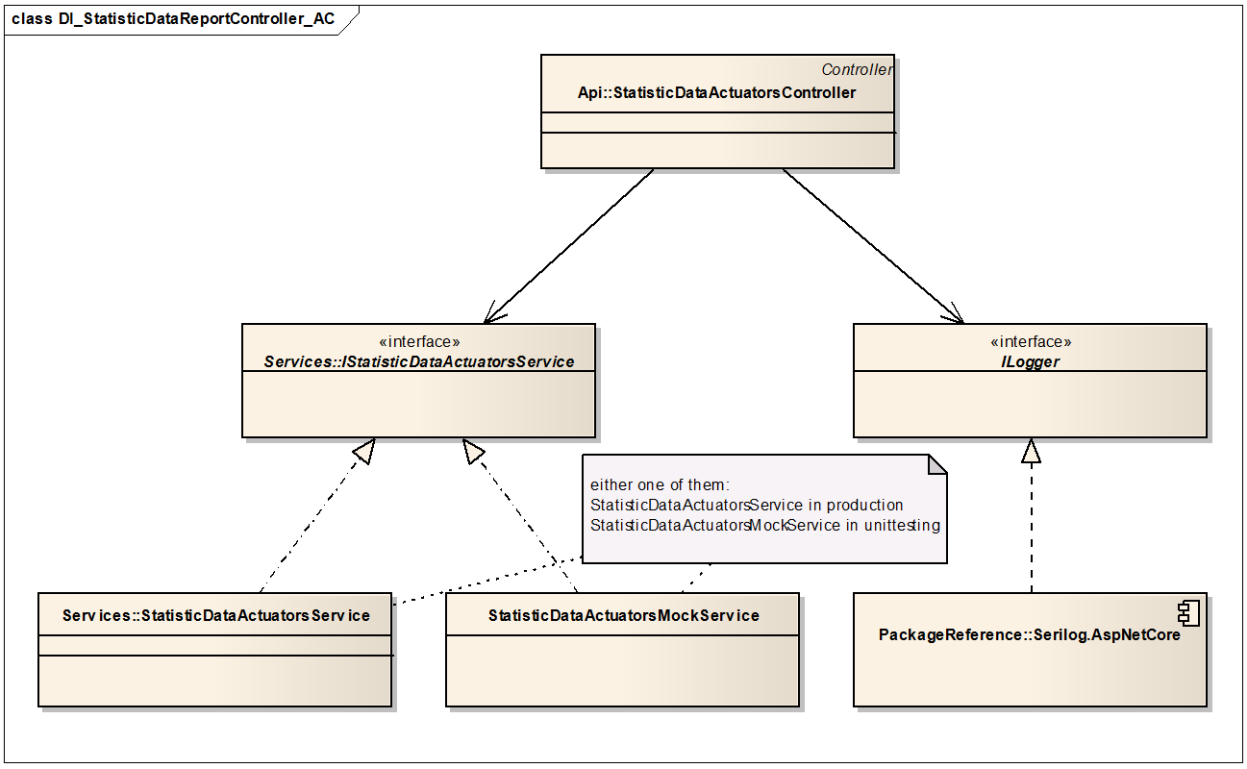


Abbildung 18: Zusammenhang zwischen «Controller» und «Service»

9.5 DataPersistence (Level 3)

Die zentrale Komponente des «DataPersistence» Moduls ist der «StatisticDataReportController». Er verwaltet die Ressource «/internalApi/StatisticDataReport». Diese Ressource hat folgende Beschreibung:

Methode	Ressource	Anwendungsfall
GET	/internalApi/StatisticDataReport	Gibt leeres Statistik-Daten-Report-Parameterset zurück.
POST	/internalApi/StatisticDataReport	UC20: Statistik-Daten-Report bekannt geben

Um einen Statistik-Daten-Report bekannt zu geben, müssen folgende Schritte durchgeführt werden:

1. Statistik-Daten-Report-Parameterset prüfen.
   → Diese Aufgabe übernimmt die Klasse «ReportParametersetHelpers».
   (Die Klasse existiert, die Funktionalität ist aber noch nicht implementiert.)
2. Sämtliche Statistik-Daten-Aktuatoren finden, die den Bedingungen entsprechen.
   → Diese Aufgabe übernimmt die Klasse «StatisticDataFindActuatorService».
3. Für jeden gefundenen Statistik-Daten-Aktuator muss die passende Statistik-Daten-Selektion erstellt werden.
   → Diese Aufgabe übernimmt die Klasse «StatisticDataSelectionService».
4. Sämtliche Statistik-Daten-Selektionen müssen zu Statistik-Daten-Reports erweitert werden.
   → Diese Aufgabe übernimmt die Klasse «StatisticDataReportService» mit Hilfe die Klasse «ValueCompareLinear», welcher das Interface «IValueCompare» realisiert.

Das Klassen-Modell zeigt, wie die verschiedenen Klassen miteinander verknüpft sind.

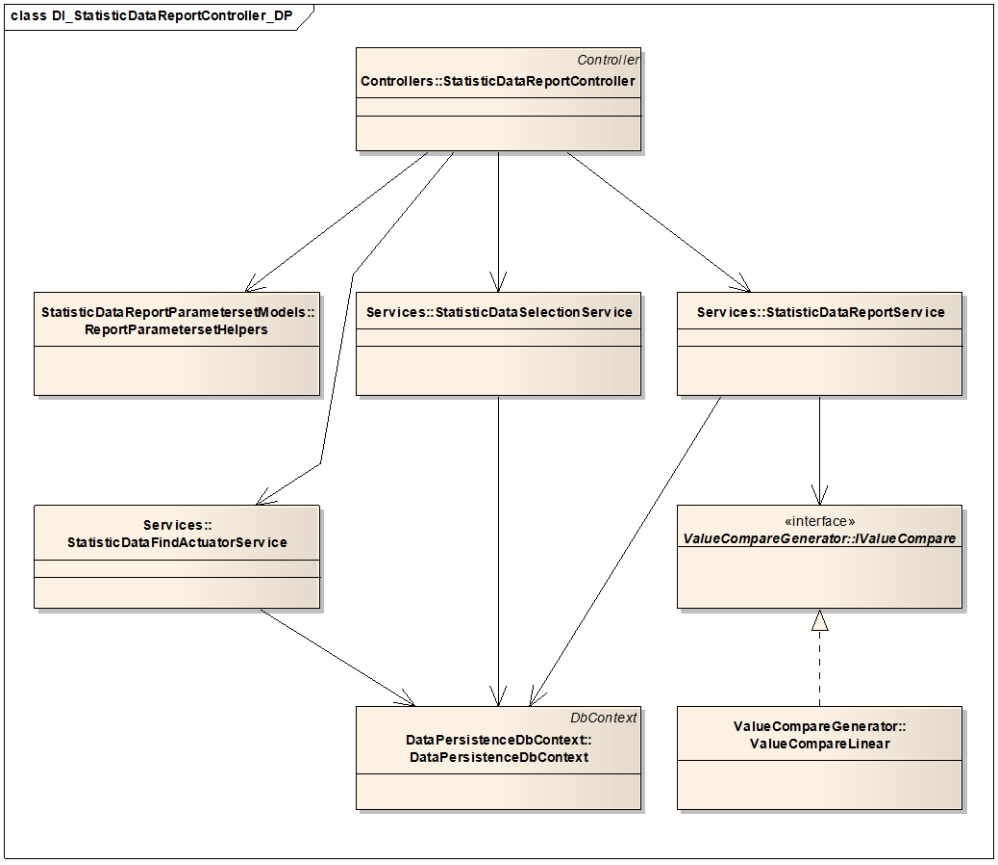


Abbildung 19: Zusammenhang zwischen «Controller» und «Service»

9.6 Datenmodelle «AccessCoordination»

9.6.1 «AccountViewModels», «ManageViewModels», «ApplicationUser», «ErrorViewModel»

Diese Modelle wurden durch das Visual Studio-Projekttemplate für Benutzerverwaltung mit Authentifizierung und Autorisierung eingeführt. Sie sind der Vollständigkeit halber aufgeführt.

9.6.2 «InformationApiModel»

Ein einfaches Objekt mit Schnittstellen-Informationen, dass bei einer Anfrage auf die Basis-Ressource «/api» zurückgegeben wird. (vgl. 3.4.3 «UC60: API-Information bekannt geben»)

9.6.3 Statistik-Daten-Aktuator / «StatisticDataActuatorApiModel»

Dieses Modell entsprich der Schnittstellenbeschreibung (vgl. 7.2.5).

9.6.4 Statistik-Daten-Abfrage-Parameterset / «StatisticDataQueryParameterset»

Die zentrale Datenstruktur des «AccessCoordination»-Moduls ist das «StatisticDataQueryParameterset». Es wird daher in verschiedenen Versionen benötigt.

StatisticDataQueryParameterset	Beschreibung
*ApiModel	Basis-Version
*DbModel	Für die Speicherung in einer Datenbank muss die Struktur auf Tabellen verteilt werden. (vgl. 11.4.1) Aus diesem Grund wurden «StatisticDataQueryParameterset», «Condition» und «SelectedStatisticDataQueryTupleNumericAttribute» um Referenzen erweitert.
*ViewModel	Bei der Implementierung gab es Probleme mit «nullable Types». Eine Lösung war das «SelectedStatisticDataQueryTupleNumericAttributeViewModel», welches ohne «nullable Type» auskommt.

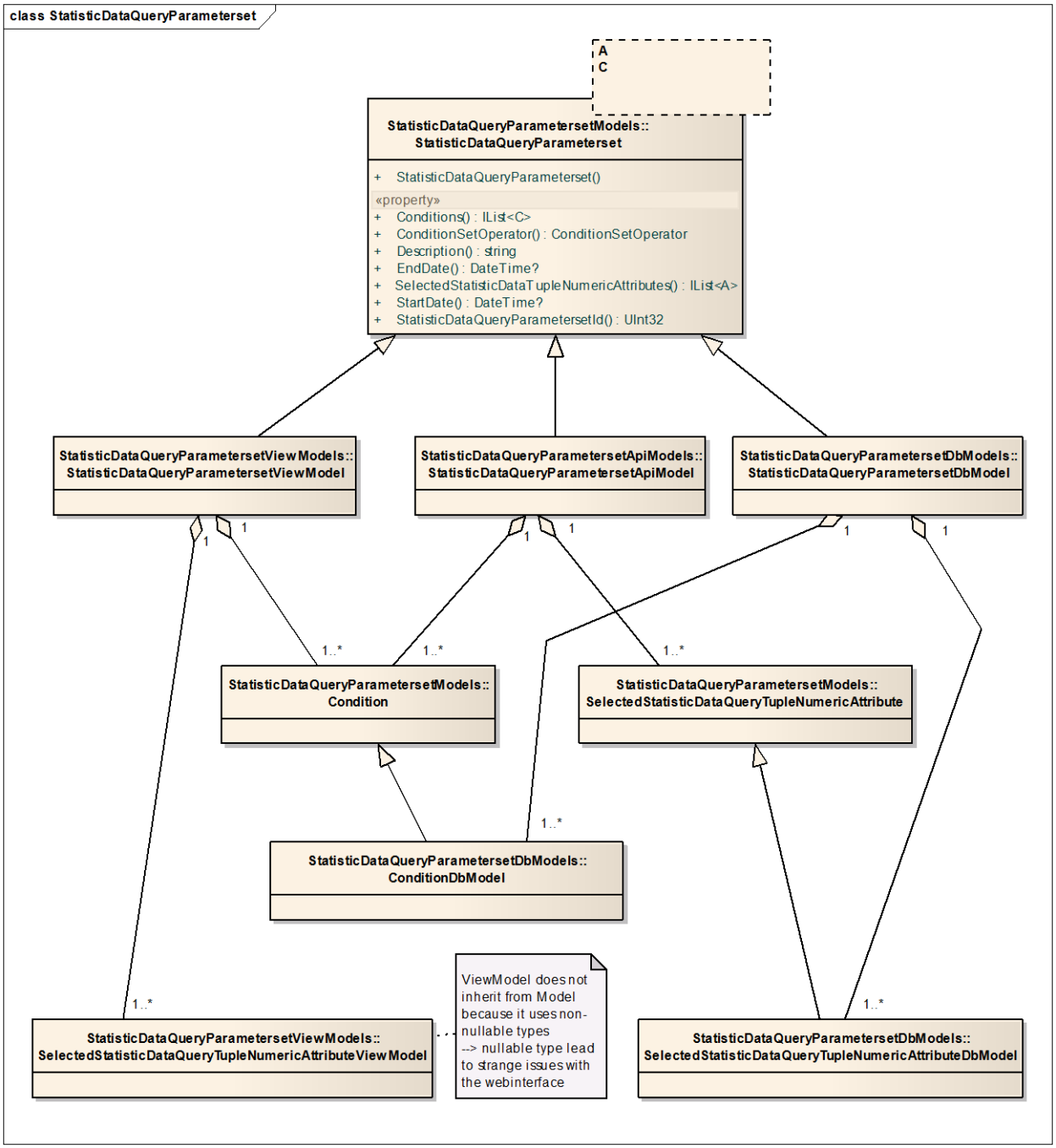


Abbildung 20: Klassendiagramm «StatisticDataQueryParameterset»

9.6.5 Statistik-Daten-Selektion / -Report / -Alarmierung / «StatisticDataQuery»

Ein weiteres zentrales Element ist die Gruppe Statistik-Daten-Selektion / -Report und -Alarmierung, welche zum «StatisticDataQuery» zusammengefasst wurden. Das «StatisticDataReportViewModel» ist etwas spezieller, da es das Modell mittels einem «Chart»-Objekt der «ChartJSCore»-Bibliothek um die graphische Darstellung der Daten erweitert.

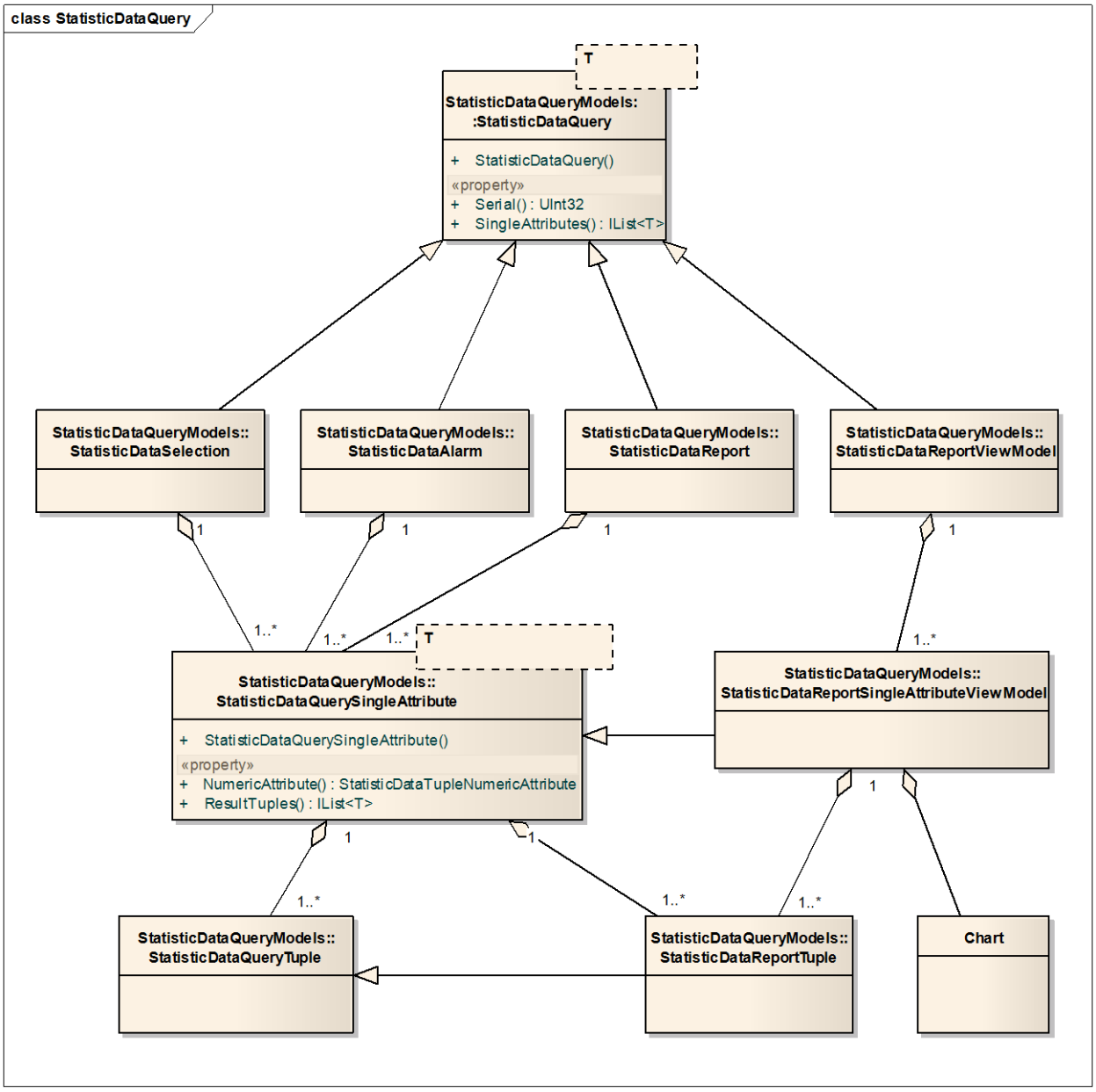


Abbildung 21: Klassendiagramm «StatisticDataQuery»

9.6.6 Statistik-Daten-Tupel / «StatisticDataTuple»

Das Statistik-Daten-Tupel ist ein einfaches Objekt, welches sämtliche primitiven Werte eines Datensatzes enthält. Für die temporäre Speicherung in der Datenbank, gibt es auch von diesem Modell ein Pedant, dass um einen Primärschlüssel erweitert wurde. Auf die Darstellung wird aus Platzgründen verzichtet. (vgl. 7.2.3 & 16.1)

9.7 Datenmodelle «DataPersistence»

9.7.1 «InformationApiModel»

Einfaches Objekt mit Schnittstellen-Informationen, dass bei einer Anfrage auf die Basis-Ressource «/internalApi» zurückgegeben wird. (vgl. 9.6.2)

9.7.2 Statistik-Daten-Aktuator / «StatisticDataActuator»

Die zentrale Datenstruktur des «DatenPersistence»-Moduls ist der «StatisticDataActuator». Die Grundlage für das Modell ist in der Schnittstellenbeschreibung (vgl. 7.2.5) dokumentiert.

Da das Modell in der Datenbank gespeichert werden muss. Ist es auf verschiedene Tabellen verteilt. Sämtliche Objekte sind darum um Referenzen erweitert worden. Als weitere Erweiterung wurden dem Modell noch ein Mechanismus für Log-Einträge hinzugefügt. Dieser Mechanismus erlaubt es, in den Log-Einträgen nachzuverfolgen, mit welchen Statistik-Daten-Tupel der Aktuator aufdatiert wurde.

Das folgende Klassen-Diagramm zeigt den Aktuator, wobei aufgrund der Übersicht die «Aggregation» zwischen den Klassen für die rechte Seite nur exemplarisch gezeichnet wurde. Die Erweiterung um Referenzen (mittels Vererbung) wurde ganz weggelassen.

(nicht öffentlich)

Abbildung 22: Klassendiagramm «StatisticDataActuator»

9.7.3 Statistik-Daten-Abfrage-Parameterset / «StatisticDataQueryParameterset»

Im Gegensatz zum Statistik-Daten-Abfrage-Parameterset des «AccessCoordination»-Moduls wird bei diesem Modell zwischen den verschiedenen Typen unterschieden. Der Grund ist, dass dieses Modell zu einem anderen Zeitpunkt entwickelt wurde. Ziel ist es, die Modell zu vereinheitlichen.

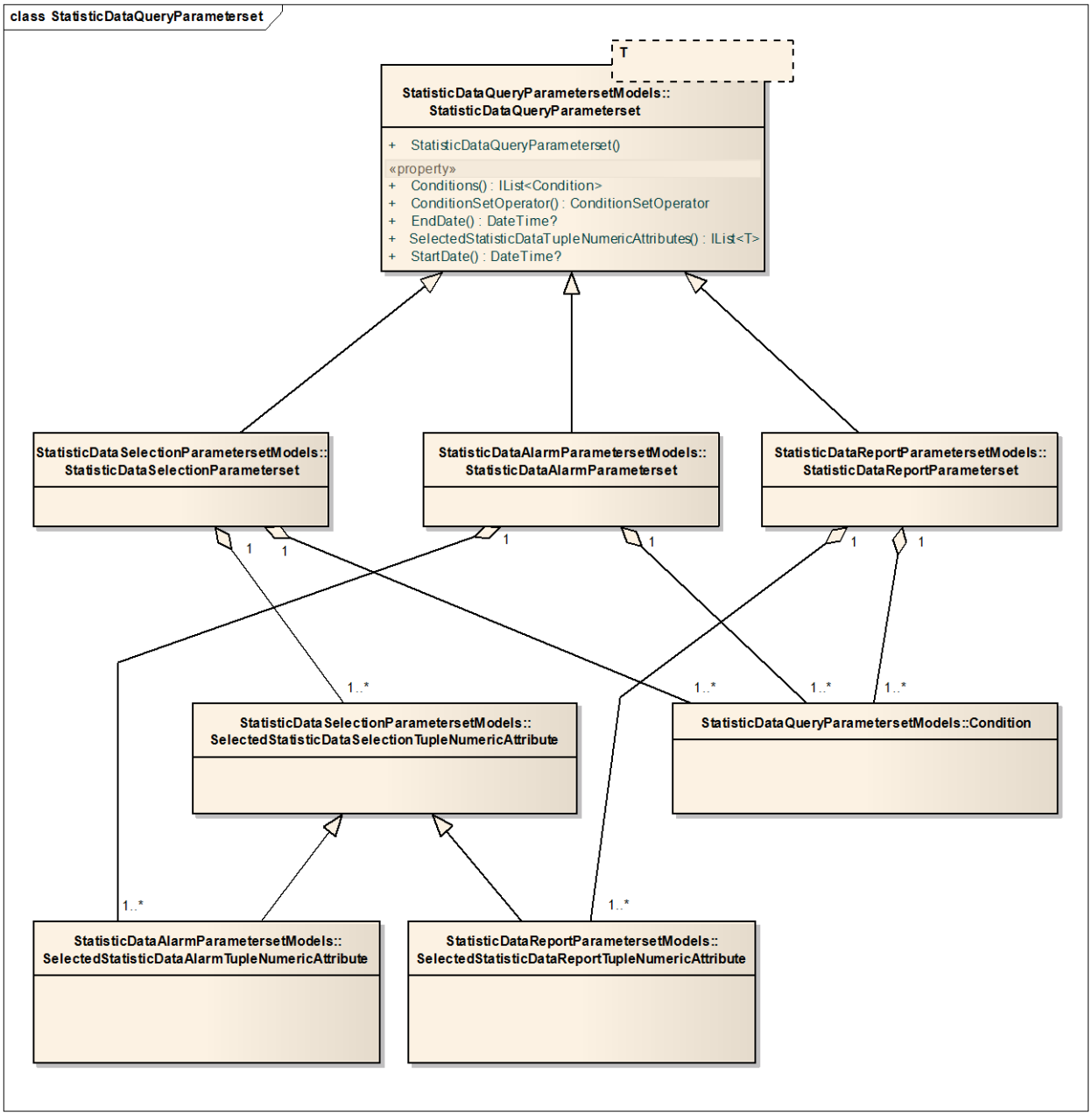


Abbildung 23: Klassendiagramm «StatisticDataQueryParameterset»

9.7.4 Statistik-Daten-Selection / -Report / -Alarmierung / «StatisticDataQuery»

Das Klassen-Diagramm für die Statistik-Daten-Selection / -Report und -Alarmierung entspricht weitgehend dem Modell des «AccessCoordination»-Moduls (vgl. 0), wobei auf das «ViewModel» verzichtet werden kann.

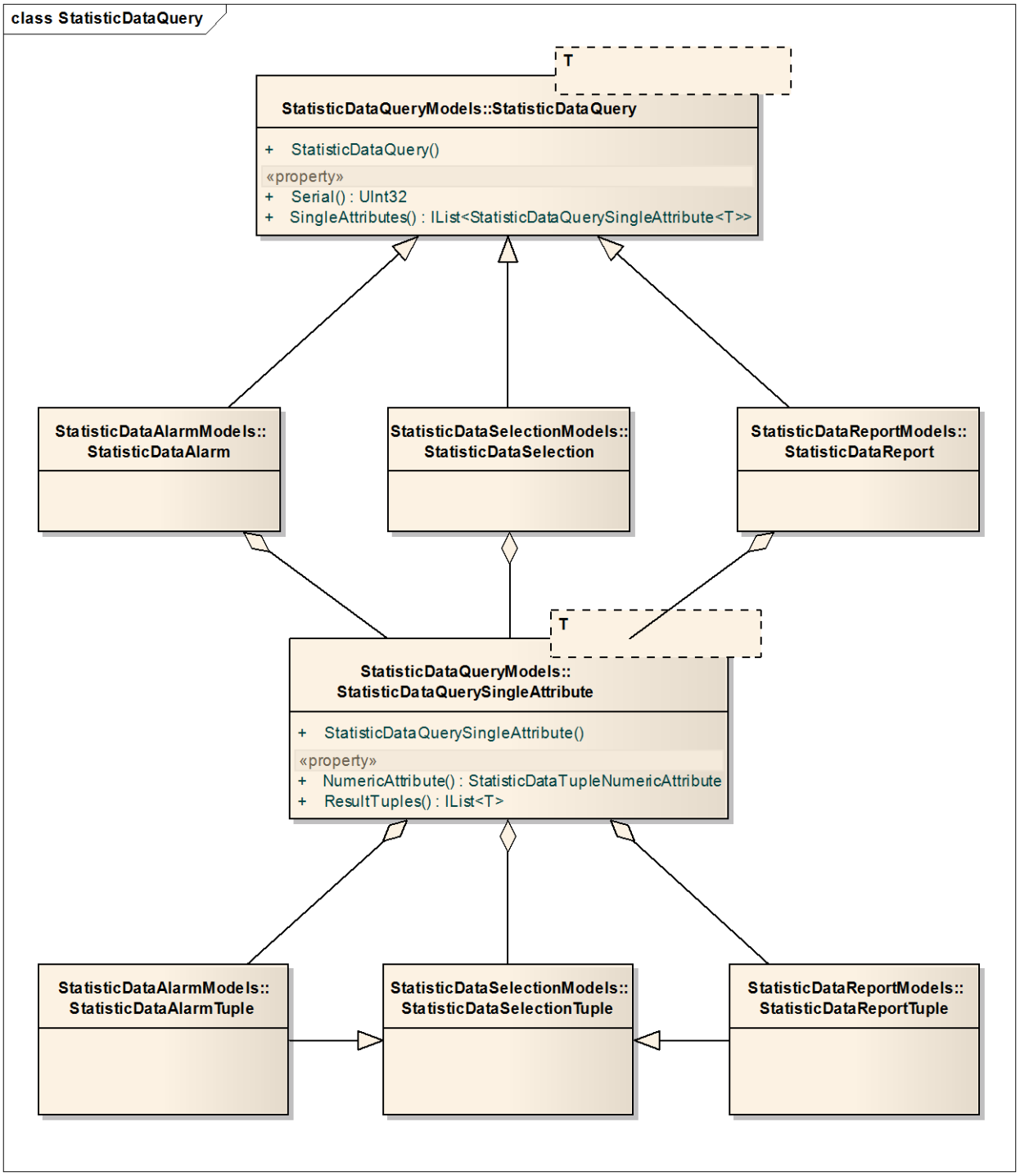


Abbildung 24: Klassendiagramm «StatisticDataQuery»

9.7.5 Statistik-Daten-Tuple / «StatisticDataTuple»

Das Statistik-Daten-Tuple entspricht weitgehend dem Modell des «AccessCoordination»-Moduls, welches auch in der Schnittstellenbeschreibung und im Anhang erläutert ist (vgl. 7.2.3 und 16.1).

Zudem gibt es noch ein reduziertes Modell «StatisticDataTupleLogDbModel», welches nur aus dem «Primärschlüssel» und Referenzen für die Datenbank besteht. Objekte von dieser Klasse werden für Logging-Zwecke permanent in der Datenbank gespeichert. Diese Log-Einträge werden für die Synchronisation mit dem «EloVarDb»-Datenbank (Legacy-System) und der Anwendung «Statistik-Daten Auswertung» benötigt.

10 Laufzeitsicht

10.1 Services

Im Kapitel 8.4.2 und 8.4.3 wurde das «Model View Controller» Muster eingeführt. Die beiden Kapitel beschreiben, wie das Muster in dieser Anwendung umgesetzt wurde. Es wurde aufgezeigt, dass die Business-Logik in den Services implementiert ist. Kapitel 8.4.4 erklärt, wie das Muster der «Dependency Injection» für das Bereitstellen der Services genutzt wird, und wie die Services während der Initialisierung beim «Dependency Injection»-Framework registriert werden. Services werden in den beiden Modulen jeweils im Paket «Services» zusammengefasst. Nachfolgend werden die Services der beiden Module kurz erläutert.

10.1.1 Services «AccessCoordination»

Hauptaufgabe der Services des «AccessCoordination»-Moduls ist das Verwalten von Statistik-Daten-Query-Parametersets und die Kommunikation mit dem «DataPersistence»-Modul.

Name	Beschreibung
Services.EmailSender (transient)	Dieser Service ist Teil des Quellcode-Gerüsts, dass bei der Initialisierung für ein Projekt mit Unterstützung von Autorisierung und Authentifizierung erzeugt wurde.
Services.StatisticDataActuatorsService (transient)	Der «ActuatorsService» beinhaltet keine eigentliche Business-Logik. Er verwaltet die Kommunikationsverbindung gegenüber der «/api/StatisticDataActuators»-Ressource des «DataPersistence»-Moduls.
Services.StatisticDataAlarmingService (Singleton)	Der «AlarmingService» implementiert das Interface «IHostedService» des Frameworks. Diese Funktionalität erlaubt es, dass Services periodische Tasks ausführen können, indem sie von einem Timer getriggert werden. EA03 Alarmierungsfunktion mittels E-Mail 1. Der «AlarmingService» prüft periodisch alle Statistik-Daten-Query-Parametersets, ob bei diesen die Alarmierung aktiviert ist. 2. Falls dies zutrifft, startet der Service eine Anfrage beim «DataPersistence» Modul. 3. Ist die Antwort positiv, wird eine E-Mail an die Adresse, die im Parameterset hinterlegt ist, versendet.
Services.StatisticDataPlottingService (transient)	Der «PlottingService» hat eine einfache Aufgabe, er befüllt ein Objekt des Typs «Chart». Dieses Objekt wird von der externen Bibliothek «ChartJSCore» für die visuelle Darstellung der Daten verwendet.
Services.StatisticDataQueryService (transient)	Der «QueryService» beinhaltet folgende Funktionalitäten: - Parametersets aus der «AccessCoordinationDb» lesen - Parametersets aus der «AccessCoordinationDb» hinzufügen - Parametersets aus der «AccessCoordinationDb» editieren - Parametersets aus der «AccessCoordinationDb» löschen Zudem verwaltet er die Kommunikationsverbindung gegenüber folgenden Ressourcen des «DataPersistence» Moduls:

	• /api/StatisticDataAlarm • /api/StatisticDataReport • /api/StatisticDataSelection
Services.StatisticDataTupleService (transient)	Der «TupleService» beinhaltet folgende Funktionalitäten: - Statistik-Daten-Tupel aus der «AccessCoordinationDb» lesen - Statistik-Daten-Tupel aus der «AccessCoordinationDb» hinzufügen - Statistik-Daten-Tupel aus der «AccessCoordinationDb» editieren - Statistik-Daten-Tupel aus der «AccessCoordinationDb» löschen Zudem verwaltet er die Kommunikationsverbindung gegenüber der «/api/StatisticDataTuple»-Ressource des «DataPersistence» Moduls.
Models.*ModelService (transient)	Die Gruppe der «*ModelService» bildet eine Ausnahme. Diese implementieren keine Logik, sondern transformieren Modelle und sind daher auch im Paket «Models» eingeordnet.

10.1.2 Services «DataPersistence»

Hauptaufgabe der Services des «DataPersistence» Moduls ist das Verwalten von Statistik-Daten-Aktuatoren, und das Erzeugen von Statistik-Daten-Selektionen / -Reports und -Alarmierungen.

Name	**Beschreibung**
Services.StatisticDataFindActuatorService	Dieser Service ermöglicht es, dass Statistik-Daten-Aktuatoren aufgrund eines Statistik-Daten-Query-Parametersets in der Datenbank gefunden werden. 1. Die Datenbank wird dabei aufgrund einer Query-Bedingung nach passenden Aktuatoren durchsucht. 2. Sämtliche gefundenen Aktuatoren werden in einer temporären Liste vermerkt. (select) 3. Besteht das Parameterset aus mehreren Bedingungen, wird dieser Vorgang wiederholt. 4. Wird die Schnittmenge gesucht, wird die temporäre Liste jeweils mit den gefundenen Aktuatoren vergangener Suchvorgänge verglichen und geschnitten. (intersect) 5. Wird die Vereinigung gesucht, werden sämtliche temporären Listen zusammengehängt. (addrange) 6. Für die Bestimmung der endgültigen Menge werden mehrfach gefundene Aktuatoren zusammengefasst. (distinct)
Services.StatisticDataSelectionService	Der «SelectionService» erzeugt die gewünschte Statistik-Daten-Selektion für die gefundenen Aktuatoren.
Services.StatisticDataReportService	Wird ein Statistik-Daten-Report erzeugt. Erweitert der «ReportService» die Statistik-Daten-Selektion zu einem Statistik-Daten-Report. Für die Berechnung des Vergleichswerts wird ein Service des «ValueCompareGenerator»-Pakets verwendet. (Im Moment gibt es nur den «ValueCompareLinear».)

Services.ValueCompareGenerator .ValueCompareLinear	Ein einfacher Service, der den Vergleichswert berechnet. Durch die Implementierung als Service kann die Berechnung gut isoliert und ausgetauscht werden. Die «Dependency Injection» würde sogar einen Austausch zur Laufzeit ermöglichen. Zudem können dadurch einfacher Unittests geschrieben werden.
Models.*Helpers (transient)	Die Gruppe der «*Helpers» bildet eine Ausnahme. Diese implementieren keine Logik, sondern transformieren Modelle und sind daher auch im Paket «Models» eingeordnet.

10.2 Hypermedia As The Engine Of Application State (HATEOAS)

Die folgenden Darstellungen ergänzen das Kapitel 8.2 und zeigen die vollständige Zustandsmaschine der REST-Programmierschnittstelle und deren Ressourcen.

10.2.1 HATEOAS «AccessCoordination»

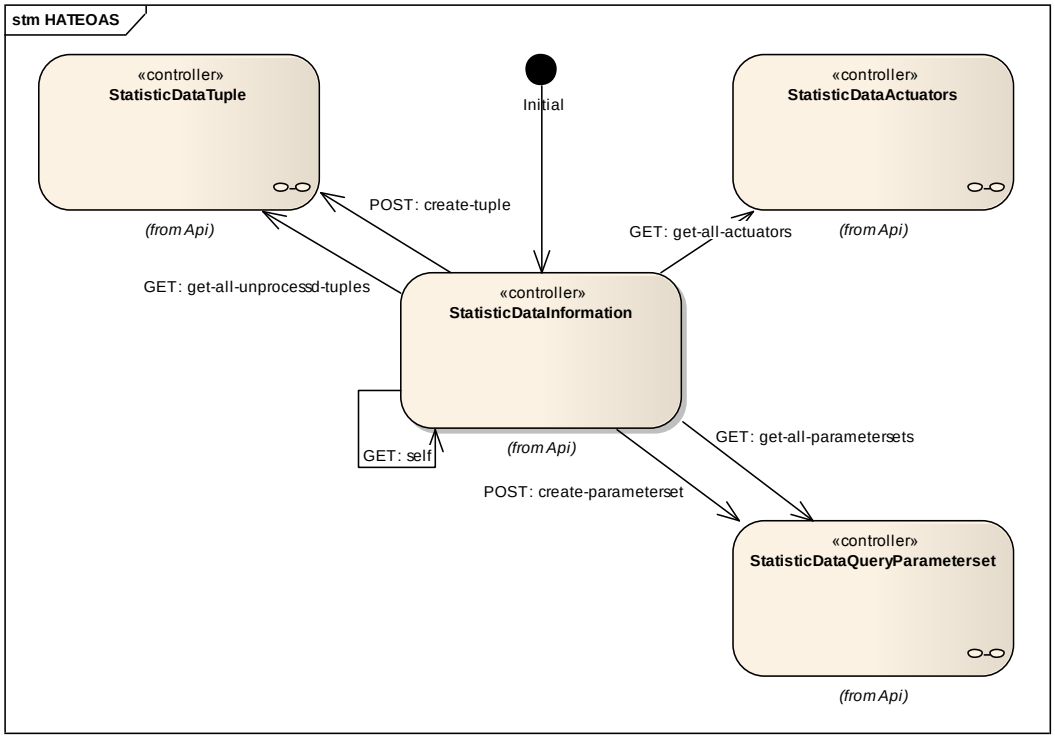


Abbildung 25: «/api»

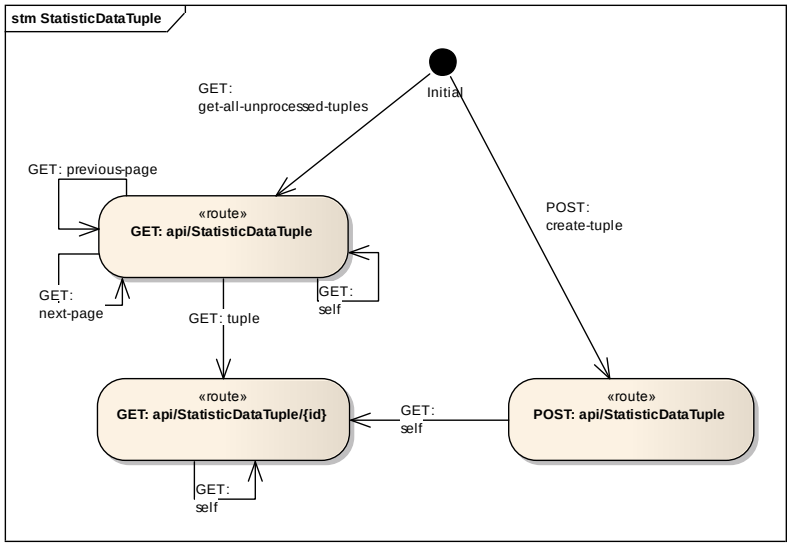


Abbildung 26: «/api/StatisticDataTuple»

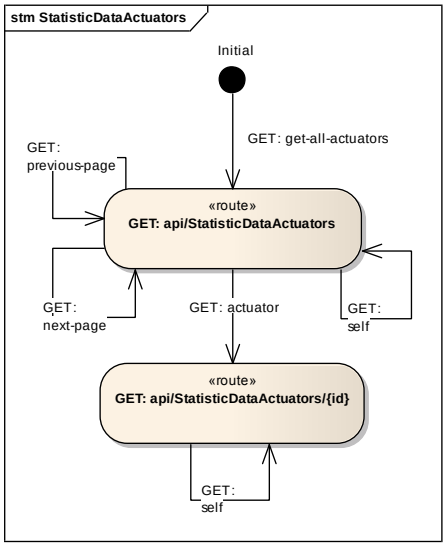


Abbildung 27: «/api/StatisticDataActuator»

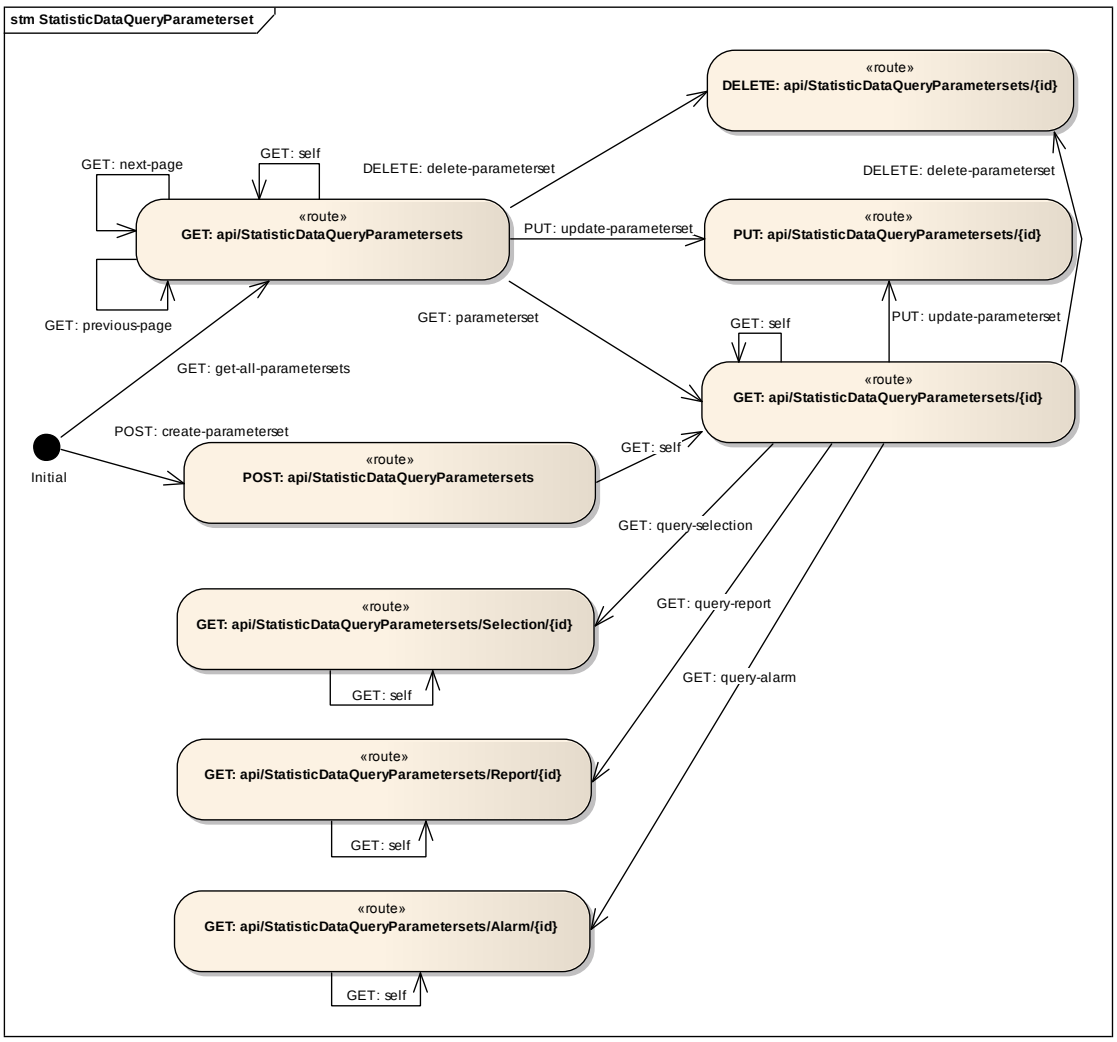


Abbildung 28: «/api/StatisticDataQueryParametersets»

10.2.2HATEOAS «DataPersistence»

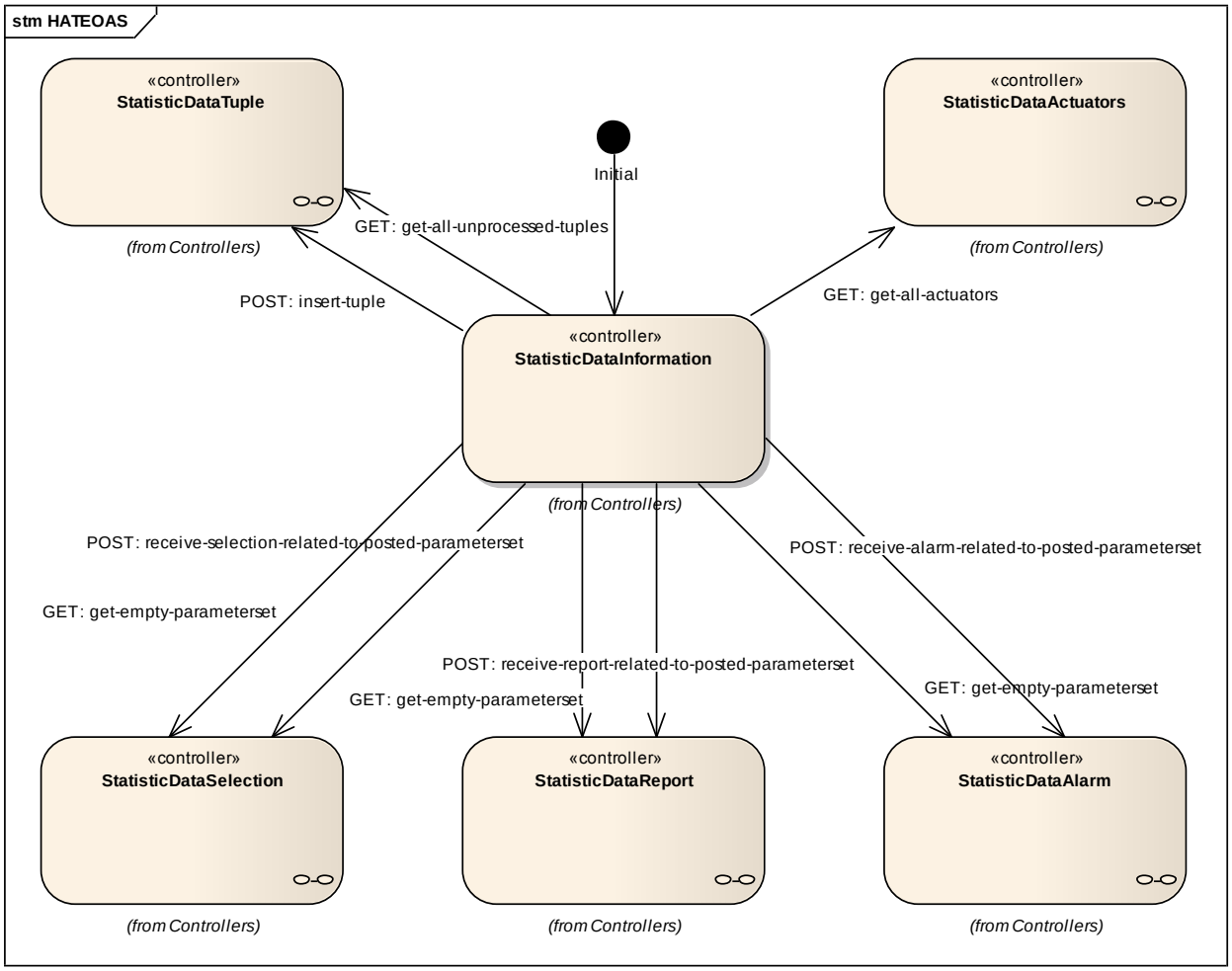


Abbildung 29: «/internalApi»

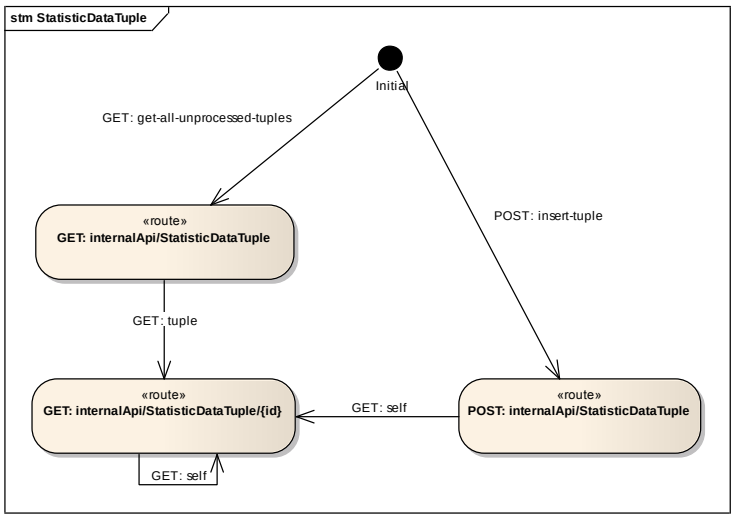


Abbildung 30: «/internalApi/StatisticDataTuple»

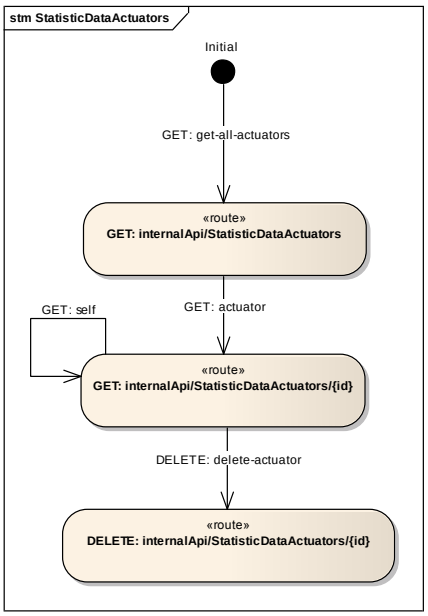


Abbildung 31: «/internalApi/StatisticDataActuator»

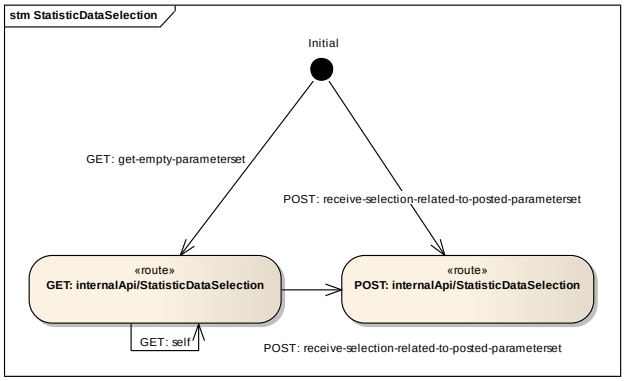


Abbildung 32: «/internalApi/StatisticDataSelection»

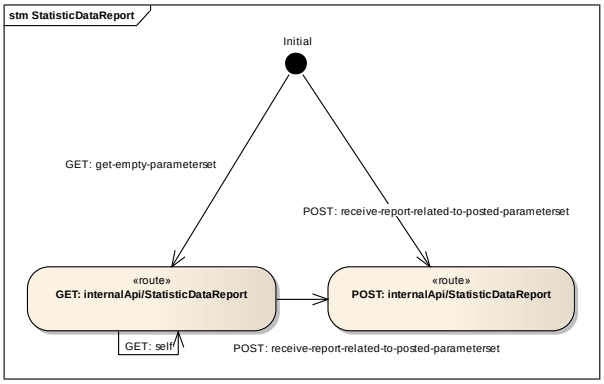


Abbildung 33: «/internalApi/StatisticDataReport»

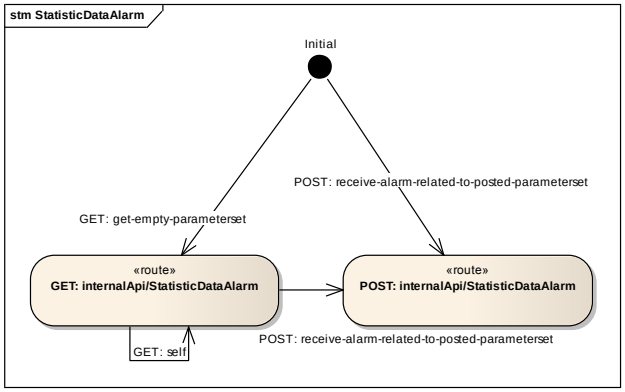


Abbildung 34: «/internalApi/StatisticDataSelection»

11 Verteilungssicht

11.1 Überblick

Die wesentlichen drei Bestandteile des Systems «Statistik-Daten Auswertung» sind «AccessCoordination», «DataPersistence» und «EloVar Datenbank». Diese drei Bestandteile können unabhängig voneinander verteilt werden, solange die Kommunikationsverbindungen zwischen den Systemen gewährleistet ist.

Es ist vorgesehen, die Module «AccessCoordination» und «DataPersistence» auf dem gleichen Host zu installieren (vgl. Abbildung 35). Die Module sind aber lose gekoppelt, und eine Verteilung auf zwei Systeme würde auch funktionieren. Das Altsystem «EloVar Database» ist gegeben und kann nicht verschoben werden.

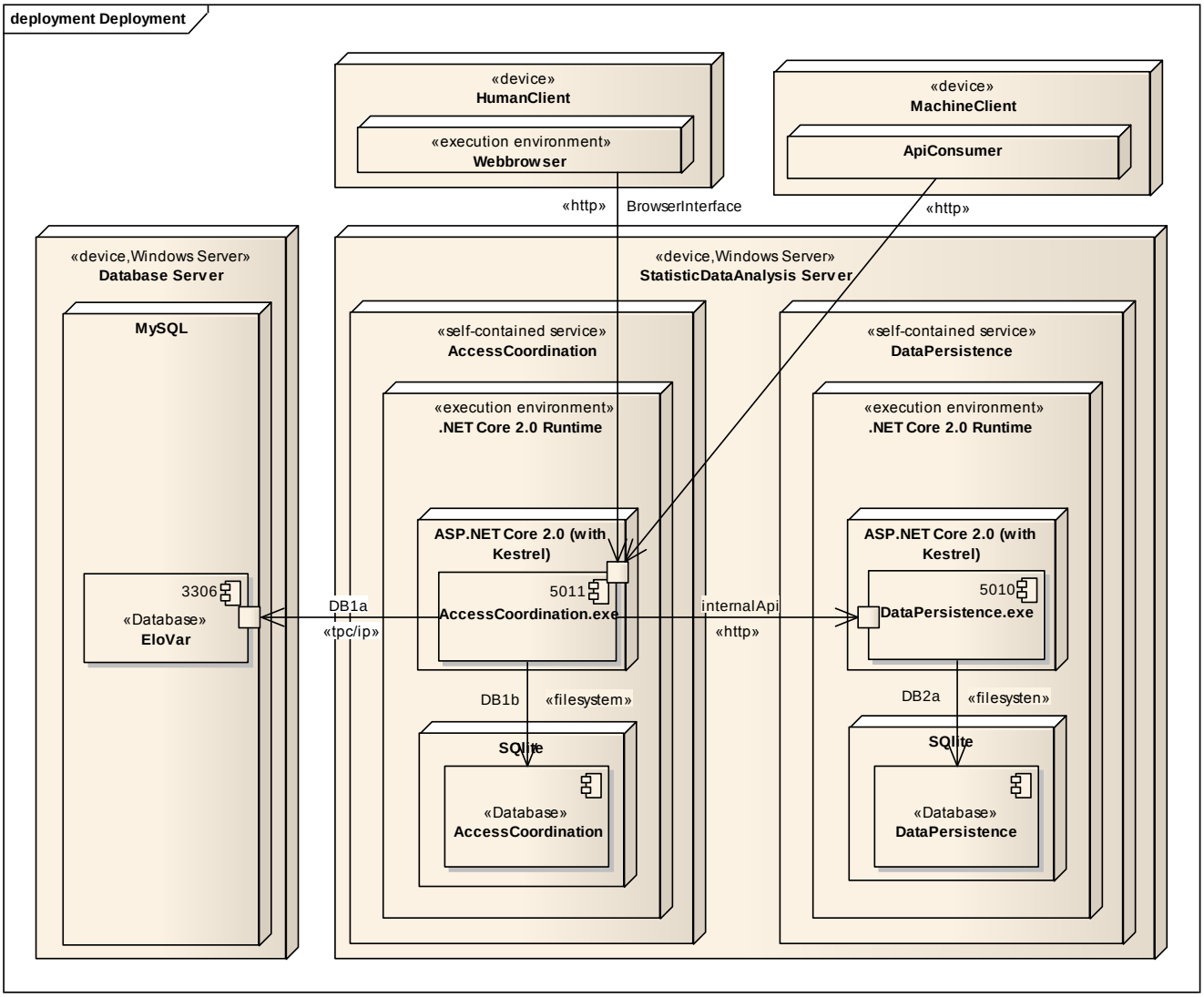


Abbildung 35: Gesamtüberblick über die Verteilung des Systems

Name	Beschreibung
AccessCoordination	Dieses Modul ist ein «self-contained» .NET Core 2.0 Service. Das heisst, sämtliche Anwendungsmodule, .NET Komponenten und Bibliotheken sind zusammen mit den Konfigurationsdaten in einem Dateiordner gebunden und können so auf eine Windows Maschine verteilt werden. Windows Versionen ab Windows 7 genügen für eine Ausführung. Es besteht die Möglichkeit die Anwendung als Service zu registrieren, wenn man sie mittels SC-Tool (sc.exe) bei Windows registriert. Für temporäre Installationen kann die Registrierung auch umgangen werden, und die Anwendung muss dann von der Konsole aus gestartet werden. Theoretisch wäre das System auch unter Linux lauffähig, dazu müsste es aber neu zusammengebaut werden. Dies wurde bis jetzt noch nie untersucht.
DataPersistence	Das «DataPersistence»-Modul ist nach demselben Prinzip wie das «AccessCoordination»-Modul erstellt worden.
EloVar Datenbank	Die EloVar Datenbank ist Bestandteil eines bestehenden MySQL-Datenbank-Servers, welcher auf Windows 2008 läuft.

11.2 Abhängigkeiten und Kommunikation

Abhängigkeiten und Kommunikation innerhalb eines «self-contained» .NET Core 2.0 Service sollen anhand des «DataPersistence»-Modul erläutert werden.

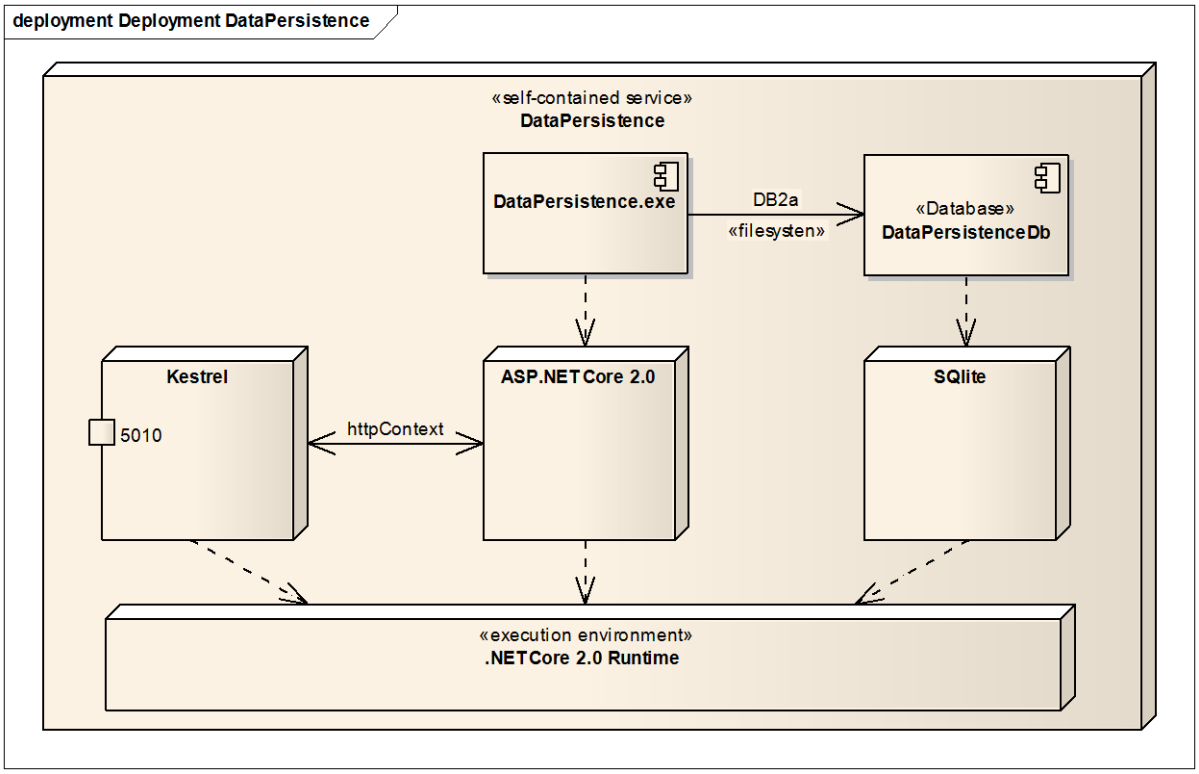


Abbildung 36: «self-contained service»

Komponente	Beschreibung
DataPersistence.exe	Die Implementation des «DataPersistence»-Modul inklusive eingebundener Bibliotheken.
DataPersistenceDb	Die SQlite-Datenbank-Datei im Dateisystem des Windows Server.
ASP.NET Core 2.0	Bibliotheken, die es erlauben eine Webanwendung auf Basis von .NET Core 2.0 zu entwickeln. z.B. Middleware, Dependency Injection, u.s.w.
SQlite	Bibliothek, die die SQlite-Datenbank-Datei verwaltet.
Kestrel	Ein leichtgewichtiger Web Server, der auf .NET Core 2.0 läuft. https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/
.NET Core 2.0 Runtime	Laufzeitumgebung, die den .NET Common Intermediate Language Code ausführt.

11.3 Konfiguration verteilter Anwendungen

Im Verzeichnis wo sich die ausführbare Datei (DataPersistence.exe oder AccessCoordination.exe) befindet, ist jeweils auch die Datei «appsettings.json» abgelegt. Diese Datei ist die zentrale Komponente, die es erlaubt Konfigurationseinstellungen nach der Kompilierung durchzuführen. So ist es zum Beispiel möglich, die TCP/IP-Port-Einstellungen zu verändern, den Speicherort und -name der SQlite-Datenbank-Datei anzupassen, Log-Level zu verändern oder den Namen des Systemadministrators anzupassen.

Mehr Details dazu sind in der Installationsdokumentation festgehalten oder aber direkt als Kommentar in der Datei.

Als Beispiel die etwas gekürzte «appsettings.json»-Datei des «DataPersistence»-Moduls ohne Logging:

```
{
  // basic setup to run the DataPersistence module
  "DataPersistenceConfiguration": {
    "SystemName": "DataPersistence",
    "Version": "1.0.0.0",
    "ApiVersion": "v1",
    "ConnectionStrings": {
      // options are sqlite|inmemory-sqlite|inmemory
      "DataPersistenceDbConnectionType": "sqlite",
      "DataPersistenceDbConnection": "DataSource=DataPersistenceDb.db"
    }
  },
  // registered user with this address has all right and can promote other users
  "SystemAdmin": "admin@company.com",
  "ApplicationName": "Statistic Data Analysis, DP",
  "TermsOfService": "Release Candidate",
  "License": "For Internal Use Only",
  "Description": "Data Persistence Module",
  // port to access the system, must also be configured for AccessCoordination
  "Port": "5010",
  // log level options are Information|Warning|Error
  "Logging": {
    ...
  }
}
```

11.4 Datenbanken

11.4.1 AccessCoordinationDb

Die «AccessCoordinationDb»-Datenbank speichert folgende Datenmodelle:

- StatisticDataQueryParametersetDbModel (vgl. 9.6.4)
- StatisticDataTupleDbModels (vgl. 9.6.6)

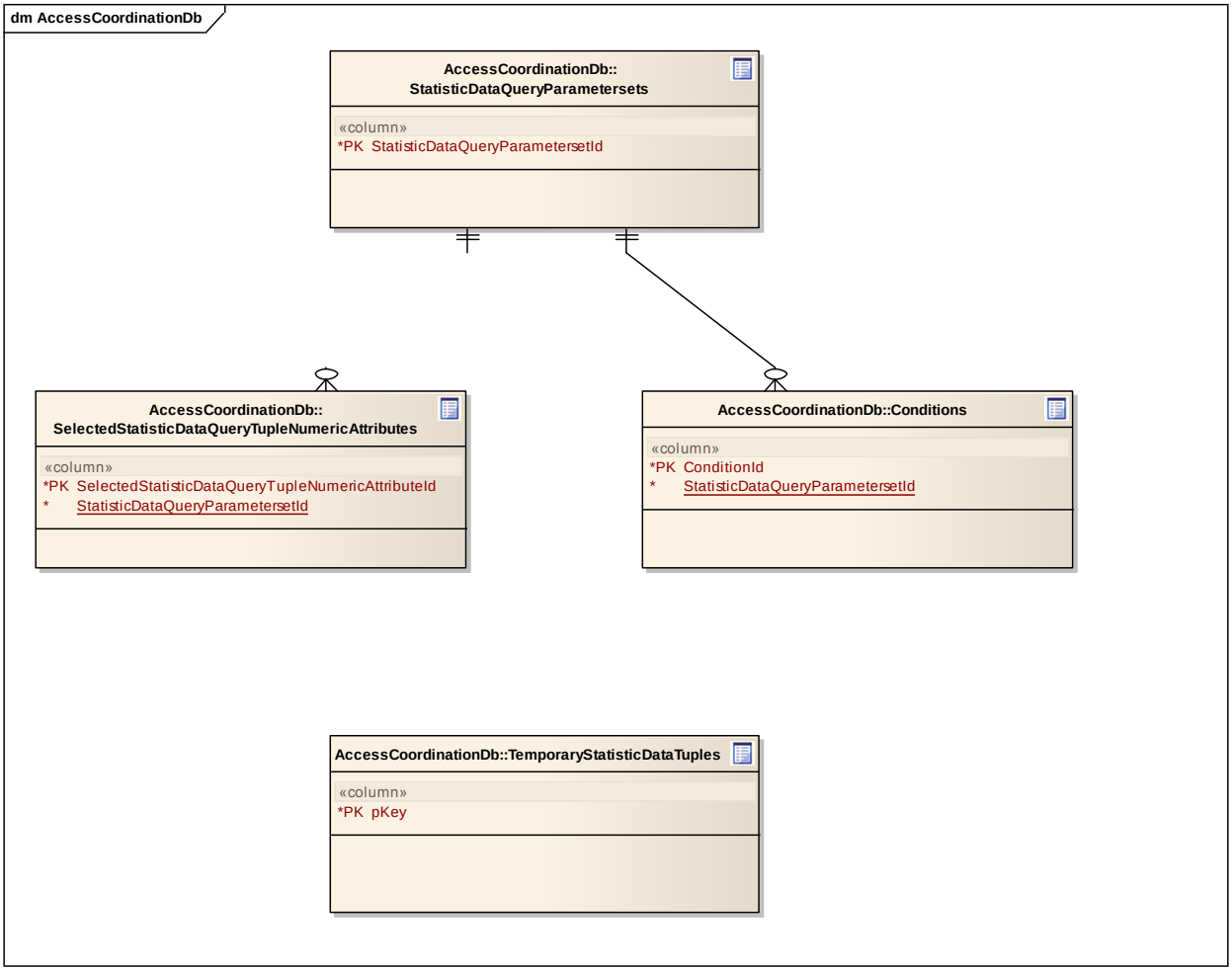


Abbildung 37: Datenbankschema der «AccessCoordinationDb»

11.4.2 DataPersistenceDb

Die «DataPersistenceDb»-Datenbank speichert folgende Datenmodelle

- StatisticDataActuator (vgl. 9.7.2)
- StatisticDataTupleLogDbModel (vgl. 0)

Folgende Grafik zeigt aus Platzgründen nur ein reduziertes Schema des «StatisticDataActuator» dargestellt. Weitere Tabellen wären analog zu «UserComment» mit «StatisticDataActuator» verknüpft.

(nicht öffentlich)

Abbildung 38: Reduziertes Schema des «StatisticDataActuator»

12 Testinfrastruktur

Details wie man in ASP.NET Core 2.0 testet, findet man wiederum in der Dokumentation des Frameworks:
Unittest: https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-dotnet-test

Integrationstest: https://docs.microsoft.com/en-us/aspnet/core/test/integration-tests

Nachfolgend ist jeweils an einem Beispiel erläutert, wie die jeweilige Teststufe implementiert wurde. Das Vorgehen bezüglich Testen ist im Testkonzept [4] festgehalten.

12.1 Unittest

Der Aufbau und Ablauf eines Unittests ist trivial. Die Test-Klasse erzeugt eine Instanz der zu testenden Klasse. Ruft die zu testende Methode auf und wertet deren Resultat aus.

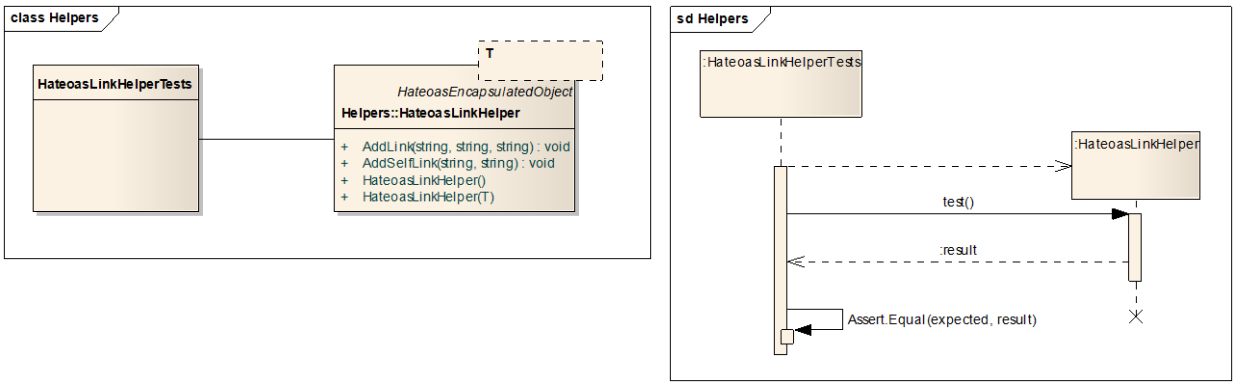


Abbildung 39: Aufbau eines Unittests

12.2 Komponententest

Im Unterschied zu einem einfachen Unittest hat beim Komponententest das «Device Under Testing» (DUT) Abhängigkeiten, die entweder mittels Mock abstrahiert werden müssen, oder aber in die Testumgebung eingebunden werden.

Im Beispiel verfügt die Testklasse über eine «ServiceCollection» mittels derer die verschiedenen Services registriert werden und dann dem DUT «StaticDataActuatorController» bei der Initialisierung mittels Konstruktor Injektion zur Verfügung gestellt werden. Somit wird die «Dependency Injection»-Unit des Frameworks (vgl. 8.4.4) nicht benötigt, was den Test wesentlich vereinfacht.

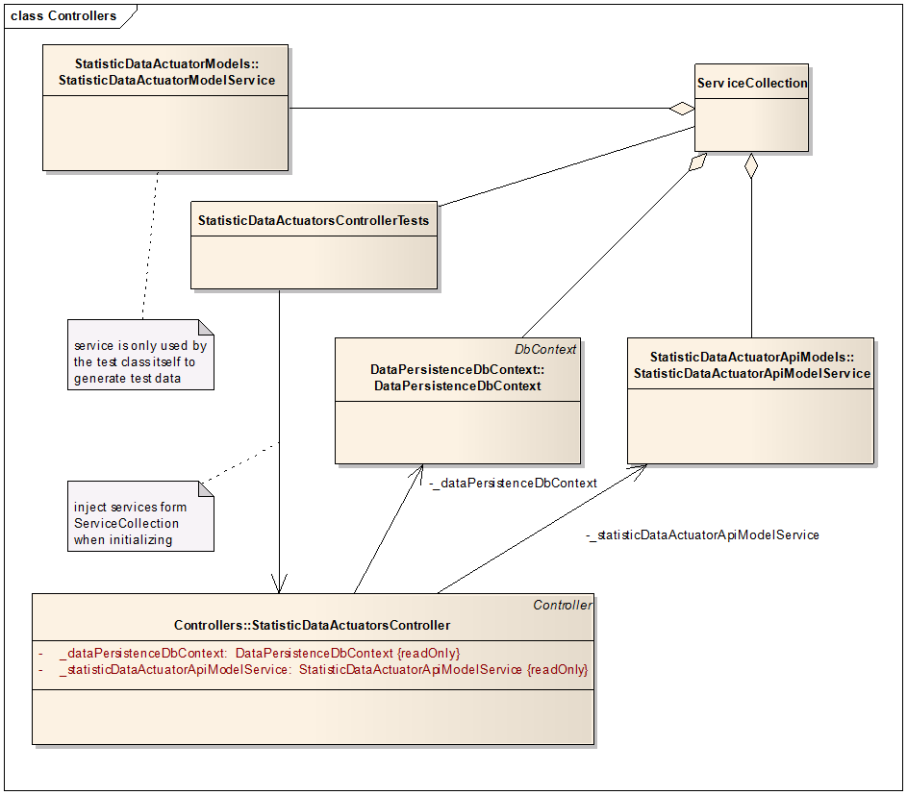


Abbildung 40: Aufbau eines Komponententests

12.3 Integrationstest

Für Integrationstest wird ein spezieller Test-Server verwendet (Microsoft.AspNetCore.TestHost). Dieser kann sehr schnell auf- und abgebaut werden, dadurch sind auch Integrationstests noch zeitlich gut zu beherrschen. Der Test-Server verwendet eine spezielle Middleware-Pipeline und eine In-Memory-Datenbank. Anfragen vom Client zum Server werden nicht über TCP/IP Socket Verbindungen abgehandelt, sondern werden direkt vom Framework verarbeitet.

Ein Testfall startet jeweils das «System Under Test», dabei wird ein Test-Server und ein Test-Client erzeugt. Durch Aufruf des Clients kann eine Anfrage an den Test-Server abgesetzt werden. Die Antwort kann danach im Testfall ausgewertet werden bevor der Testfall abgeschlossen und das «System Under Test» abgebaut wird.

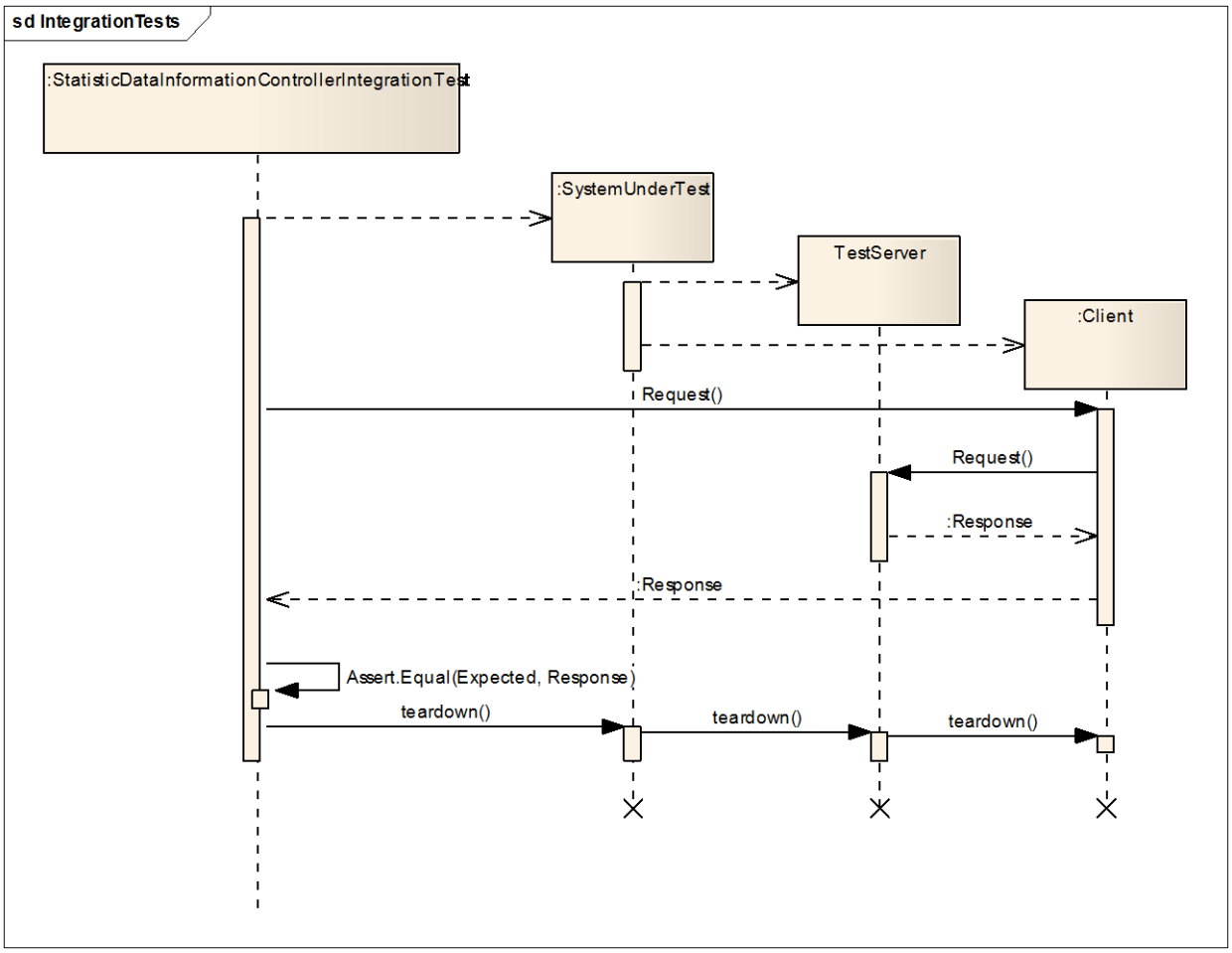


Abbildung 41: Aufbau eines Integrationstests

12.4 Systemtest

Die Infrastruktur für einen Systemtest ist etwas aufwendiger als für einen Integrationstest. Das Problem ist, dass das System zwei Webserver benötigt, die miteinander kommunizieren. Es wurde keine Möglichkeit gefunden, wie zwei spezielle Testserver für Integrationstests gleichzeitig betrieben werden könnten. Es wurde darum der Ansatz gewählt, dass das «DataPersistence»-Module als vollwertiger Server mit In-Memory-Datenbanken vom Framework gestartet wird.

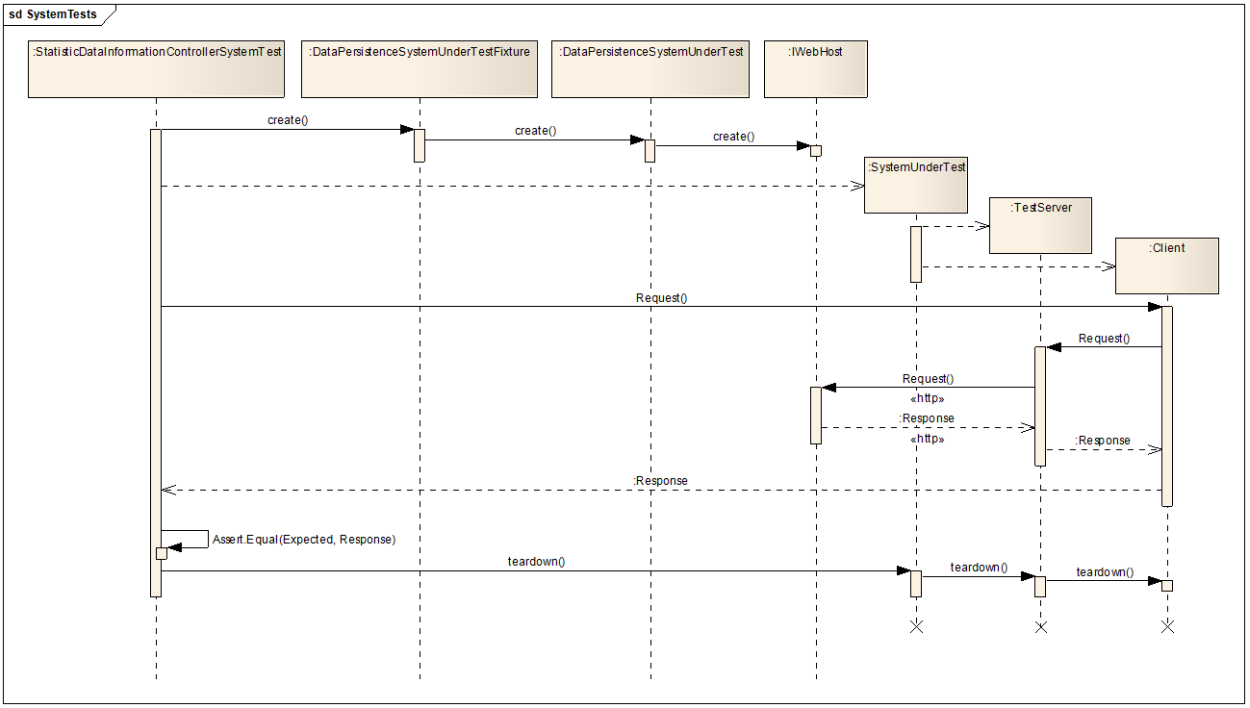


Abbildung 42: Aufbau eines Systemtests

Als erstes wird eine Instanz des «DataPersistence»-Modul gestartet. Diese Instanz empfängt Anfragen auf einem TCP/IP Port, genau gleich wie ein Produktivsystem. Als zweites wird analog dem Integrationstest ein Test-Server vom Typ «AccessCoordination» und ein Test-Client gestartet.

Der Test-Client wird nun benutzt um eine Anfrage an das «AccessCoordination»-Modul zu versenden. Dieses wiederum leitet die Anfrage zur Verarbeitung (über TCP/IP localhost:port) an das «DataPersistence» weiter. Die Antwort wird nun auf dem gleichen Weg zurück versendet und schliesslich ausgewertet.

Im Gegensatz zu den vorherigen Testsetups, wird das «DataPersistence»-Modul zwischen den Tests nicht abgebaut. Dies muss beim Schreiben der Tests berücksichtigt werden. Zudem ist der Aufbau des Testsystems ziemlich zeitaufwendig, die Anzahl der Systemtests sollte daher geringgehalten werden. Des Weiteren sollte beachtet werden, dass ein Systemtest nicht isoliert abläuft, da allenfalls andere Anwendungen auf dem Testsystem bereits den selben TCP/IP-Port verwenden könnten.

13 Toolchain

13.1 Enterprise Architect, Sparx Systems, v7.1.832

Graphische Dokumentation der Architektur

Bemerkung: Diese etwas ältere Version unterstützt nicht den vollen Sprachumfang von C#. Bevor der Quellcode importiert werden kann, sollten die kritischen Befehle «async» und «await» durch ein Batch-Skript auskommentiert werden. replace_async_await_for_EA.bat

Die Projektdatei befindet sich im Verzeichnis: 11_Architekturdokumentation

13.2 Swagger

Dokumentation der Schnittstelle

Bemerkung: Wie bereits in Kapitel 7 erwähnt, wird Swagger auch benützt, um eine eingebettete Dokumentation der Schnittstelle zu erzeugen.

Die statische Dokumentation befindet sich in den Verzeichnissen:

11_Architekturdokumentation\Swagger_Doc\AccessCoordination

11_Architekturdokumentation\Swagger_Doc\DataPersistence

13.3 SQliteBrowser

Version: v3.10.1

Verwaltung von SQlite Datenbanken

13.4 .NET Runtime und SDK

Runtime: v2.0.9

SDK: v2.1.202

13.5 Visual Studio 2017, Microsoft

Version: v15.7.3

IDE für Entwicklung

13.5.1 Zusätzliche Plug-Ins zu Visual Studio

Xunit Testrunner: v2.4.0

Visual Assist: v10.9.2283.2

SQlite/SQL Server Compact Toolbox: v4.7.534

14 Entwurfsentscheidungen

14.1 Abgrenzende Entscheidungen

14.1.1EA01	**Keine direkte Anbindung von Hardware**
Datum	11.06.2018
Entscheidungsträger	Teamleiter Projektleitung
Gründe	• Anbindung von Hardware ist schwierig und daher sehr zeitaufwendig.
Konsequenzen	• Legacy-System wird vorerst als Schnittstelle zwischen Hardware (z.B. Aktuatoren) und der «Statistik-Daten Auswertung» benutzt. • Legacy-System kann vorerst nicht durch die «Statistik-Daten Auswertung» ersetzt werden.
Alternativen	• Erste Versuche, Hardware mittels «serial-port-json-server» [*] zu integrieren, waren zwar erfolgreich, die Lösung von Problemen mit dem Protokoll und der Schnittstelle wären aber zu zeitaufwendig gewesen. [*] https://github.com/chilipeppr/serial-port-json-server

14.1.2EA02	**Event-Log Daten werden nicht in das System übernommen**
Datum	11.06.2018
Entscheidungsträger	Teamleiter Projektleitung
Gründe	• Event-Log Daten sind temporäre Daten, die bei der Entwicklung und Fehlersuche helfen. • Es gibt keinen Mehrwert, die Event-Log Daten in einer Datenbank zu sammeln.
Konsequenzen	• Anpassung der Anforderungen • Umbenennung des Systems in «Statistik-Daten Auswertung»
Alternativen	• n/a

14.1.3EA03	**Alarmierungsfunktion mittels E-Mail**
Datum	03.07.2018
Entscheidungsträger	Projektleitung
Gründe	• Funktionsweise und Ablauf der Alarmierung sind noch nicht abschliessend geklärt.
Konsequenzen	• Für einen Pilot-Betrieb werden Alarmierungen als E-Mails versendet. • Weitere Anforderungen können danach aufgenommen werden.
Alternativen	• Verknüpfung zu Jira

14.2 Funktionale Entscheidungen

14.2.1EF01	**Lose Kopplung zwischen «AccessCoordination» und «DataPersistence»-Modul**
Datum	10.05.2018
Entscheidungsträger	Projektleitung
Gründe	• Ziel ist es, ein lose gekoppeltes System zu entwickeln. Die Vorteile sind, dass die Module unabhängig entwickelt, erweitert, ausgetauscht, betrieben und verteilt werden können.
Konsequenzen	• Es werden zwei Software-Projekte geführt. • Die Datenmodelle müssen in beiden Modulen bekannt sein, das führt zur Quellcode-Verdoppelungen. • Systemtests müssen verteilte Systeme handhaben können.
Alternativen	• Entwicklung eines monolithischen Moduls. → Das Ziel der MAS-Arbeit ist aber die Entwicklung einer verteilten Architektur.

14.2.2EF02	**Änderung des zentralen Datenmodells vom Statistik-Daten-Tupel zum Statistik-Daten-Aktuator**
Datum	15.05.2018
Entscheidungsträger	Projektleitung
Gründe	• Der Betreuer hat vorgeschlagen, das bereits bestehende Datenmodell des Legacy-Systems zu überdenken. • Ein hierarchisches Datenmodell, dass den Aktuator als zentrales Element hat, modelliert die Realität besser. • Es kann einfacher erweitert werden. • Daten sind einfacher abfragbar, wenn sie bereits thematisch und nach Aktuator, in die Datenbank eingeordnet werden. • (vgl. 6.1.2)
Konsequenzen	• Umbau des «DataPersistence» Moduls notwendig. • Suchalgorithmus muss angepasst werden. • Einführung des Anwendungsfalls: UC50: Statistik-Daten-Aktuator bekannt geben (vgl. 3.4.2)
Alternativen	• Kein Umbau → «Design smell»

14.2.3EF03	**Benutzerverwaltung mit Authentifizierung und Autorisierung**
Datum	11.07.2018
Entscheidungsträger	Projektleitung
Gründe	• Für die Implementierung der Rollen musste eine Benutzerverwaltung mit Authentifizierung und Autorisierung eingeführt werden.
Konsequenzen	• Es gibt eine weitere Rolle: Der «Guest» ist ein Benutzer, der sich registriert hat, aber noch nicht auf die E-Mail zu Authentifizierung reagiert hat. • Das Software-Projekt muss mit einem neuen Template nochmals aufgesetzt werden. • Der E-Mail Versand für die Authentifizierung musste implementiert sein.
Alternativen	• Benutzerverwaltung ohne Authentifizierung und Autorisierung wäre hinfällig, da keine Berechtigungen verteilte werden könnten.

14.2.4EF04	**Programmierschnittstellen implementieren keine Authentifizierung und Autorisierung**
Datum	21.07.2018
Entscheidungsträger	Projektleitung
Gründe	• Aus Zeitgründen wird die Authentifizierung und Autorisierung nicht auf die Programmierschnittstellen erweitert.
Konsequenzen	• System API: Den Umsystemen kann keine Rolle zugewiesen werden. Sie müssen sich an die ihnen zugedachte Rolle halten. • interne API: Die Kommunikation zwischen «AccessCoordination» und «DataPersistence» Modul ist nicht verschlüsselt.
Alternativen	• Zeitplan lässt keine Alternativen zu.

14.2.5EF05	**Programmierschnittstellen um HATEOAS Links erweitern**
Datum	29.07.2018
Entscheidungsträger	Projektleitung
Gründe	• Ziel ist eine möglichst REST-konformen Schnittstelle. • (vgl. 8.2)
Konsequenzen	• System API: - Erweiterung der Schnittstelle um Links - Einführung des Anwendungsfalls: UC60: API-Informationen bekannt geben (vgl. 3.4.3) • interne API: - Erweiterung der Schnittstelle um Links (analog System API) - Das «AccessCoordination»-Modul bindet sich beim Startvorgang dynamisch an das «DataPersistence» Modul.
Alternativen	• ein tieferer REST-Level (vgl. 8.1)

14.3 Technologische Entscheidungen

14.3.1ET01	**Verwendung: ASP.NET Core 2.0 Framework**
Datum	05.03.2018
Entscheidungsträger	Projektleitung
Gründe	• Es bestehen bereits minimale Erfahrung mit dem Umgang des Web Frameworks. • C#-Knowhow innerhalb des Teams vorhanden. • Verfügbarkeit der Toolchain (13.4) und IDE (13.5) ist gegeben. • sehr gute Dokumentation • ASP.NET Core 2.1 ist erst seit Ende Mai 2018 verfügbar.
Konsequenzen	• Anwendung läuft nur auf Systemen, für die eine .NET Runtime besteht.
Alternativen	• Python als Programmiersprache → keine Erfahrung mit grossen Projekten nur Skripts • Django oder Flask als Web Framework → keine Erfahrung

14.3.2ET02	**Verwendung: Entity Framework Core als OR-Mapper**
Datum	05.03.2018
Entscheidungsträger	Projektleitung
Gründe	• Wird von Microsoft für neue ASP.NET Core Projekte empfohlen. • Unterstützt LINQ • Benötigt keine SQL-Experten-Kenntnisse. • Unterstützt die Code-First Herangehensweise. • Unterstützt das SQlite-Datenbanksystem. • Unterstützt das MySQL-Datenbanksystem. • Abstrahiert Datenbank → Datenbank kann einfach ausgetauscht werden. Z.B. Unittests
Konsequenzen	• Anwendung läuft nur auf Systemen, für die eine .NET Runtime besteht.
Alternativen	• Entity Framework 6 → Mix von Standard .NET und .NET Core ist nicht gewünscht. • Third-party OR-Mapper → keine Erfahrung

14.3.3ET03	**SQlite-Datenbanksystem**
Datum	10.05.2018
Entscheidungsträger	Projektleitung
Gründe	• Hat minimale externe Abhängigkeiten. • Wird von Entity Framework Core unterstützt. • Kann in die Anwendung integriert werden → keine Abhängigkeiten. • Wird in der Dokumentation von Microsoft neben den Microsoft-Produkten auch unterstützt.
Konsequenzen	• SQlite unterstützt nicht den vollen SQL-Sprachumfang. • Ist nicht so performant wie andere Datenbanksysteme (z.B. MySQL) • Anwendung und Datenbank können nicht verteilt werden.
Alternativen	• MySQL Datenbanksystem des Legacy-Systems → Keine Testumgebung verfügbar, Schreibzugriff auf Produktivsystem wäre sehr riskant. • Microsoft SQL Server oder anderes SQL-Datenbanksystem → keine Erfahrung • NoSQL-Datenbanksystem → keine Erfahrung

14.3.4ET04	**Verteilung als «Self-Contained Windows Service»**
Datum	30.07.2018
Entscheidungsträger	Projektleitung
Gründe	• Sämtliche Abhängigkeiten sind in einem Paket gebündelt. • «Self-Contained» Anwendungen sind portabel.
Konsequenzen	• Installationspakete werden sehr umfangreich. • Keine Integration in die Managementmöglichkeiten von Microsoft IIS.
Alternativen	• verwaltet durch Microsoft IIS • verwaltet durch Apache HTTP Server

15 Risiken und technische Schulden

Id	Risiko	Beschreibung
15.1 Funktionale Risiken und technische Schulden		
TS01	Erweiterung der Anforderungsspezifikation	Die Anwendung hat im Moment einen guten Stand und kann von den Stakeholdern benutzt werden. Die Pilotphase muss aber unbedingt ausgewertet werden, bevor die momentan bekannten technischen Schulden abgearbeitet werden. Es ist wahrscheinlich, dass der Pilotbetrieb neue Anforderungen ergeben wird, wodurch sich die Prioritäten ändern werden.
TS02	Alarmierung	Die Alarmierung ist momentan sehr einfach gehalten, da noch nicht alle Anforderungen geklärt werden konnten. Diese Anforderungen müsse im Pilotbetrieb noch erörtert werden. Die momentane Implementierung im «AccessCoordination»-Modul als E-Mail-Versand ist ein Proof of Concept.
TS03	Import von Daten	Die Funktionalität zum Import von Daten via Dateisystem wurde nur zu Test- und Demozwecken implementiert. Es hat sich aber gezeigt, dass eine solche Funktion durchaus wünschenswert wäre, falls die Daten aus einem Zip-Container mit Binärdaten extrahiert werden könnten. Der Aufwand dafür darf aber nicht unterschätzt werden, zudem ist zu klären, ob das Extrahieren der Daten auf Client- oder Server-Seite implementiert werden kann.
TS04	Hardware-Anbindung	Ursprünglich wurde eine direkte Anbindung von Hardware angedacht. Es wurde aber keine Möglichkeit gefunden, wie dies im Rahmen der Arbeit zu bewerkstelligen gewesen wäre. Diese Anforderung muss noch einmal überdacht und konkretisiert werden. (vgl. EA01)
15.2 Nicht-funktionale Risiken und technische Schulden		
TS05	Code-Verdoppelung	Die beiden Module wurden strikt getrennt entwickelt, dadurch wurden Quellcode-Teile kopiert. Vor allem bei den Schnittstellen und deren Datenmodellen gibt es Überschneidungen. Es sollte ein Weg gefunden werden, wie dieser Quellcode vereinheitlicht und zusammengeführt werden kann, z.B. mit einem GIT submodule auf Stufe Repo, oder die gemeinsame Quellcode-Basis wird in einem dritten GIT-Projekt verwaltet.
TS06	Benennung von Variablennamen	Das gewählte Schema führt zu sehr langen Variablennamen. Das Schema sollte daher nochmals kritisch beurteilt werden.
TS07	LINQ-Queries	Dynamische LINQ Queries sind nicht so einfach zu handhaben. Einerseits gibt es keine einfache Standardlösung, oder aber die Lösung kann nicht universell verwendet werden. z.B. FromSql funktioniert nicht mit der In-Memory-Datenbank: https://docs.microsoft.com/en-us/ef/core/querying/raw-sql Es gibt sicher bessere Lösungen als die im Quellcode gewählte. Hierzu sollten weitere Untersuchungen gemacht werden.
TS08	Testframework	Unittest haben zu viel repetitiven Quellcode. Das Framework sollte dahingehend erweitert werden, das mehrfach verwendete Funktionen zentral verwaltet werden.
TS09	Testframework	Das Testframework muss um eine Möglichkeit erweitert werden, dass Testdaten zentral erzeugt werden können. Momentan müssen sie oft «on-the-fly» erzeugt werden. Tests sollten parametrisierbar sein, so dass systematisch erzeugte Testdaten verwendet werden können.

Id	Risiko	Beschreibung
TS10	UI-Darstellung	Die Darstellung entspricht dem Standard-Beispiel Template von Microsoft. Das UI sollte übersichtlicher gestaltet werden, besonders im Hinblick darauf, dass sehr viele Aktuatoren verwaltet werden müssen.
TS11	Eingabe-Validierung	Bis jetzt werden Eingaben nur auf Typ, Länge und Bedingung geprüft. Es könnten aber auch noch die Abhängigkeiten zwischen den Eingaben geprüft werden. Zum Beispiel, wird in einer Bedingung eine Zahl geprüft, dann sollt die Eingabe auch auf numerische Eingabe prüfen auch wenn das Feld generisch ist und daher Text zu lässt.
TS12	Migration von Datenbanken	Sollte das Datenbankschema oft Änderungen unterliegen muss ein Weg gefunden werden, wie man das Problem umgehen kann, dass SQlite nur einen limitierten ALTER TABLE Befehl unterstützt. https://www.sqlite.org/lang_altertable.html"
TS13	Kommunikation zwischen den Modulen	Die Kommunikation zwischen "AccessCoordination" und "DataPersistence" ist essentiell. Bis jetzt wurden keine Anstrengungen unternommen, diese auf Optimierung zu untersuchen. Mögliche generelle Vorteile könnten Caching bringen. Zudem sollte jede Zugriffsfunktion nochmals dahingehend untersucht werden, ob nur die nötigsten Daten angefordert werden. z.B. sollten nicht alle Aktuatoren angefordert werden, wenn schlussendlich dem Client nur eine "Seite" dargestellt wird. (vgl. Pagination)
TS14	ASP.NET Core 2.0 End of Life	ASP.NET Core 2.1 wurde im Mai 2018 freigegeben, daher wurde es nicht für die Arbeit in Betracht gezogen. Leider ist der Vorgänger ASP.NET Core 2.0 bereits abgekündigt. Eine Migration sollte daher angestrebt werden.
TS15	Authentifizierung und Autorisierung wurden nicht auf Sicherheit geprüft	Authentifizierung und Autorisierung wurden gemäss den Vorgaben von Microsoft übernommen, dabei wurde aber kein Sicherheitsreview gemacht, um sicher zu gehen, dass bei der Implementation keine Fehler gemacht wurden.
TS16	Authentifizierung und Autorisierung Programmierschnittstellen aktivieren	Aus Zeitgründen wurde die Authentifizierung und Autorisierung nicht für die Programmierschnittstellen aktiviert. Auch die Kommunikation zwischen den Modulen ist nicht verschlüsselt. Diese sollte aber auch geschützt sein.
TS17	Logging/Serilog	Eine Funktionalität zum Loggen wurde integriert. Die Logging-Punkt sind aber zu wenig strukturiert gesetzt, so ist eine saubere Analyse des Laufzeitverhaltens nicht immer möglich.
TS18	Language localisation	Die Anwendung ist im Moment nur in englischer Sprache verfügbar. Die Anwendung soll dahingehend erweitert werden, dass die Sprache durch den Benutzer gewählt werden kann.

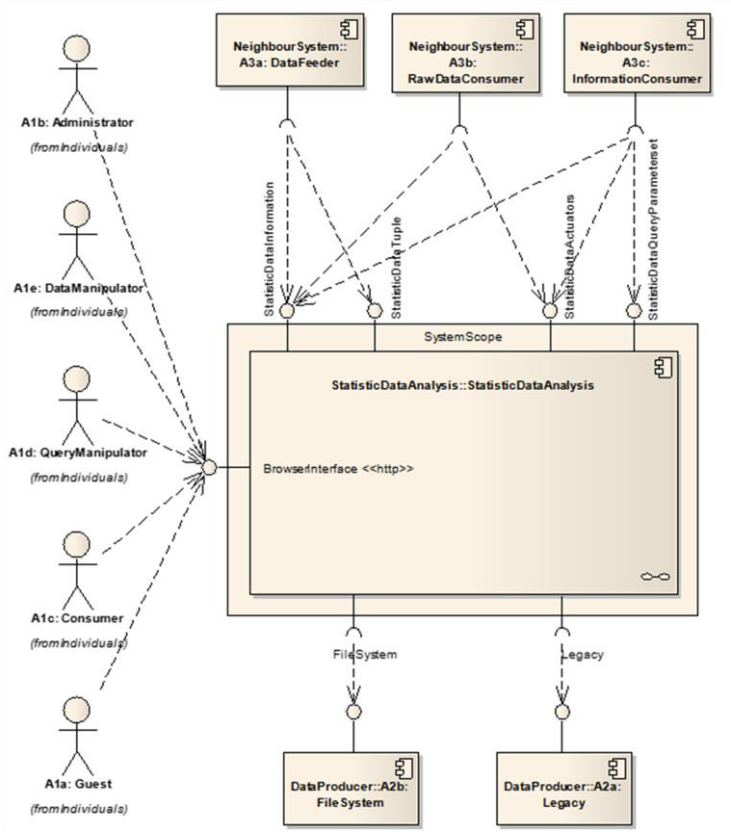
15.3 Infrastruktur Risiken und technische Schulden

TS19	Build Automatisierung	Die Anwendung muss ins kontinuierliche Build-System eingebunden werden. Nur so kann eine zielgerichtete Weiterentwicklung garantiert werden.
TS20	Versionsverwaltung	Im Rahmen der MAS-Arbeit wurde GIT als Versionsverwaltung benutzt, da eine verteilte Versionsverwaltung wesentliche Vorteile gegenüber einer zentralen hat, wenn der Zugriff zur Versionsverwaltung nicht immer gewährleistet ist. Schlussendlich muss die Anwendung aber in das Subversion-System der Abteilung integriert werden.
TS21	Release Version	Es ist noch nicht geklärt, welche Informationen auf der Frontseite dargestellt werden sollen.

16 Anhang

16.1 Statistik-Daten-Tupel

(nicht öffentlich)



Statistik-Daten Auswertung

Masterarbeit HSR-MAS-SE 2016

Testkonzept

Version:	003
Status:	RELEASE
Release Date:	26.08.2018
Classification:	PUBLIC

Document-ID: n/a
testkonzept.docx

Inhaltsverzeichnis

Abbildungsverzeichnis

Zu diesem Dokument

1.1 Zweck

Im Testkonzept werden Entscheidungen, Vorgehensweise, Planung und Abgrenzung der geplanten Testaktivitäten bezüglich der «Statistik-Daten Auswertung» Software definiert und festgehalten. Das Testkonzept gibt die Richtung und das Ziel der Testaktivitäten vor.

Im Testkonzept wird darauf verzichtet eine Einführung zum Projekt zu beschreiben. Projekthintergrund und Auftrag sind im Projektantrag [1] und der Anforderungsspezifikation [2] nachzulesen.

1.2 Gültigkeit

Dieses Dokument ist über die ganze Projektdauer gültig.

1.3 Verfügbarkeit des Dokumentes

Dieses Dokument befindet sich im Projektverzeichnis: 12_Testaktivitaeten

2 Weiterführende Informationen

2.1 Definitionen, Akronyme und Abkürzungen

Sämtliche Definitionen, Akronyme und Abkürzungen werden zentral in einem Projektglossar dokumentiert. Das Projektglossar wird kontinuierlich angepasst.

Das Projektglossar befindet sich im Projektverzeichnis: 09_Glossar

2.2 Ergänzende Dokumente

Ref.	Beschreibung
[1]	Projektantrag: „Statistik und Event-Log Auswertung“, VER011
[2]	Anforderungsspezifikation: «Statistik-Daten Auswertung, Anforderungsspezifikation», VER006
[3]	Architekturdokumentation: «Statistik-Daten Auswertung, Architekturdokumentation», VER006

2.3 Prozessbezogene Dokumente

Ref.	Beschreibung
[4]	Kursunterlagen MAS-SE «Software Testing»
[5]	Kursunterlagen MAS-SE «Qualitätsmanagement»
[6]	Essentials of software engineering, Third edition, Frank Tsui, Orlando Karam, Barbara Bernal ISBN: 978-1-4496-9199-8
[7]	Software-Prüfung, 6. Auflage, Karol Frühauf, Jochen Ludewig, Helmut Sandmayr ISBN: 978-3-7281-3059-4
[8]	Basiswissen Softwaretest, 3. Auflage, Andreas Spillner, Tilo Linz ISBN: 3-89864-358-1
[9]	https://docs.microsoft.com/en-us/aspnet/core/test/integration-tests

2.4 Versionshistorie

(nicht öffentlich)

2.5 Verteilung

Ver.	Name	Rolle / Titel / Link
003	Abgabe MAS-Arbeit	Abgabe MAS-Arbeit

3 Test-/Prüfobjekte

3.1 Statische Objekte

Nr.	Objekt	Beschreibung	Prüfung
1	Projektantrag	Der Projektantrag beschreibt das generelle Konzept, das Einsatzgebiet und die Motivation für die Entwicklung des Produkts.	Nein
2	Projektleitdokumentation	Der Zweck der Projektleitdokumentation ist das Projekt zu definieren, um eine Basis für das Projektmanagement zu bilden und eine Bewertung des Projekterfolges zu ermöglichen.	Ja
3	Anforderungsspezifikation	Die Anforderungsspezifikation beschreibt die Abgrenzung zwischen „Statistik-Daten Auswertung“ Produkt und Umsystemen, Anwendungsfälle, die logischen Datenstrukturen der Systemschnittstellen, die nichtfunktionalen Anforderungen und den Kern der für die Funktionserfüllung notwendigen Informationen und Abläufe.	Ja
4	Architekturdokumentation	Die Architekturdokumentation beschreibt Sachverhalte, Zusammenhänge und Voraussetzungen, die für das Verständnis des Systems oder einzelner Entwurfsentscheidungen relevant sind.	Ja
5	Testkonzept	Im Testkonzept werden Entscheidungen, Vorgehensweise, Planung und Abgrenzung der geplanten Testaktivitäten bezüglich des Produkts definiert und festgehalten.	Ja
6	Systemtest-Testspezifikation	Die Systemtest-Testspezifikation dokumentiert die Testfälle, mit denen das System getestet werden kann. Zudem dokumentiert es die Herleitung der benützten Äquivalenzklassen.	Ja
7	Testbericht	Zusammenfassung und Bestandesaufnahme der Testresultate am Ende der MAS-Arbeit.	Ja
8	Installations- und Betriebsdokumentation	Die Installations- und Betriebsdokumentation beschreibt die Voraussetzungen für eine erfolgreiche Installation und Betrieb des Produkts und deren Durchführung.	Ja
9	Benutzerdokumentation	Die Benutzerdokumentation ist eine Sammlung von Informationen für Benutzer zum sicheren und bestimmungsgemässen Umgang mit dem Produkt.	Ja
10	Statisches Software-Produkt	Das Software Produkt besteht aus einem statischen und einen dynamischen Teil. Beide Teile müssen auf ihre Weise geprüft werden. Zum statischen Teil gehört der eigentliche Quellcode, und die darin enthaltene Beschreibung des Quellcodes.	Nein

3.2 Dynamische Objekte

Es gibt nur ein dynamisches Objekt, das getestet wird: «Statistik-Daten Auswertung» Software

3.2.1 Kontext des «System Under Test»

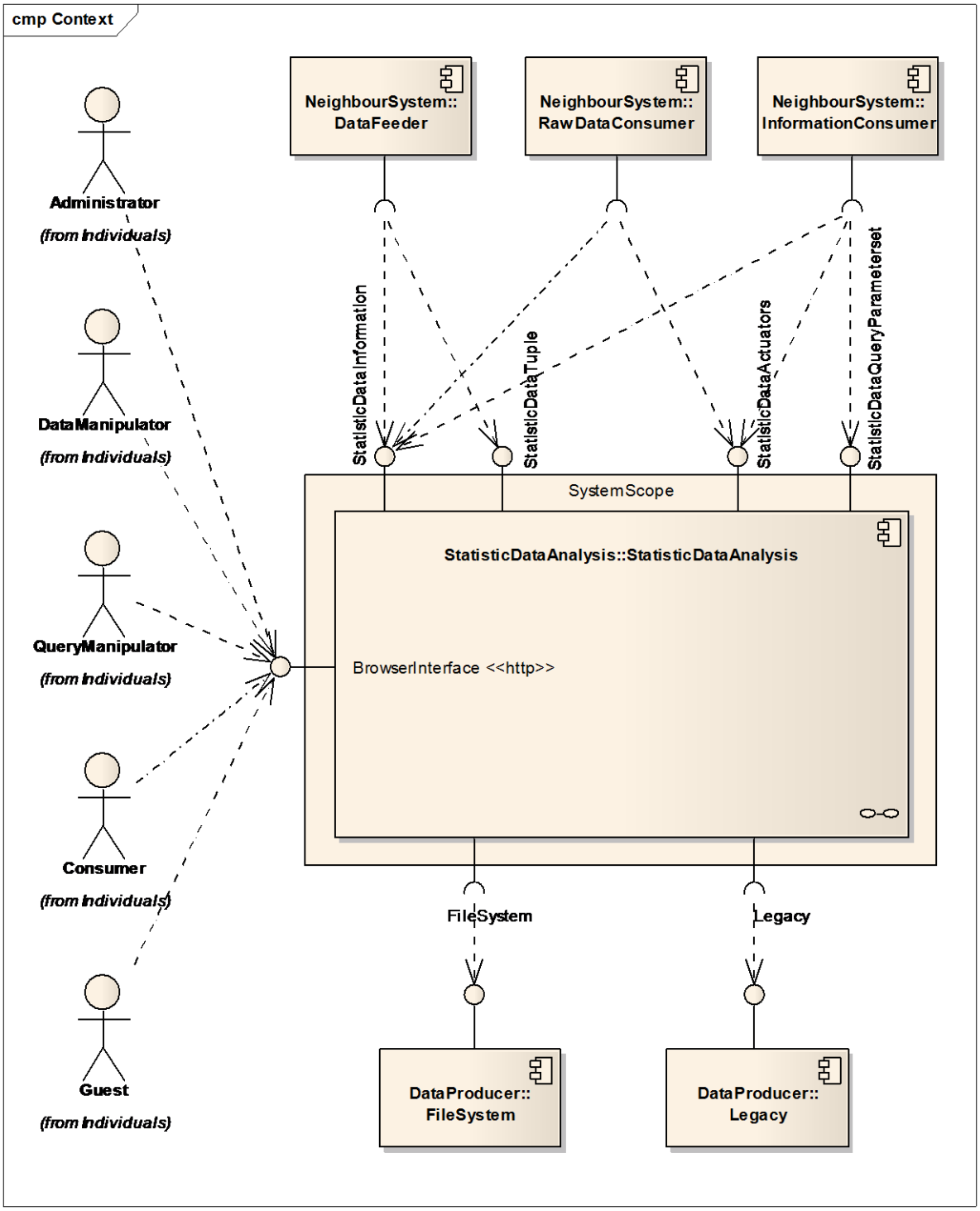


Abbildung 1: Kontextdiagramm

3.2.2 Schnittstellen

Ausführliche Beschreibungen zu den Systemschnittstellen sind in der Architekturdokumentation zu finden. Folgende Tabelle gibt einen kurzen Überblick zum Verständnis:

ID	Name	Beschreibung		Prüfung
P1	BrowserInterface	Typ: Implementierung: Protokoll: Hauptzweck:	Provided Interface, Akteure greifen darauf zu Browser Schnittstelle (Webinterface) http Über dieses Interface kann ein Akteur mittels Standard-Webbrowser (z.B. Mozilla Firefox) mit dem System interagieren.	Ja
P2	/api/StatisticDataInformation	Typ: Implementierung: Protokoll: Hauptzweck:	Provided Interface, externe Systeme greifen darauf zu REST-API Ressource http Über dieses Interface können externe Systeme Information zu weiteren REST-API Ressourcen abfragen.	Ja
P3	/api/StatisticDataTuple	Typ: Implementierung: Protokoll: Hauptzweck:	Provided Interface, externe Systeme greifen darauf zu REST-API Ressource http Über dieses Interface können externe Systeme neue Daten in das System einspeisen.	Ja
P4	/api/StatisticDataActuators	Typ: Implementierung: Protokoll: Hauptzweck:	Provided Interface, externe Systeme greifen darauf zu REST-API Ressource http Über dieses Interface können externe Systeme die Aktuatorenkenngrössen auslesen.	Ja
P5	/api/StatisticDataQueryParameterset	Typ: Implementierung: Protokoll: Hauptzweck:	Provided Interface, externe Systeme greifen darauf zu REST-API Ressource http Über diese Schnittstelle können externe Systeme einerseits Query-Parametersets verwalten und andererseits mittels dieser Query-Parametersets Statistik-Daten in Form von Selektionen, Reports oder Alarmierung abfragen.	Ja
R1	Legacy	Typ: Implementierung: Protokoll: Hauptzweck:	Required Interface, für Daten-Import benötigt Datenbank-Zugriff TCP/IP Über diese Schnittstelle greift das System auf das bestehende Legacy-Data-Repository zu.	Nein
R2	FileSystem	Typ: Implementierung: Protokoll: Hauptzweck:	Required Interface, für Entwicklung und Test benötigt .NET-Framework filesystem Über diese Schnittstelle werden Daten aus Dateien gelesen.	Nein

3.2.3 Gesamtsystem

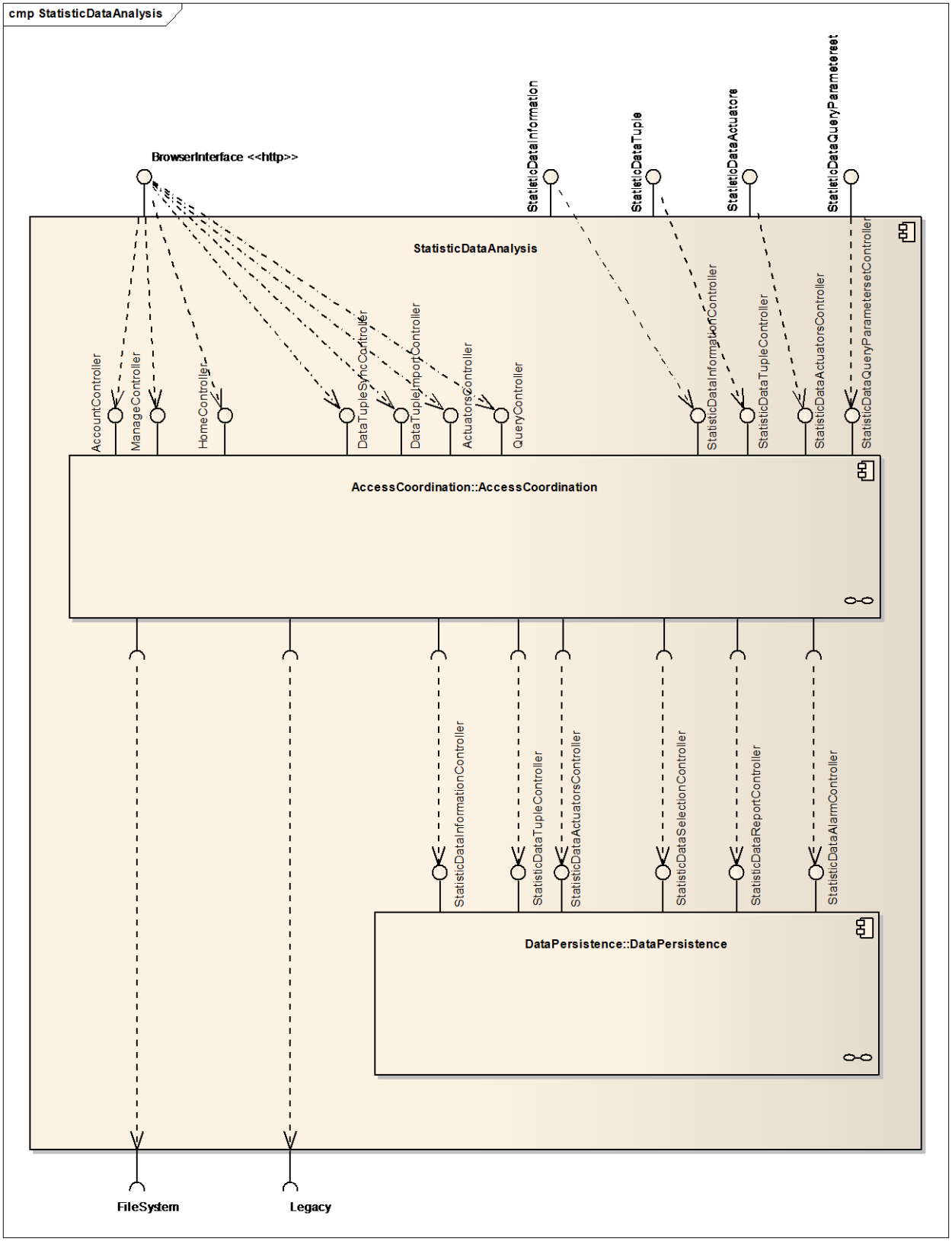


Abbildung 2: Gesamtsystem

3.2.4 Komponenten

Das System besteht im Wesentlichen aus zwei Blöcken, welche unabhängig voneinander getestet werden können.

ID	Name	Beschreibung	Prüfung
AC	AccessCoordination	Diese Komponente verwaltet und steuert den Zugriff zwischen der «DataPersistence» Komponente und den Umsystemen. Die Komponente stellte einerseits eine Web API zur Verfügung, die dem REST-Programmierparadigma folgt und andererseits kann mittels Webinterface auf das System zugegriffen werden.	Ja
DP	DataPersistence	Zentrale Komponente, welche die Statistik-Daten verwaltet, Reports generiert und Alarme auslöst.	Ja

4 Zu testende Leistungsmerkmale

4.1 Funktionale Anforderungen

UC10: Statistik-Daten-Selektion bekannt geben

UC11: Alle Statistik-Daten-Abfrage-Parametersets bekannt geben

UC12: Statistik-Daten-Abfrage-Parameterset entgegennehmen

UC20: Statistik-Daten-Report bekannt geben

UC40: Statistik-Daten-Alarmierung bekannt geben

UC30: Statistik-Daten-Tupel entgegennehmen

UC31: Modifiziertes Statistik-Daten-Tupel entgegennehmen

UC50: Statistik-Daten-Aktuator bekannt geben

UC60: API-Informationen bekannt geben

4.2 Nicht-funktionale Systemanforderungen

NF05: Das System prüft die eingegebenen Daten auf Vollständigkeit. Sind die eingegebenen Daten nicht vollständig, wird der Benutzer darüber informiert.

NF08/09: Eine Datenabfrage, Reportabfrage muss innerhalb 5 Sekunden eine gültige Antwort zurückgeben.

NF10: Der Zugriff auf die Statistik-Daten ist rollenbasiert und mittels Passwort geschützt.

NF14: Das System muss auf einem virtuellen Microsoft Windows Server (ab 2008) lauffähig sein.

5 Leistungsmerkmale, die nicht getestet werden

Aus Zeitgründen können folgende Anforderungen nicht geprüft werden.

5.1 Funktionale Anforderungen

UC100: Benutzerregistrierung entgegennehmen

UC101: Benutzer-Rollen-Wechsel entgegennehmen

5.2 Nicht-funktionale Systemanforderungen

NF01: Die Durchführung der Anwendungsfälle sollte für einen Mitarbeiter mittels Tutorial innerhalb 30 Minuten selbstständig erlernbar sein.

NF02: Die Reports sollen komplette Artefakte darstellen, d.h. keine Nachbearbeitung oder Konvertierungen benötigen. Die genaue Form des Artefakts (z.B. pdf, Excel-Dokument) muss noch geklärt werden.

NF03: Der Zugriff auf das System muss während den Bürozeiten (7:00 – 18:00) gewährleistet sein.

NF04: Das System muss von 5 Benutzern gleichzeitig benutzt werden können.

NF06: Die Fehlertoleranz des Systems gemessen in MTBF (Mean Time Between Failure) soll mindestens 30 Tage im ersten Betriebsjahr sein, danach ist eine Reevaluation notwendig.

NF07: Statistik-Daten, die vom System geprüft und verarbeitet wurden, müssen nach einem Betriebsausfall rekonstruierbar sein, sofern die Randbedingung geben ist, dass ein allfälliges Datenbank- und Betriebssystem gegen einen Betriebsausfall gesichert ist.

NF11: Das System darf nur von Mitarbeitern bedient und benutzt werden.

NF12: Das System darf nur in einem Netzwerk betrieben werden, welches vor Fremdzugriffen geschützt ist.

NF13: Änderungen am System müssen innerhalb eines Arbeitstages verbreitet werden können.

NF15: Für die Wartung des Systems dürfen nicht mehr als 12 Arbeitstage pro Jahr aufgewendet werden.

6 Testarten

6.1 Durchgeführte Tests

- Funktionale Tests
- Schnittstellen Tests

6.2 Nicht durchgeführte Tests

Aus Zeitgründen können folgende Testarten nicht berücksichtigt werden.

- Usability Tests
- Reliability Tests
- Third Party Komponenten Tests
- Installation Tests
- Performance Tests
- Long-Term Tests
- Security und Vulnerability Tests
- Failure and Recovery Tests

7 Risiken

In seiner ersten Version (MAS-Arbeit) übernimmt das System Daten des Vorgängersystems und bereitet diese Daten in Form von Selektionen, Reports und Alarmierungen auf. Die ursprünglichen Daten werden dabei kopiert aber nicht manipuliert. Das Risiko von Datenverlust ist daher äusserst gering.

In erster Linie muss bei der Prüfung sichergestellt werden, dass bei der Aufbereitung in Form von Selektionen, Reports und Alarmierungen der Daten die Abfrage- und Auswahlbedingungen korrekt ausgewertet werden. Datenabfragen müssen korrekte Ergebnisse liefern.

8 Testziele

Im Rahmen der MAS-Arbeit sind die Ziele bezüglich der Testtätigkeiten wie folgt definiert:

1. Funktionale Tests bezüglich der Datenaufbereitung in Form von Selektionen, Reports und Alarmierungen
2. Funktionale Tests bezüglich der Auswertung von Abfrage- und Auswahlbedingungen
3. Schnittstellen Tests bezüglich der Kommunikation zwischen System und Umsystemen
4. Entwicklung und Bereitstellung der Testumgebung für die jeweiligen Teststufen
5. Bestandesaufnahme der erfüllten Anforderungen bei Abschluss der MAS-Arbeit
6. Bestandesaufnahme der gefundenen Defekte
7. Testfälle sollten für Regressionstests verwendbar sein.
8. Testfälle sollten für die kontinuierliche Integration verwendbar sein.

9 Teststrategie

9.1 Statische Testobjekte

Bei den zu prüfenden statischen Testobjekten handelt es sich grösstenteils um Textdokumente. Diese werden mittels informellen Reviews geprüft und freigegeben.

Eine Ausnahme ist der Quellcode der Software, dieser kann zusätzlich auch mittels statischer Code-Analyse geprüft werden. Aufgrund fehlender Tool-Erfahrung wird die statische Code-Analyse aber nicht innerhalb der MAS-Arbeit durchgeführt.

Beschreibung	- Mittels informeller Reviews werden Dokumente auf Korrektheit und Konsistenz durch eine weitere Person geprüft.
Testziele	- Dokumente sollen korrekt und konsistent sein.
Teststartkriterien	- Sämtliche Abschnitte sind eingearbeitet. - Dokument wurde vom Autor für ein Review freigegeben.
Testendkriterien	- Review sämtlicher Abschnitte abgeschlossen. - Reviewbefunde im Dokument: xx_ListeDerBefunde.doc eingetragen oder als Word-Datei mit Änderungsverfolgung festgehalten.
Testabbruchkriterien	- keine
Abnahmekriterien	- Sämtliche Befunde wurden beurteilt, ob eine Nachbearbeitung notwendig ist. - Sämtliche Befunde, die einer Nachbearbeitung bedürfen, wurden im Dokument korrigiert.
Ausführung	- Microsoft Offce 365
Dokumentation	- Liste der Befunde oder Kopie des Originals mit Review-Kommentaren im Unterverzeichnis «review».
Ressourcen/Personal	- Reviewer (vgl. Projektleitdokumentation)

9.2 Dynamische Testobjekte

Bei dem zu prüfenden dynamischen Testobjekt handelt es sich um die «Statistik-Daten Auswertung» Software. Das System besteht im Wesentlichen aus zwei Blöcken, welche unabhängig voneinander getestet werden können.

Anzumerken ist, dass im Rahmen der MAS-Arbeit aus Zeitgründen auf die kontinuierliche Integration mittels Continua CI verzichtet wird. Ziel ist es aber, die automatisierten Tests zu einem späteren Zeitpunkt in das System für die kontinuierliche Integration zu übernehmen.

9.2.1 «DataPersistence»-Komponente

Gemäss Projektantrag [1] liegt der Fokus der MAS-Arbeit auf der zentral-verwalteten Komponenten des Systems. Daher ist die Teststrategie darauf ausgerichtet, dass die Testziele vor allem für die zentrale Komponente «DataPersistence» erfüllt werden.

Der Fokus liegt auf den selbst entwickelten Modulen, das heisst, die Anbindung der Datenbank und die reinen Daten-Objekte werden nicht explizit getestet.

Die folgende Tabelle und das Package Diagramm geben einen Überblick, welche Teile der «DataPersistence»-Komponente getestet werden:

Wird getestet (Typ)	**Wird nicht explizit getestet (Typ)**
Controller (Klassen)	Configuration (Klassen)
Helpers (Klassen)	DataPersistenceDb (SQlite-Datenbank)
Services (Klassen)	DataPersistenceDbContext (Klassen)
StatisticDataActuatorsController (Schnittstelle)	Models (Klassen)
StatisticDataAlarmController (Schnittstelle)	
StatisticDataInformationController (Schnittstelle)	
StatisticDataReportController (Schnittstelle)	
StatisticDataSelectionController (Schnittstelle)	
StatisticDataTupleController (Schnittstelle)	

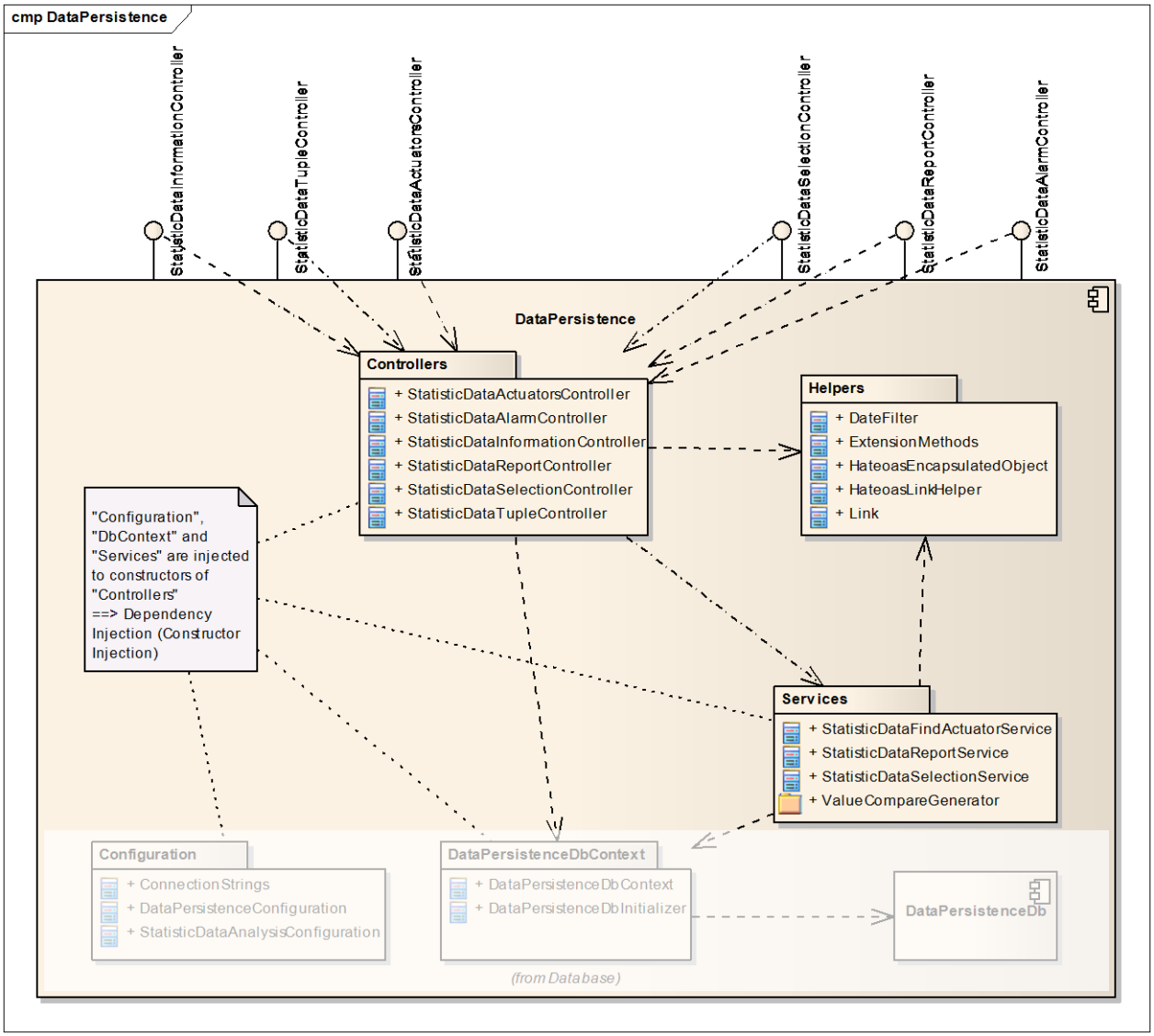


Abbildung 3: Package Diagramm «DataPersistence» (ohne «Models» Package)

9.2.2 «AccessCoordination»-Komponente

Die «AccessCoordination»-Komponente stellt sämtliche Schnittstellen zu den Umsystemen bereit. Aus Zeitgründen können nicht alle Funktionalitäten mit der gleichen Testtiefe getestet werden.

Der Fokus liegt auf den selbst entwickelten Modulen, das heisst, die Anbindung der Datenbank und die reinen Daten-Objekte werden nicht explizit getestet. Zudem werden die Webinterface-Komponenten nur im Zusammenhang mit manuellen Systemtests geprüft. Die Benutzerverwaltung wird nicht durch systematische Tests geprüft.

Die folgende Tabelle und das Package Diagramm geben einen Überblick, welche Teile der «AccessCoordination» Komponente getestet werden:

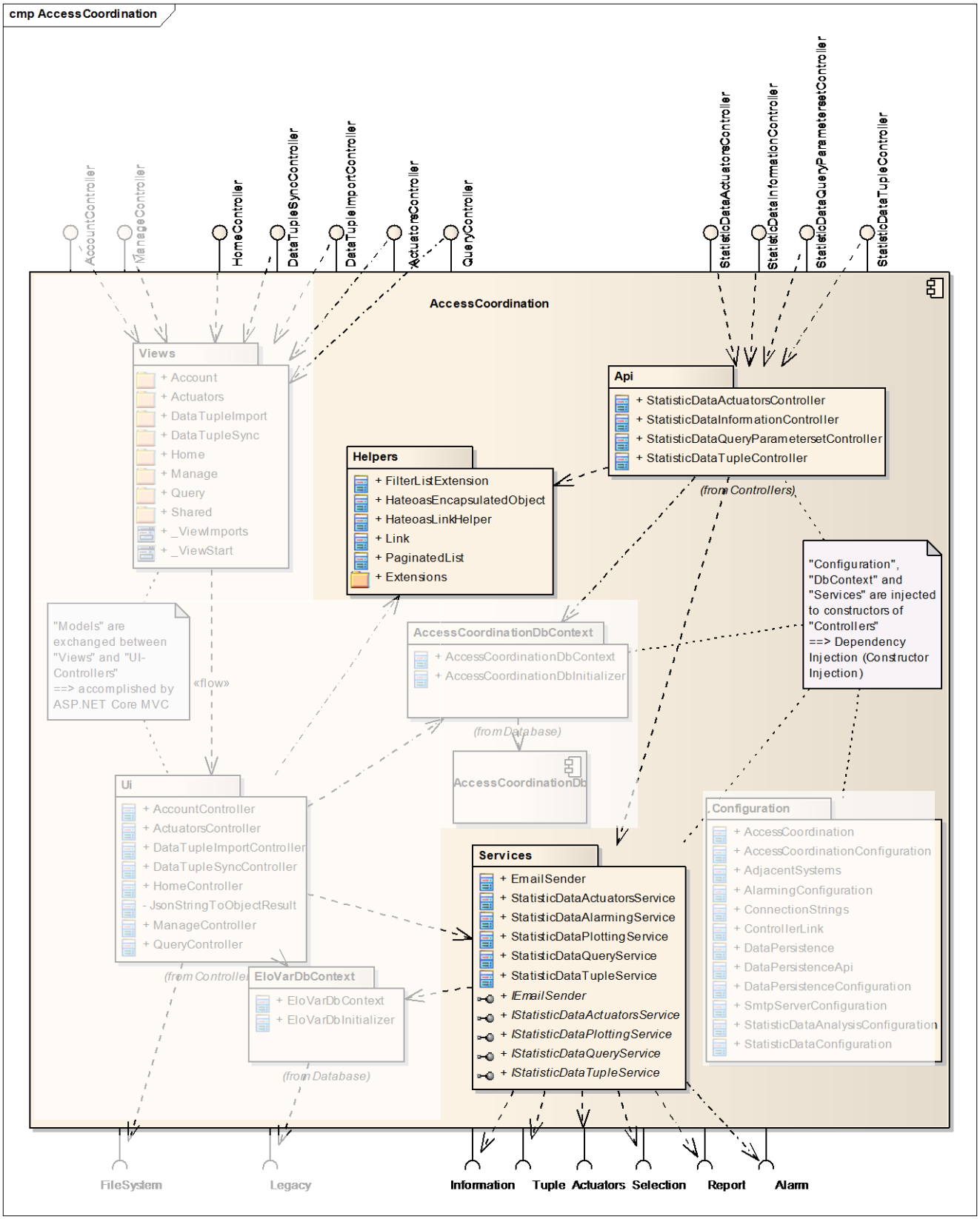


Abbildung 4: Package Diagramm «AccessController» (ohne «Models» Package)

Wird getestet (Typ)	**Wird nicht explizit getestet (Typ)**
API-Controllers (Klassen)	Configuration (Klassen)
Helpers (Klassen)	AccessCoordinationDb (SQlite-Datenbank)
Services (Klassen)	AccessCoordinationDbContext (Klassen)
StatisticDataActuatorsController (Schnittstelle)	EloVarDbContext (Klassen)
StatisticDataInformationController (Schnittstelle)	Models (Klassen)
StatisticDataQueryParameterController (Schnittstelle)	UI-Controllers (Klassen)
StatisticDataTupleController (Schnittstelle)	Views (cshtml-Dateien)
StatisticDataTupleController (Schnittstelle)	AccountController (Schnittstelle)
ActuatorsController (Schnittstelle)	ManageController (Schnittstelle)
QueryController (Schnittstelle)	HomeController (Schnittstelle)
	DataTupleSyncController (Schnittstelle)
	DataTupleImportController (Schnittstelle)

9.4 Teststufen

Das Ziel ist, dass automatisierte Unit-, Komponenten-, Integrations- und Systemtests garantieren, dass das System auch nach der MAS-Arbeit wartbar bleibt. Die Tests sollen bei zukünftigen Erweiterung des Systems für die Regression benützt werden können. Manuelle System- und Abnahmetest sollen vor allem zeigen, dass die definierten Anforderungen der MAS-Arbeit erfüllt wurden.

Die Teststrategie sieht vor, dass die Software «Statistik-Daten-Auswertung» gemäss den aus der Norm IEEE 829 bekannten Teststufen geprüft wird.

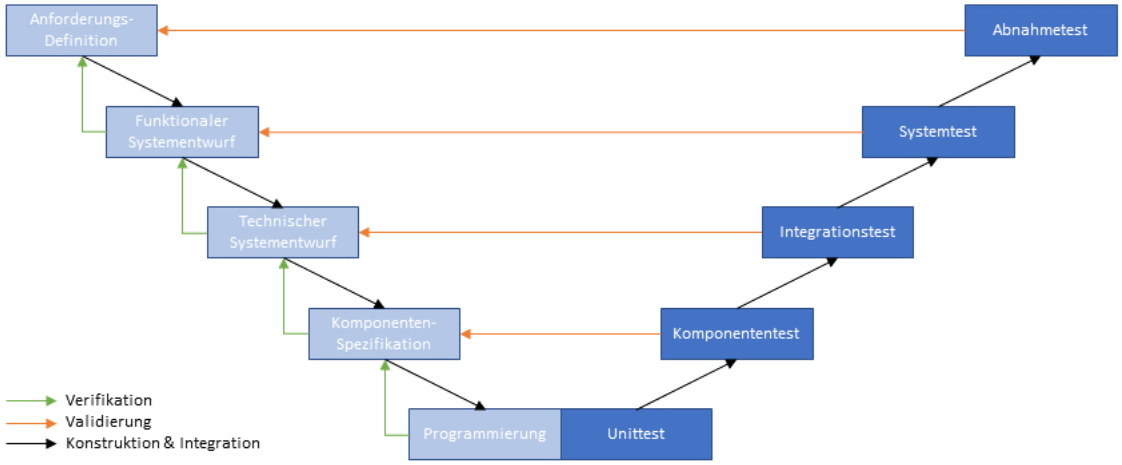


Abbildung 5: Allgemeines V-Modell mit Teststufen. Source: [8]

9.4.1 Unit- / Komponententests

Beschreibung	- Es werden einzelne Module und Funktionen der Komponenten getestet, die selbstständig entwickelt wurden. - Es sind automatisierte Tests
Testziele	- Bestätigung, dass Module und Funktionen sich so verhalten, wie vom Entwickler angedacht. Unittests helfen dem Entwickler beim Überprüfen seiner Arbeit. - Sollen für die Regression verwendet werden können.
Implementierung Unittest	- White-Box-Test - Unittests haben keine Abhängigkeiten zu anderen Units oder Komponenten - kurze Laufzeit - ASP.NET Core 2.0, Microsoft NET Test Sdk, xunit
Implementierung Komponententests	- White-Box-Test - Komponenten haben Abhängigkeiten, z.B. Dependency Injection, OR-Mapper-Zugriff. Diese Abhängigkeiten müssen «gemockt» werden. - kurze bis mittlere Laufzeit - ASP.NET Core 2.0, Microsoft NET Test Sdk, xunit, Moq/Moq4, EF Core 2.x In-Memory Database
Teststartkriterien	- Funktion, Algorithmus oder Komponente wurde durch den Entwickler getestet und freigegeben. - Das Testobjekt erfüllt gemäss Entwickler die Anforderungen (oder eine Teilmenge davon).
Testendkriterien	- Sämtliche Unit- / Komponententests werden erfolgreich ausgeführt.
Testabbruchkriterien	- Die Entwicklung eines einzelnen Tests wird unter-/abgebrochen, falls es nicht möglich ist, die Funktion für einen Test zu isolieren. - Die Entwicklung des Tests wird abgebrochen, wenn das Verhältnis zwischen Testlaufzeit und Zweckdienlichkeit durch den Testentwickler nicht positiv beurteilt werden kann.
Abnahmekriterien	- Sämtliche automatisierte Tests können ohne Benutzerinteraktion ausgeführt werden, und sämtliche Prüfkriterien werden erfüllt.
Ausführung	- Microsoft Visual Studio 2017, .NET Core 2.0 Runtime, xunit.runner.visualstudio
Dokumentation	- Testbeschreibung im Quellcode - Protokoll/Auswertung im Testreport
Ressourcen/Personal	- Entwickler (vgl. Projektleitdokumentation)

9.4.2 Integrationstests

Beschreibung	- Es wird das Zusammenspiel von zwei oder mehreren Komponenten getestet. Es wird dabei aber nicht das ganze System integriert und getestet. - Die Komponenten werden via deren Schnittstellen getestet.
Testziele	- Sollen gewährleisten, dass komplette Abläufe auch über Komponenten hinweg korrekte Ergebnisse liefern. - Sollen für die Regression verwendet werden können.
Implementierung	- Black-Box-Test - Komponenten haben Abhängigkeiten, z.B. Dependency Injection, OR-Mapper-Zugriff. Diese Abhängigkeiten werden bei Integrationstests im Gegensatz zu Komponententests nicht «gemockt», sondern integriert. - mittlere Laufzeit - ASP.NET Core 2.0, Microsoft NET Test Sdk, Microsoft.AspNetCore.TestHost, xunit, EF Core 2.x In-Memory Database TestHost ist ein In-Memory Webhost speziell für Integrationstests. Er kann schneller aufgesetzt und wieder abgebaut werden, da die Kommunikation nicht über Sockets geht: https://docs.microsoft.com/en-us/aspnet/core/test/integration-tests
Teststartkriterien	- Das Zusammenspielt der beteiligten Komponente wurde durch den Entwickler getestet und freigegeben. - Das Testobjekt erfüllt gemäss Entwickler die Anforderungen (oder eine Teilmenge davon). - Unit- und Komponententest der beteiligten Komponenten (vgl. 9.4.1) sind erfolgreich abgenommen.
Testendkriterien	- Sämtliche Integrationstests werden erfolgreich ausgeführt.
Testabbruchkriterien	- Die Entwicklung eines Tests wird unter-/abgebrochen, falls es nicht möglich ist, die Funktion für einen Test zu isolieren. - Die Entwicklung des Tests wird abgebrochen, wenn das Verhältnis zwischen Testlaufzeit und Zweckdienlichkeit durch den Testentwickler nicht positiv beurteilt werden kann.
Abnahmekriterien	- Sämtliche automatisierte Tests können ohne Benutzerinteraktion ausgeführt werden, und sämtliche Prüfkriterien werden erfüllt.
Ausführung	- Microsoft Visual Studio 2017, .NET Core 2.0 Runtime, xunit.runner.visualstudio
Dokumentation	- Testbeschreibung im Quellcode - Protokoll/Auswertung im Testreport
Ressourcen/Personal	- Entwickler (vgl. Projektleitdokumentation)

9.4.3 Automatisierte Systemtests

Beschreibung	- Es wird das gesamte System gegen die funktionalen Anforderungen getestet. (Nicht-funktionale Anforderungen werden in den automatisierten Systemtests aus Zeitgründen nicht getestet.) - Das System wird via dessen Systemschnittstellen getestet.
Testziele	- Sollen gewährleisten, dass komplette Szenarien / Use Cases die erwarteten Ergebnisse liefern. - Überprüfung ob und welche funktionalen Anforderungen das System erfüllt. - Sollen für die Regression verwendet werden können.
Implementierung	- Black-Box-Test - Beide Hauptkomponenten werden voll integriert getestet, das heisst, dass zwei Webhosts gleichzeitig ausgeführt werden. Aufgrund der http-Kommunikation zwischen den beiden Komponenten muss die «DataPersistence»-Komponente als realer Webhost (Kestrel Web Server) ausgeführt werden, während die «AccessCoordination»-Komponente als TestHost ausgeführt wird. Für sämtliche Datenbanken werden In-Memory Datenbases verwendet. - lange Laufzeit - ASP.NET Core 2.0, Microsoft NET Test Sdk, Microsoft.AspNetCore.TestHost, xunit, EF Core 2.x In-Memory Database, Mircosoft Kestrel Web Server
Teststartkriterien	- Das Zusammenspiel der beiden beteiligten Komponenten «AccessCoordination» und «DataPersistence» wurde durch den Entwickler getestet und freigegeben. - Das Testobjekt erfüllt gemäss Entwickler die Anforderungen (oder eine Teilmenge davon). - Unit-, Komponenten und Integrationstest der beteiligten Komponenten (vgl. 9.4.1/9.4.2) sind erfolgreich abgenommen.
Testendkriterien	- Sämtliche Systemtests werden erfolgreich ausgeführt.
Testabbruchkriterien	- Die Entwicklung des Tests wird abgebrochen, wenn das Verhältnis zwischen Testlaufzeit und Zweckdienlichkeit durch den Testentwickler nicht positiv beurteilt werden kann.
Abnahmekriterien	- Sämtliche automatisierte Tests können ohne Benutzerinteraktion ausgeführt werden, und sämtliche Prüfkriterien werden erfüllt.
Ausführung	- Microsoft Visual Studio 2017, .NET Core 2.0 Runtime, xunit.runner.visualstudio
Dokumentation	- Testbeschreibung im Quellcode - Protokoll/Auswertung im Testreport
Ressourcen/Personal	- Entwickler (vgl. Projektleitdokumentation)

9.4.4 Manuelle Systemtests

Beschreibung	- Es wird das gesamte System gegen die funktionalen Anforderungen getestet. Die manuellen Systemtests ergänzen die automatisierten Systemtests. - Das System wird nur via Webinterface geprüft.
Testziele	- Sollen gewährleisten, dass komplette Szenarien / Use Cases die erwarteten Ergebnisse liefern. Überprüfung ob und welche funktionalen Anforderungen das System erfüllt. - Es werden primär die Szenarien geprüft, für die es keine automatisierten Systemtests gibt.
Implementierung	- Black-Box-Test - Beide Hauptkomponenten werden voll integriert getestet, das heisst, beide beteiligten Komponenten «AccessCoordination» und «DataPersistence» werden als Testinstallation ausgeführt. Im Gegensatz zu den automatisierten Systemtests, werden bei diesen Testfällen produktive Web Server, Datenbanken und Testdaten verwendet.
Teststartkriterien	- Das Zusammenspiel der beiden beteiligten Komponenten «AccessCoordination» und «DataPersistence» wurde durch den Entwickler getestet und freigegeben. - Das System erfüllt gemäss Entwickler die Anforderungen. - Unit-, Komponenten, Integrations und automatisierte Systemtests der beteiligten Komponenten (vgl. 9.4.1 / 9.4.2 / 9.4.3) sind erfolgreich abgenommen. - Der Testfall für den jeweiligen manuellen Test ist freigegeben. - Das System ist betriebsbereit und mit Testdaten aufdatiert.
Testendkriterien	- Sämtliche Tests gemäss Testfall sind durchgeführt. - Befunde sind im Protokoll schriftlich festgehalten.
Testabbruchkriterien	- Das System ist nicht betriebsbereit, oder der Betrieb kann nicht für die Dauer des Tests aufrechtgehalten werden.
Abnahmekriterien	- Sämtliche Befunde wurden beurteilt, ob eine Nachbearbeitung notwendig ist.
Ausführung	- Mozilla Firefox
Dokumentation	- Testfall-Beschreibung - Testprotokoll (Protokoll/Auswertung im Testreport)
Ressourcen/Personal	Entwickler (vgl. Projektleitdokumentation)

9.4.5 Abnahmetest

Beschreibung	- Es wird zum Ende der MAS-Arbeit eine Bestandesaufnahme gemacht. Dieser Test wird nur einmal durchgeführt. - Das System wird nur via Webinterface geprüft.
Testziele	- Soll zeigen, welche geforderten funktionalen Anforderungen das System erfüllt und welche nicht.
Implementierung	- Black-Box-Test - Das System wird in der Produktivumgebung getestet.
Teststartkriterien	- Aufgrund der Rahmenbedingungen dieses Projektes als MAS-Arbeit wird am Ende der Arbeit ein Abnahmetest/Schlussbewertung durchgeführt. - Der Testplan für den Abnahmetest ist freigegeben. - Das System ist betriebsbereit.
Testendkriterien	- Sämtliche Tests gemäss Testplan sind durchgeführt. - Befunde sind im Protokoll schriftlich festgehalten.
Testabbruchkriterien	- keine
Abnahmekriterien	- Sämtliche Befunde wurden schriftlich festgehalten.
Dokumentation	- Testplan mit Testfällen - Testprotokoll (Protokoll/Auswertung im Testreport)
Ressourcen/Personal	- Auftraggeber, Entwickler (vgl. Projektleitdokumentation)

9.5 Übersicht Testobjekte und Testtyp

Testobjekt	Testtyp						
	Informelle Reviews	Unittests	Komponententests	Integrationstests	Automatisierte Systemtests	Manuelle Systemtests	Abnahmetests
9.5.1 Statische Objekte (3.1)							
Projektantrag							
Projektleitdokumentation	■						
Anforderungsspezifikation	■						
Architekturdokumentation	■						
Testkonzept	■						
Systemtest-Testspezifikation	■						
Testbericht	■						
Installations- und Betriebsdokumentation	■						
Benutzerdokumentation	■						
Statisches Software-Produkt							
9.5.2 DataPersistence (3.2.4)							
Configuration (Klassen)				■	■	■	■
Models (Klassen)				■	■	■	■
Controller (Klassen)			■	■	■	■	■
Helpers (Klassen)		■	■	■	■	■	■
Services (Klassen)		■	■	■	■	■	■
StatisticDataActuatorsController (Schnittstelle)				■	■	■	■
StatisticDataAlarmController (Schnittstelle)				■	■	■	■
StatisticDataInformationController (Schnittstelle)				■	■	■	■
StatisticDataReportController (Schnittstelle)				■	■	■	■
StatisticDataSelectionController (Schnittstelle)				■	■	■	■
StatisticDataTupleController (Schnittstelle)				■	■	■	■
DataPersistenceDb (SQlite-Datenbank)				■	■	■	■
DataPersistenceDbContext (Klassen)				■	■	■	■

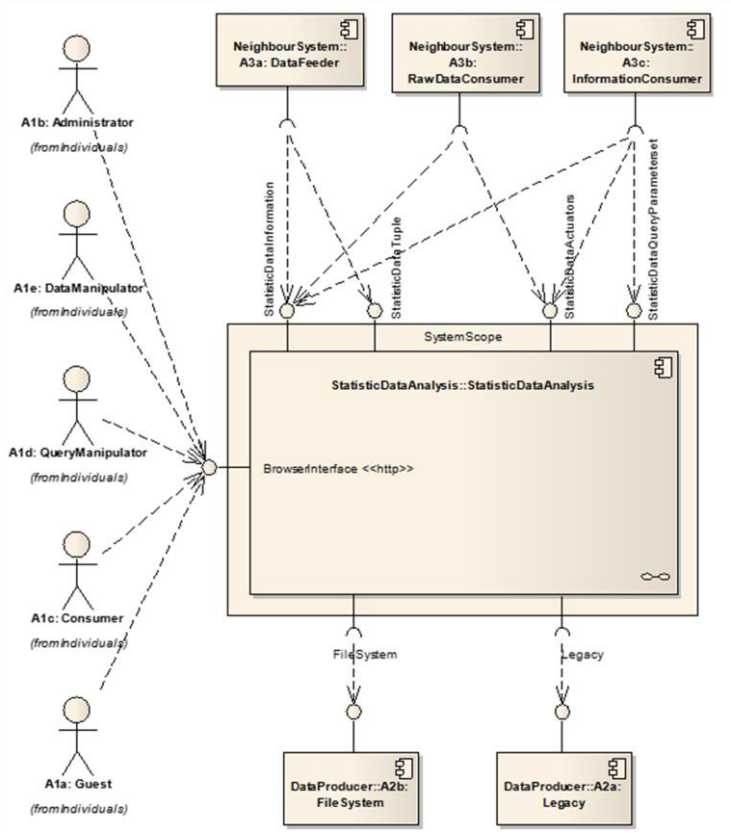
9.5.3 AccessCoordination (3.2.4)							
Configuration (Klassen)							
Models (Klassen)							
API-Controllers (Klassen)							
Helpers (Klassen)							
Services (Klassen)							
StatisticDataActuatorsController (Schnittstelle)							
StatisticDataInformationController (Schnittstelle)							
StatisticDataQueryParameterController (Schnittstelle)							
StatisticDataTupleController (Schnittstelle)							
AccessCoordinationDb (SQlite-Datenbank)							
AccessCoordinationDbContext (Klassen)							
EloVarDbContext (Klassen)							
UI-Controllers (Klassen)							
Views (cshtml-Dateien)							
AccountController (Schnittstelle)							
ManageController (Schnittstelle)							
HomeController (Schnittstelle)							
DataTupleSyncController (Schnittstelle)							
DataTupleImportController (Schnittstelle)							
ActuatorsController (Schnittstelle)							
QueryController (Schnittstelle)							

Expliziter Testfall

Impliziter Testfall, d.h. der Testfall betriff das Testobjekt, der Testfall ist aber nicht explizit für das Testobjekt

10 Zeit-/Arbeitsplan

Die Planung der Testaktivitäten ist ein Teil der Gesamtplanung der MAS-Arbeit und in der Projektleitdokumentation festgehalten. Sämtliche Testaktivitäten müssen bis zum Ende der MAS-Arbeit abgeschlossen sein.



Statistik-Daten Auswertung

Masterarbeit HSR-MAS-SE 2016

Testbericht

Version:	002
Status:	RELEASE
Release Date:	26.08.2018
Classification:	PUBLIC

Document-ID: n/a
testbericht.docx

Inhaltsverzeichnis

Zu diesem Dokument

1.1 Zweck

Dieser Testbericht ist eine Zusammenfassung, Bestandesaufnahme und Beurteilung der Testresultate am Ende der MAS-Arbeit.

1.2 Gültigkeit

Dieses Dokument ist über die ganze Projektdauer gültig.

1.3 Verfügbarkeit des Dokumentes

Dieses Dokument befindet sich im Projektverzeichnis: 12_Testaktivitaeten

2 Weiterführende Informationen

2.1 Definitionen, Akronyme und Abkürzungen

Sämtliche Definitionen, Akronyme und Abkürzungen werden zentral in einem Projektglossar dokumentiert. Das Projektglossar wird kontinuierlich angepasst.

Das Projektglossar befindet sich im Projektverzeichnis: 09_Glossar

2.2 Ergänzende Dokumente

Ref.	Beschreibung
[1]	Testkonzept: «Statistik-Daten Auswertung, Testkonzept», VER003
[2]	Architekturdokumentation: «Statistik-Daten Auswertung, Architekturdokumentation», VER006
[3]	Systemtest-Testspezifikation: «Statistik-Daten Auswertung, Systemtest-Testspezifikation», VER004

2.3 Prozessbezogene Dokumente

Ref.	Beschreibung
[4]	Kursunterlagen MAS-SE «Software Testing»
[5]	Kursunterlagen MAS-SE «Qualitätsmanagement»
[6]	Basiswissen Softwaretest, 3. Auflage, Andreas Spillner, Tilo Linz ISBN: 3-89864-358-1

2.4 Versionshistorie

(nicht öffentlich)

2.5 Verteilung

Ver.	Name	Rolle / Titel / Link
002	MAS-Arbeit	MAS-Arbeit

3 Testdurchführung

3.1 Übersicht Testobjekte

Nr.	Objekt	Version	Prüfung
1	Projektantrag	VER011	Nein
2	Projektleitdokumentation	VER005	Ja
3	Anforderungsspezifikation	VER006	Ja
4	Architekturdokumentation	VER006	Ja
5	Testkonzept	VER003	Ja
6	Systemtest-Testspezifikation	VER004	Ja
7	Testbericht	VER002	Ja
8	Installations- und Betriebsdokumentation	Keine Versionierung in Confluence	Ja
9	Benutzerdokumentation	Keine Versionierung in Confluence	Ja
11	AccessCoordination.exe	v1.0.0.0	Ja
11	DataPersistence.exe	v1.0.0.0 (Nachprüfung mittels v1.0.0.1)	Ja

3.2 Übersicht Testtyp

Gemäss Testkonzept Kapitel: 9.5

Testobjekt	Testtyp						
	Informelle Reviews	Unittests	Komponententests	Integrationstests	Automatisierte Systemtests	Manuelle Systemtests	Abnahmetests
3.2.1 Statische Objekte							
Projektantrag							
Projektleitdokumentation							
Anforderungsspezifikation							
Architekturdokumentation							
Testkonzept							
Testfall (manuelle Systemtests)							
Testbericht							
Installations- und Betriebsdokumentation							
Benutzerdokumentation							

3.2.2 DataPersistence							
Configuration (Klassen)							
Models (Klassen)							
Controller (Klassen)							
Helpers (Klassen)							
Services (Klassen)							
StatisticDataActuatorsController (Schnittstelle)							
StatisticDataAlarmController (Schnittstelle)							
StatisticDataInformationController (Schnittstelle)							
StatisticDataReportController (Schnittstelle)							
StatisticDataSelectionController (Schnittstelle)							
StatisticDataTupleController (Schnittstelle)							
DataPersistenceDb (SQlite-Datenbank)							
DataPersistenceDbContext (Klassen)							
3.2.3 AccessCoordination							
Configuration (Klassen)							
Models (Klassen)							
API-Controllers (Klassen)							
Helpers (Klassen)							
Services (Klassen)							
StatisticDataActuatorsController (Schnittstelle)							
StatisticDataInformationController (Schnittstelle)							
StatisticDataQueryParameterController (Schnittstelle)							
StatisticDataTupleController (Schnittstelle)							
AccessCoordinationDb (SQlite-Datenbank)							
AccessCoordinationDbContext (Klassen)							
EloVarDbContext (Klassen)							
UI-Controllers (Klassen)							
Views (cshtml-Dateien)							
AccountController (Schnittstelle)							
ManageController (Schnittstelle)							
HomeController (Schnittstelle)							
DataTupleSyncController (Schnittstelle)							
DataTupleImportController (Schnittstelle)							
ActuatorsController (Schnittstelle)							
QueryController (Schnittstelle)							

Expliziter Testfall

Impliziter Testfall, d.h. der Testfall betriff das Testobjekt, der Testfall ist aber nicht explizit für das Testobjekt

3.3 Änderung

3.3.1 AccessCoordination

Services (Klassen)							
StatisticDataActuatorsController (Schnittstelle)							
StatisticDataInformationController (Schnittstelle)							
StatisticDataQueryParameterController (Schnittstelle)							
StatisticDataTupleController (Schnittstelle)							

Die Services des «AccessCoordination»-Moduls sind für die Kommunikation zwischen «DataPersistence» und «AccessCoordination» zuständig. Es wurde in der gegebenen Zeit keine Möglichkeit gefunden, die Service-Klassen zu isolieren und die http-Kommunikation zu simulieren. Aus diesem Grund konnten auch die Schnittstellen nicht auf Stufe Integrationstest geprüft werden.

Auf Stufe Systemtests werden die Schnittstellen des Systems explizit mit automatisierten Test geprüft. Die Service-Klassen werden dabei implizit geprüft.

4 Testresultate

4.1 Reviews

Testobjekt	Testtyp					Resultat
	Informelle Reviews	Unittests	Komponententests	Integrationstests	Automatisierte Systemtests	
4.1.1 Statische Objekte						
Projektantrag						
Projektleitdokumentation						Review
Anforderungsspezifikation						Review
Architekturdokumentation						Review
Testkonzept						Review
Testfall (manuelle Systemtests)						Review
Testbericht						Review
Installations- und Betriebsdokumentation						Review
Benutzerdokumentation						Review

4.2 Automatisierte Tests

Testobjekt	Testtyp					Resultat
	Informelle Reviews	Unittests	Komponententests	Integrationstests	Automatisierte Systemtests	
4.2.1 DataPersistence						
Configuration (Klassen)						
Models (Klassen)						
Controller (Klassen)						17/17
Helpers (Klassen)						10/10
Services (Klassen)						9/9
StatisticDataActuatorsController (Schnittstelle)						2/2
StatisticDataAlarmController (Schnittstelle)						1/1
StatisticDataInformationController (Schnittstelle)						1/1
StatisticDataReportController (Schnittstelle)						1/1
StatisticDataSelectionController (Schnittstelle)						1/1
StatisticDataTupleController (Schnittstelle)						3/3
DataPersistenceDb (SQlite-Datenbank)						
DataPersistenceDbContext (Klassen)						
4.2.2 AccessCoordination						
Configuration (Klassen)						
Models (Klassen)						
API-Controllers (Klassen)						13/13
Helpers (Klassen)						11/11
Services (Klassen)						
StatisticDataActuatorsController (Schnittstelle)						2/2
StatisticDataInformationController (Schnittstelle)						2/2
StatisticDataQueryParameterController (Schnittstelle)						1/1
StatisticDataTupleController (Schnittstelle)						4/4
AccessCoordinationDb (SQlite-Datenbank)						
AccessCoordinationDbContext (Klassen)						
EloVarDbContext (Klassen)						
UI-Controllers (Klassen)						
Views (cshtml-Dateien)						

AccountController (Schnittstelle)						
ManageController (Schnittstelle)						
HomeController (Schnittstelle)						
DataTupleSyncController (Schnittstelle)						
DataTupleImportController (Schnittstelle)						
ActuatorsController (Schnittstelle)						
QueryController (Schnittstelle)						

4.3 Manuelle Systemtests

Testobjekt		
Manual-UC10	UC10: Statistik-Daten-Selektion bekannt geben	
Manual-UC20	UC20: Statistik-Daten-Report bekannt geben	
Manual-UC40	UC40: Statistik-Daten-Alarmierung bekannt geben	
Manual-UC11	UC11: Alle Statistik-Daten-Abfrage-Parametersets bekannt geben	
Manual-UC12	UC12: Statistik-Daten-Abfrage-Parameterset entgegennehmen	
Manual-UC30	UC30: Statistik-Daten-Tupel entgegennehmen	
Manual-UC31	UC31: Modifiziertes Statistik-Daten-Tupel entgegennehmen	vgl. 4.3.1
Manual-UC50	UC50: Statistik-Daten-Aktuator bekannt geben	

4.3.1 Nachprüfung

Der Testfall «Manual-UC31» hat bei der Erstprüfung nicht das gewünschte Ergebnis geliefert. Der Fehler konnte in der Nachfolgeversion des «DataPersistence»-Modul v1.0.0.1 behoben werden. Der Test wurde nochmals durchgeführt. Das Resultat entsprach den Erwartungen.

5 Abnahmetest

Für den Abnahmetest wurden die Testfälle der manuellen Systemtests verwendet. Im Gegensatz zu den Systemtests wurden die Testfälle aber auf dem Pilot-System mit produktiven Datenbanken ausgeführt.

5.1 Testfälle

Testobjekt		
Manual-UC10	UC10: Statistik-Daten-Selektion bekannt geben	
Manual-UC20	UC20: Statistik-Daten-Report bekannt geben	
Manual-UC40	UC40: Statistik-Daten-Alarmierung bekannt geben	
Manual-UC11	UC11: Alle Statistik-Daten-Abfrage-Parametersets bekannt geben	n/a
Manual-UC12	UC12: Statistik-Daten-Abfrage-Parameterset entgegennehmen	
Manual-UC30	UC30: Statistik-Daten-Tupel entgegennehmen	
Manual-UC31	UC31: Modifiziertes Statistik-Daten-Tupel entgegennehmen	
Manual-UC50	UC50: Statistik-Daten-Aktuator bekannt geben	

5.2 System Under Test

Release-Paket: StatisticDataAnalysisService_RC_20180823.zip

AccessCoordination
Version: 1.0.0.0
ApiVersion: v1

DataPersistence
Version: 1.0.0.1
ApiVersion: v1

5.3 Bemerkung

Beim Abnahmetest konnte der Systemtestfall «Manual-UC11» nicht durchgeführt werden.

Grund: Beim Produktivsystem ist es nicht möglich zu überprüfen, ob das System alle Statistik-Daten-Abfrage-Parametersets darstellt, die in der Datenbank gespeichert sind.

6 Bewertung

Das System wird für den Pilot-Betrieb freigegeben.

7 Nicht gelöste Probleme

Das Testframework für automatisierte Tests ist zuwenig ausgereift und muss erweitert werden. Diese technische Schuld ist bereits in der Architekturdokumentation festgehalten.

TS08	Testframework	Unittest haben zu viel repetitiven Code. Das Framework sollte dahingehend erweitert werden, das mehrfach verwendete Funktionen zentral verwaltet werden.
TS09	Testframework	Das Testframework muss um eine Möglichkeit erweitert werden, dass Testdaten zentral erzeugt werden können. Momentan müssen sie oft on-the-fly erzeugt werden. Test sollten parametrisierbar sein, so dass systematisch erzeugte Testdaten verwendet werden können.

8 Anhang

8.1 Testprotokolle

8.1.1 Reviews (4.1)

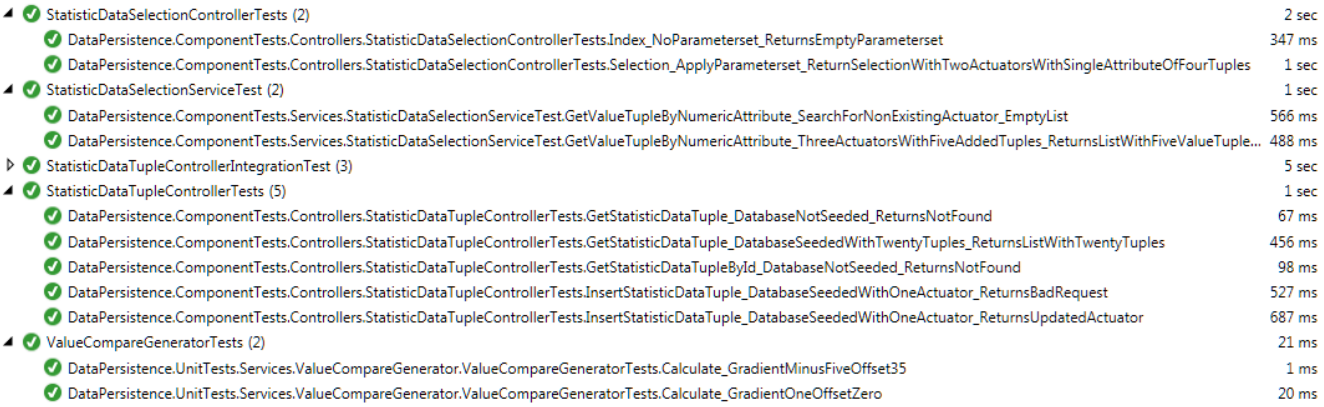
Zugehörige Review Dokumente befinden sich in einem separaten Unterverzeichnis im jeweiligen Verzeichnis des freigegeben Dokuments.

8.1.2 Automatisierte Tests (4.2)

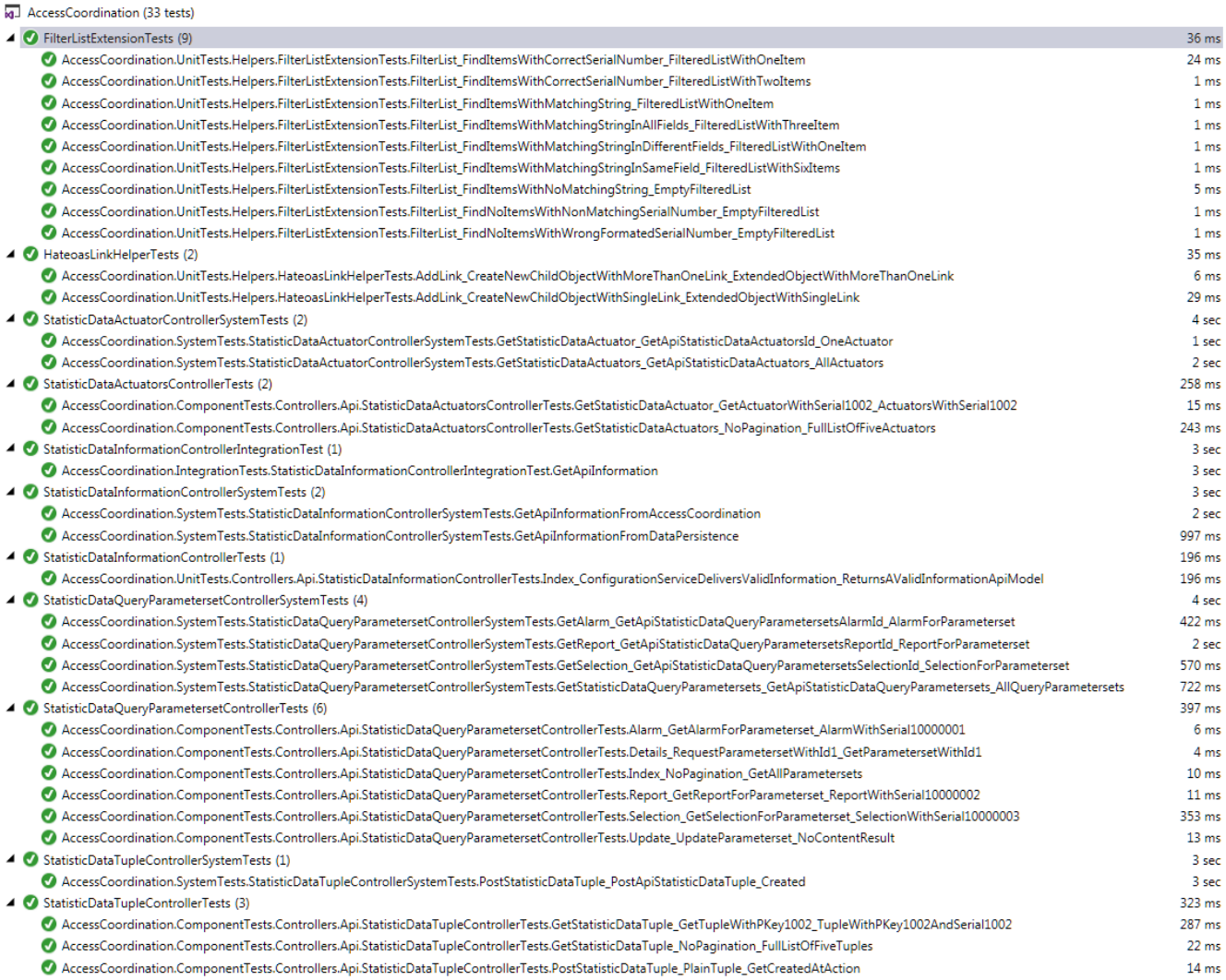
Ausgabe Xunit für DataPersistence

DataPersistence (45 tests)

DateFilterTests (8)	63 ms
DataPersistence.UnitTests.Helpers.DateFilterTests.Filter_EndDateEqualStartDate_ListWithOneValuesInOneDays	4 ms
DataPersistence.UnitTests.Helpers.DateFilterTests.Filter_EndDateLessThanStartDate_ListWithFourValuesInFourDays	1 ms
DataPersistence.UnitTests.Helpers.DateFilterTests.Filter_ListWithTenValuesInTenDays_ListWithFourValuesInFourDays	1 ms
DataPersistence.UnitTests.Helpers.DateFilterTests.Filter_ListWithTenValuesInTwentyDaysOddDays_ListWithAddedStartValue	4 ms
DataPersistence.UnitTests.Helpers.DateFilterTests.Filter_ListWithTenValuesInTwentyDaysOddDays_ListWithThreeValuesInFiveDays	1 ms
DataPersistence.UnitTests.Helpers.DateFilterTests.Filter_NoEndDate_ListOfSevenDays	1 ms
DataPersistence.UnitTests.Helpers.DateFilterTests.Filter_NoStartDate_ListOfFourDays	50 ms
DataPersistence.UnitTests.Helpers.DateFilterTests.Filter_NoStartNoEndDate_SameList	1 ms
HateoasLinkHelperTests (2)	20 ms
DataPersistence.UnitTests.Helpers.HateoasLinkHelperTests.AddLink_CreateNewChildObjectWithMoreThanOneLink_ExtendedObjectWithMoreThanOneLink	1 ms
DataPersistence.UnitTests.Helpers.HateoasLinkHelperTests.AddLink_CreateNewChildObjectWithSingleLink_ExtendedObjectWithSingleLink	19 ms
StatisticDataActuatorsControllerIntegrationTest (2)	2 sec
DataPersistence.IntegrationTests.StatisticDataActuatorsControllerIntegrationTest.GetStatisticDataActuator_GetInternalApiStatisticDataActuatorsId_ActuatorWithId	2 sec
DataPersistence.IntegrationTests.StatisticDataActuatorsControllerIntegrationTest.GetStatisticDataActuators_GetInternalApiStatisticDataActuators_AllActuators	449 ms
StatisticDataActuatorsControllerTests (5)	3 sec
DataPersistence.ComponentTests.Controllers.StatisticDataActuatorsControllerTests.DeleteStatisticDataActuator_DatabaseSeededWithTenActuators_ReturnDeleteActuator	576 ms
DataPersistence.ComponentTests.Controllers.StatisticDataActuatorsControllerTests.DeleteStatisticDataActuator_DatabaseSeededWithTenActuators_ReturnNotFound	289 ms
DataPersistence.ComponentTests.Controllers.StatisticDataActuatorsControllerTests.GetStatisticDataActuator_DatabaseSeededWithTenActuators_ReturnActuatorWithSerialFive	2 sec
DataPersistence.ComponentTests.Controllers.StatisticDataActuatorsControllerTests.GetStatisticDataActuator_DatabaseSeededWithTenActuators_ReturnNotFound	259 ms
DataPersistence.ComponentTests.Controllers.StatisticDataActuatorsControllerTests.GetStatisticDataActuators_DatabaseSeededWithTenActuators_ReturnsListWithTenActuators	316 ms
StatisticDataAlarmControllerIntegrationTest (1)	1 sec
DataPersistence.IntegrationTests.StatisticDataAlarmControllerIntegrationTest.Alarm_PostInternalApiStatisticDataAlarm_MatchingAlarm	1 sec
StatisticDataAlarmControllerTests (2)	2 sec
DataPersistence.ComponentTests.Controllers.StatisticDataAlarmControllerTests.Index_NoParameterset_ReturnsEmptyParameterset	12 ms
DataPersistence.ComponentTests.Controllers.StatisticDataAlarmControllerTests.Selection_ApplyParameterset_ReturnAlarmWithTwoActuatorsWithSingleAttributeOfFourTuples	2 sec
StatisticDataFindActuatorServiceTest (3)	2 sec
DataPersistence.ComponentTests.Services.StatisticDataFindActuatorServiceTest.FindMatchingActuators_IntersectionOfDeviceIdAndProductionLot_TwoActuators	1 sec
DataPersistence.ComponentTests.Services.StatisticDataFindActuatorServiceTest.FindMatchingActuators_UnionOfDeviceIdAndProductionLot_EightActuators	272 ms
DataPersistence.ComponentTests.Services.StatisticDataFindActuatorServiceTest.FindMatchingActuators_UnionOfTwoSerialNumbers_TwoActuators	265 ms
StatisticDataInformationControllerIntegrationTest (1)	4 sec
DataPersistence.IntegrationTests.StatisticDataInformationControllerIntegrationTest.GetApiInformation_GetInternalApi_InformationApiModel	4 sec
StatisticDataInformationControllerTests (1)	156 ms
DataPersistence.ComponentTests.Controllers.StatisticDataInformationControllerTests.Index_ConfigurationServiceDeliversValidInformation_ReturnsAValidInformationApiModel	156 ms
StatisticDataReportControllerIntegrationTest (1)	4 sec
DataPersistence.IntegrationTests.StatisticDataReportControllerIntegrationTest.Report_PostInternalApiStatisticDataReport_MatchingReport	4 sec
StatisticDataReportControllerTests (2)	709 ms
DataPersistence.ComponentTests.Controllers.StatisticDataReportControllerTests.Index_NoParameterset_ReturnsEmptyParameterset	18 ms
DataPersistence.ComponentTests.Controllers.StatisticDataReportControllerTests.Report_DatabaseSeededWithTenActuators_ReturnActuatorWithSerialFive	691 ms
StatisticDataReportServiceTest (2)	47 ms
DataPersistence.ComponentTests.Services.StatisticDataReportServiceTest.GenerateStatisticDataCompareTuples_CompareActualListToCompareListWithSameValues_SameValues	1 ms
DataPersistence.ComponentTests.Services.StatisticDataReportServiceTest.GenerateStatisticDataCompareTuples_TestWithoutAGivenStartDate_SameValues	46 ms
StatisticDataSelectionControllerIntegrationTest (1)	4 sec
DataPersistence.IntegrationTests.StatisticDataSelectionControllerIntegrationTest.Selection_PostInternalApiStatisticDataSelection_MatchingSelection	4 sec



Ausgabe Xunit für AccessCoordination



8.1.3 Manuelle Systemtests (4.3)

Die Testprotokolle der manuellen Systemtests sind im Dokument «Statistik-Daten Auswertung, Systemtest-Testspezifikation» [3] integriert.