Modellgetriebene homogenisierte Datenhaltung und Synchronisation mit vorhersagbaren Eigenschaften für plattformübergreifende Anwendungen

Förderkennzeichen: 01 S12012

AP 2

Dokumentation Validierungsergebnisse

Autor: E. Gailus (B2M)

Co-Autoren: G. Hübsch (CAS)

A. Schwichtenberg (CAS)

H. Groenda (FZI)

Fertiggestellt am: 30.06.2014

Schlagworte: Anforderungsanalyse, Validierungsergebnisse

Projektpartner

GEFÖRDERT VOM

Änderungshistorie

Version	Änderungen von	Datum	Anmerkung
1.0	Eric Gailus	06.06.2014	Initiale Fassung
1.1	Eric Gailus	12.06.2014	Kapitel 2.2 hinzugefügt
1.2	Eric Gailus	12.06.2014	Kapitel 3.2 und 3.5 hinzugefügt
1.3	Henning Groenda	16.06.2014	Kapitel 3.1, 3.2 und 3.6 hinzugefügt Kommentare Kapitel 2.2
1.4	Henning Groenda	18.06.2014	Review Anteil B2M
1.5	Gerald Hübsch	20.06.2014	Anteil CAS ergänzt. Review.
1.6	Henning Groenda	26.06.2014	Review

ToDos

Abschnitt	von	Beschreibung	Pritorität	Todo

Inhalt

Abbildungen

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

1 Einleitung

Bei der Entwicklung mobiler Anwendungen existiert die Problematik, dass für jede mobile Plattform andere Technologien und APIs zur Entwicklung genutzt werden und entsprechend unterschiedliche, plattformspezifische Funktionalitäten eingesetzt werden. Dies ist der Fall, da keine befriedigende allgemein akzeptierte technische Basis für alle Plattformen zur Verfügung steht, was den Entwicklungsaufwand bzw. den Aufwand der Portierung einer Anwendung für mehrere Plattformen ungemein erhöht.

Die Lösung dieser Problematik wurde im Rahmen des MOHITO-Projekts durch eine homogene und plattformübergreifende Datenhaltung und -Synchronisation verbunden mit dem Einsatz modellgetriebener Techniken angestrebt. Zu Beginn des Projektes wurde hierzu eine Anforderungsanalyse erstellt, die sowohl die Anforderungen an das zu entstehende MOHITO Framework als auch die anzubindenden Bestandsanwendungen und die zu verwendenden Technologien berücksichtigte. Ziel dieses Dokumentes ist die Validierung der definierten Anforderungen sowie die Beschreibung eventuell existierender Einschränkungen und der Umsetzung im Projekt sowie dem quelloffenen Demonstrator.

2 Anwendungsszenarios

2.1 Offlinefähiges xRM

In diesem Abschnitt wird die im Rahmen des MOHITO Projekt erreichte Umsetzung der im Ergebnisdokument 2.1 (Konsortium, D2.1 Anwendungsszenarios und Anforderungsanalyse, 2013) erfassten und umfangreich dokumentierten Anforderungen an ein mobiles, offlinefähiges xRM System tabellarisch dokumentiert. Dort findet sich ebenfalls eine detaillierte Beschreibung des Szenarios. Hier steht die Funktionalität des erstellten xRM Demonstrators im Mittelpunkt, der auf dem MOHITO Framework aufbaut. Die Umsetzung der an das MOHITO Framework selbst gestellten Anforderungen werden hingegen in Kapitel 3 Anforderungsanalyse diskutiert.

Nummer	**Anforderung**	**Gewichtung**	**Anmerkung zur Umsetzung**
	Generell		
A2.1.1-1	Offline Fähigkeit	Muss	Erfüllt. Der Demonstrator erlaubt eine Verwendung ohne eine Ver-

			bindung zum Server. Einzeldatensätze und Listen können in diesem Zustand aus der lokalen Datenbank des Clients abgerufen und editiert werden. Einzige Einschränkung ist die fehlende Möglichkeit, neue Datensätze Offline anzulegen. Dies ist systembedingt nicht möglich, da der CAS Open Server bei der Vergabe der Datensatzprimärschlüssel das führende System ist. Ein Umbau des Servers wurde mit dem Ergebnis geprüft, dass eine Vergabe der Primärschlüssel durch den Client durch eine Anpassung des Servers möglich wäre. Darauf wurde im Rahmen des Demonstrators verzichtet da die Anbindung des Produktivsystems CAS PIA das Ziel war.
A2.1.2-1	Automatisch alle relevanten Daten offline	Soll	Teilweise. Als Verfahren zur Relevanzbestimmung wurde die Verwendung von benutzerdefinierbarer Views (serverseitig ausgewertete Filter über den Datensätzen eines Benutzers, Client gibt SQL vor) diskutiert. Eine entsprechende Erweiterung des Demonstrators wurde bis zum Zeitpunkt der Erstellung dieses Dokument jedoch aus Komplexitätsgründen nicht durchgeführt. Die Voraussetzungen existieren jedoch sowohl im Open Server (Unterstützung für CAS SQL Queries über REST wurde in der Projektlaufzeit hinzugefügt) und im MOHITO Framework (Criteria API).
A2.1.2-2	Vermeidung von Datenverlust	Muss	Erfüllt. Datenverluste durch Verbindungsunterbrechungen während der Kommunikation des Demonstrators mit dem Server werden durch das lokale Vorhalten eine Kopie des übertragenen Datensatzes bis zu seiner vollständigen Speicherung auf dem Server verhindert.
A2.1.2-3	Für alle gängigen Betriebssysteme	Muss	Erfüllt. Der Demonstrator unterstützt das im Projekt als maßgeblich anerkannte Betriebssystem

			Android. Im Rahmen des Projekts wurde durch die CAS darüber hinaus eine Proof-of-Concept Implementierung für einen offlinefähigen iOS Client erstellt und präsentiert.
A2.1.2-4	Start ohne Internetverb. möglich	Muss	Erfüllt. Der Demonstrator lässt sich ohne Internetverbindung starten.
A2.1.2-5	Gesamte Funktionalität offline verfügbar	Kann	Teilweise. Ein Anlegen von Datensätzen beispielsweise ist Offline systembedingt noch nicht möglich.
A2.1.2-6	Offline nur Consumer-Funktion	Kann	Übertroffen. Datensätze sind Offline editierbar.
A2.1.2-7	Filter (meine, aktuelle)	Muss	Teilweise. Der Demonstrator ruft alle Datensätze eines Typs ab, auf die der Nutzer Zugriff hat („meine“). Eine Einschränkung auf aktuelle Datensätze wird nicht unterstützt, wäre technisch aber umsetzbar.
A2.1.2-8	Kontakte des Nutzers offline	Muss	Erfüllt. Die Kontakte eines Nutzers sind Offline zugreifbar.
A2.1.2-9	Termine des Nutzers offline	Muss	Erfüllt. Auf Offline-Zugriff auf Termine eines Nutzers wurde zunächst zugunsten der Entwicklung der notwendigen Basisfunktionen zum Offline-Zugriff auf beliebige Datentypen im Demonstrator verzichtet, die am Beispiel der Kontakte realisiert wurden. Die Nachrüstung der Terminunterstützung erfolgte im Rahmen einer Überprüfung der gesamten MOHITO Prozessschritte zum Projektende (Erzeugung Datenmodell, Generierung Android Datenhaltungsstack, Integration in den xRM Client, Tests gegen den Open Server) und war erfolgreich. Weder am generierten Code noch an den Templates selbst waren Anpassungen erforderlich.
A2.1.2-10	*n* relevanteste Datensätze offline	Soll	Nicht erfüllt. Auf eine Relevanzbestimmung wurde aus Komplexitätsgründen verzichtet.
A2.1.2-11	Systemgestützte Relevanzermittlung für Datensätze	Soll	Nicht erfüllt. Auf eine Relevanzbestimmung wurde aus Komplexitätsgründen verzichtet.
A2.1.2-12	Relevanzbestimmung konfigurierbar	Kann	Nicht erfüllt. Auf eine Relevanzbestimmung wurde aus Komplexitätsgründen verzichtet.

A2.1.2-13	Prefetching der Verknüpfungen aller Termine der nächsten Woche	Soll	Nicht erfüllt. Der Demonstrator bietet keine Möglichkeit zum Prefetching verknüpfter Datensätze. Generell lassen sich aber Prefetching-Strategien auf Basis der Criteria API des MOHITO Frameworks umsetzen.
	Manuelles Herunterladen		
A2.1.2-14	Herunterladen einzelner Daten	Muss	Erfüllt. Einzelne Datensätze werden beim Öffnen heruntergeladen und dabei automatisch Offline verfügbar gemacht.
A2.1.2-15	Herunterladen gesamte Akten	Soll	Nicht erfüllt. Der Demonstrator erlaubt prinzipbedingt kein Herunterladen von Akten.
A2.1.2-16	Herunterladen von Typen von Daten	Kann	Teilweise erfüllt. Der Datenhaltungs-Stack des Demonstrators erlaubt das Herunterladen beliebiger Datensatztypen. Unterstützt werden Kontakte und Termine (geplant, siehe A 2.1.2-9).
A2.1.2-17	Herunterladen verknüpfter Daten	Muss	Teilweise. Auf ein vollautomatisches Herunterladen verknüpfter Datensätze wurde aus Komplexitätsgründen verzichtet. Die Verweise selbst werden bereits abgeglichen. Die prinzipielle Umsetzbarkeit ist aber durch den Proxy-Mechanismus des MOHITO Frameworks gegeben.
A2.1.2-18	Herunterladen aller Verknüpfungen mit konfigurierbarer Verknüpfungstiefe	Kann	Nicht erfüllt, da m:n Verknüpfungen nicht unterstützt werden.
A2.1.2-19	Beeinflussen der Verdrängungsstrategie manuell heruntergeladener Daten	Soll	Nicht erfüllt, da das MOHITO Framework keine konfigurierbare Verdrängungsstrategie unterstützt.
A2.1.2-22	Nachfrage "Verbindung herstellen" beim Öffnen von nicht-heruntergeladenen Datensätzen.	Soll	Nicht unterstützt.
	Visualisierung		
A2.1.2-20	Direkte Sichtbarkeit welche Daten heruntergeladen wurden	Muss	Nicht mehr relevant, da die für den Demonstrator gewählte Konfiguration des MOHITO Frameworks zur Anwendung hin nicht zwischen heruntergeladenen und lokal erzeugten Datensätzen unterschei-

			det und eine für den Nutzer transparente Synchronisation mit dem Verfügbarwerden einer Online-Verbindung durchführt. Eine Visualisierung wäre durch die im MOHITO Framework geschaffene Möglichkeit zum unabhängigen Zugriff auf client-lokale und serverseitig gespeicherte Datensätze leicht möglich, ist aber insbesondere im Hinblick auf das Kosten/Nutzen Verhältnis (Serverzugriff teuer) im Demonstrator nicht sinnvoll.
A2.1.2-21	Offline alle Server-Daten als Liste	Soll	Erfüllt. Die vom Client abgerufenen Datensatzlisten sind Offline zugreifbar.
A2.1.2-23	Filter nur heruntergeladene Daten	Soll	Nicht mehr relevant (siehe A2.1.2-20).
A2.1.2-28.	Direkte Sichtbarkeit welche Daten synchronisiert sind, nicht synchronisiert oder im Konflikt	Muss	Nicht mehr relevant (siehe A2.1.2-20). Die Konfliktlösung erfolgt automatisch durch das MOHITO Framework nach der im Demonstrator ausgewählten Konfliktlösungsstrategie des Frameworks (wahlweise ‚Server Wins', ‚Client Wins').
A2.1.2-37	Filter nur nicht synchronisierte Daten (Ausgangsbox)	Soll	Nicht mehr relevant (siehe A2.1.2-20).
	Synchronisation		
A2.1.2-24	Automatisches Sync. im online Modus	Muss	Erfüllt. Die für den Demonstrator gewählte Konfiguration des MOHITO Frameworks synchronisiert die Datensätze automatisch beim Verfügbarwerden einer Online-Verbindung.
A2.1.2-25	Manuelles Sync. einzelner Daten	Muss	Nicht mehr relevant (siehe A2.1.2-20).
A2.1.2-26	Manuelles Sync. aller Daten	Muss	Nicht mehr relevant (siehe A2.1.2-20).
A2.1.2-27	Nachfrage "Verbindung herstellen?" beim manuellen Sync. im offline-Modus	Soll	Nicht mehr relevant (siehe A2.1.2-20).
A2.1.2-33	Automatisches Sync. aller Daten	Muss	Erfüllt (siehe A2.1.2-24).
A2.1.2-34	Nachfrage ob synchronisiert werden soll und Benachrichtigung welche (beim automati-	Muss	Nicht erfüllt, da für den Demonstrator nicht relevant. Eine Nachfrage kann beispielsweise für ein

	schen Sync.)		Produktivsystem leicht integriert werden.
A2.1.2-35	Priorität der zu synchronisierenden Daten	Soll	Im Demonstrator nicht erfüllt, jedoch auf Grundlage der im MOHITO Framework existierenden Mechanismen umsetzbar.
A2.1.2-36	Priorität konfigurierbar	Soll	
A2.1.2-38	Reihenfolge der zu synchronisierenden Daten explizit angeben	Soll	Nicht mehr relevant, da für den Benutzer zu komplex. Auf Grundlage der im MOHITO Framework existierenden Mechanismen jedoch umsetzbar.
A2.1.2-39	Abwählen einzelner zu synchronisierender Daten	Soll	Nicht mehr relevant, da für den Benutzer zu komplex. Auf Grundlage der im MOHITO Framework existierenden Mechanismen jedoch umsetzbar.
A2.1.2-29	Konfliktbehebung auf Smartphone	Kann	Erfüllt. Die Konfliktbehebung erfolgt auf dem Client nach der gewählten Strategie.
A2.1.2-30	Konfliktbehebung auf PC	Muss	Nicht mehr relevant, die Konfliktbehebung erfolgt auf dem Client nach der gewählten Strategie.
A2.1.2-31	Anzeige der Server-Version der konfliktiven Daten	Kann	Nicht mehr relevant, da eine vorgewählte Konfliktlösungsstrategie verwendet wird.
A2.1.2-32	Vorschlag zum automatischen Merge	Kann	
	Sicherheit		
A2.1.2-40	Lesen der offline Daten nicht möglich	Soll	Nicht umgesetzt. Als Verschlüsselungsmechanismus für die eingesetzte SQLite DB wurde SQLCipher[1] in Betracht gezogen, jedoch ist die sichere Ablage des Schlüssels auf mobilen Clients ohne Wechseldatenträger oder Hardware Schlüssel problematisch.
A2.1.2-41	Beschränkung der Option zur offline-Verfügbarkeit (nicht alle Unternehmensdaten)	Kann	Nicht umgesetzt. Auf Grundlage der im MOHITO Framework existierenden Mechanismen jedoch umsetzbar. Durchsetzung sollte in jedem Fall durch den Server sichergestellt werden.
A2.1.2-42	Verschlüsselte Datenübertragung	Muss	Erfüllt. Eine Kommunikation zwischen Client und Open Server über HTTPS ist möglich.

[1] https://guardianproject.info/code/sqlcipher/

A2.1.2-43	Sicherheit des Servers	Muss	Erfüllt. Authentifizierung des Benutzers vor Abruf von Serverdaten notwendig. Verschlüsselte Datenübertragung. Rechtesystem auf Datensatzebene (Discretionary Access Control).

2.2 Demonstrator Bibliotheksanwendung

Die nachfolgend dargestellten Anforderungen wurden, wie auch die Referenzimplementierung, basierend auf dem „Hier Bin Ich“ Dienst (HereIAm App) entwickelt. Diese Anwendung erwies sich in mehrerlei Hinsicht für eine Verwendung im Rahmen eines Demonstrators ungeeignet. Die grundlegende Idee für den Demonstrator war bei der Konzeption von Anfang an, diesen im Rahmen des „Open Source“ Software-Release einsetzen und als eine Art Einführung in das MOHITO Framework, also als eine Art Tutorial, verwenden zu können. Dazu musst der Demonstrator einfach genug gehalten sein, um als leicht verständliches Beispiel dienen zu können. Die „Hier Bin Ich“ Referenzanwendung erwies sich für Einsteiger als zu komplex und mit vielen Details versehen, deren Komplexität in der Anwendung nicht jedoch im MOHITO Framework lag. Der Demonstrator durfte des Weiteren ausschließlich auf quelloffener Software basieren, ein Punkt, der bei der Referenzanwendung durch die Verwendung von Bibliotheken für die Darstellung in der Implementierung verletzt war. Der Mehraufwand für eine Umstellung auf quelloffene Alternativen hätte hohen Aufwand erfordert oder die Qualität der Darstellung reduziert. Zusätzlich wurde im Lauf der Entwicklung identifiziert, dass insbesondere Einsteiger von einer Einführung in EMF und die Erstellung von Modellen profitieren. Eine Anlehnung an bestehende Beispiele aus der EMF Welt anlehnen bietet zusätzlich einen Wiedererkennungswert für erfahrene EMF Entwickler.

Aus all diesen Überlegungen heraus, wurde der B2M Demonstrator an das bestehende Tutorial „Generating an EMF Model“ angelehnt (Eclipse Foundation, 2007). Dieses Tutorial beschreibt eine einfache Bücherei Anwendung, in welcher Bücher verschiedener Autoren in unterschiedlichen Kategorien vorhanden sind, die dann entsprechend ausgeliehen, also aus dem System entfernt, und später wieder zurückgebracht, also im System wieder verfügbar gemacht werden können. Des Weiteren bietet der Demonstrator die Möglichkeit die Eigenschaften bestehender Bücher, wie beispielsweise die Seitenzahl, oder die Kategorie, welcher sie zugeordnet sind verändern zu können und diese geänderten Informationen zu einem späteren Zeitpunkt mit dem Server zu synchronisieren. Somit nutzt und demonstriert die Anwendung alle Kerneigenschaften des MOHITO Frameworks.

Dennoch sollte auch die dabei entstehende „Bücherei“ Anwendung - sofern möglich und im Kontext der Anwendungsdomäne passend´- alle ursprünglich an die Referenzimplementie-

rung gestellten Anforderungen erfüllen. Dies ist auch für alle zum Kern des Mohito Projekts gehörenden Anforderungen geschehen, wie in der nachfolgenden Tabelle dargestellt.

	Anforderung	**Gewich-tung**	**Anmerkung zur Umsetzung in der „Bibliotheks Anwendung“**
A2.2-1	Offline Fähigkeit	Muss	Vollständig erfüllt. Bücher werden mit dem Server synchronisiert und sind jederzeit offline verfügbar.
A2.2-2	Dienstprofile und Nutzerdaten müssen immer verfügbar sein	Muss	Macht im Kontext des Demonstrators keinen Sinn, da hier keine Nutzerdaten vorhanden sind.
A2.2-3	Nicht mehr aktuelle Daten müssen entsprechend visualisiert werden	Soll	Bei dem Versuch die Komplexität des Demonstrators einzuschränken ist diese Anforderung implizit dadurch erfüllt, das nicht mehr aktuelle Daten, bei der nächsten Synchronisation auf den neusten Stand gebracht und entsprechend visualisiert werden.
A2.2-4	Anbindung an mehrere beteiligte Dienste	Muss	Macht im Kontext des Demonstrators keinen Sinn, da hier keine unterschiedlichen Dienste angeboten werden.
A2.2-5	Bandbreitenlimitierung mobiler Geräte muss berücksichtigt werden (Caching)	Soll	Vollständig erfüllt. Eine Synchronisation mit dem Server erfolgt nur explizit von dem Benutzer initiiert, danach wird der Cache genutzt.
A2.2-6	Mechanismus zur Reduktion des Datenvolumens	Soll	Vollständig erfüllt. Anfragen an der Sever werden nur zu dedizierten Zeitpunkten gestellt, ansonsten wird auf lokal vorhandenen Daten gearbeitet
A2.2-7	Automatische Konfliktbehebung	Muss	Vollständig erfüllt. Bei der Konfliktlösung gewinnt im Konfliktfall entweder der Server oder Client.
A2.2-8	Manuelle Konfliktbehebung auf Smartphone	Kann	Um die Komplexität des Demonstrators gering zu halten wurde auf eine manuelle Konfliktlösung auf dem mobilen Gerät verzichtet.

3 Anforderungsanalyse

In diesem Kapitel werden die technischen Anforderungen an Datenschichtmodellierung und den Datenzugriff dokumentiert. Weiterhin werden Anforderungen an die zu realisierende Werkzeugunterstützung, an Laufzeitmechanismen, sowie nicht-technische Anforderungen beschrieben.

3.1 Datenschichtmodellierung und domänenspezifische Sprache

3.1.1 Modellierung von Entitäten und Beziehungen

MOHITO Datenmodelle erlaube die Modellierung komplexer Zusammenhänge. Attribute, Aufzählungen, Referenzen und Kardinalitäten können bei der Modellierung verwendet werden. Neben den Basisdatentypen werden auch komplexe und zusammengesetzte Datentypen unterstützt. Die Ausdrucksmächtigkeit ist durch die Aufzählungen sogar noch größer als initial gefordert.

Das Modell für Client und Server ist identisch. Eine weitere Validierung daher nicht notwendig.

Die Modellierung der Datenstruktur ist durch die Verwendung von EMF und UML Klassendiagrammen möglichst nah an etablierten Sprachen gehalten. Die Umsetzung damit erfolgreich.

3.1.2 Modellierung von Dateneigenschaften und –Behandlung

Die Varianz über Anwendung und Domäne hinweg wird durch das MOHITO Framework unterstützt, in dem grundlegende Funktionen zum Abgleich zur Verfügung gestellt werden das Intervall jedoch in der Anwendung umgesetzt werden kann. Dies erlaubt insbesondere auf Spezialfälle abhängig von der Konnektivität und Benutzerführung auf mobilen Geräten exakt reagieren zu können.

Die notwendigen Metadaten zur Verwaltung von MOHITO Entities wurden in den Demonstratoren validiert und haben sich als erfolgreich erwiesen.

Die Umsetzung der Sichten zur Berücksichtigung unterschiedlicher Bearbeitungskontexte als Reiter in Eigenschaftsseiten erlaubt eine getrennte Bearbeitung durch Entwickler. Diese können wie gewünscht fokussiert arbeiten.

3.1.3 Anforderungsübersicht

Die folgende Tabelle fasst die zuvor beschriebenen Anforderungen und Anmerkungen nochmals zusammen:

	Anforderung	Gewichtung	Anmerkung zur Umsetzung
A3.1-1	Datenstruktur-Modellierung angelehnt an etablierte Modellierungssprachen	Muss	Durch UML Klassendiagramme und EMF erfüllt.
A3.1-2	Modellierung eines Datenmodells zentralen für alle Zielplattformen	Muss	Durch ein Modell für alle Plattformen erfüllt.
A3.1-3	DSL zur Spezifikation von Dateneigenschaften und –behandlung (ggf. Integration von Metadaten)	Muss	Erfüllt.
A3.1-4	Möglichkeit getrennter Sichten zur Modellierung unterschiedlicher Aspekte	Muss	Erfüllt.

3.2 Toolunterstützung

Das zentrale Werkzeug ist eine Eclipse-basierte IDE, welche die Bearbeitung von Modellen und deren Annotation erlaubt (siehe auch (Konsortium, L7.2: Dokumentation verfeinerte DSL, 2013)). Im Rahmen der Demonstratoren und des Tutorials (Konsortium, L4.2: xRM Demonstrator, 2014) (Konsortium, L4.3: mobiler Demonstrator, 2014) erwies sich die Lösung als erfolgreich.

Die Generatoren für die unterschiedlichen Zielplattformen und Laufzeitartefakte (Konsortium, L 8.2: Finale Version der Generatortemplates und des Modellinterpreters für die unterstützten Plattformen, 2013) wurde ebenfalls im Rahmen der Demonstratoren und des Tutorials (Konsortium, L4.2: xRM Demonstrator, 2014) (Konsortium, L4.3: mobiler Demonstrator, 2014) getestet. Die Umsetzung erwies sich als erfolgreich.

Die Eclipse-Platform wurde weitest möglich genutzt. Alle MOHITO Werkzeuge basieren auf der Plattform.

Die folgende Tabelle fast die Anforderungen an die Werkzeugunterstützung zusammen und stellt Anmerkungen dar.

	Anforderung	Gewichtung	Anmerkung zur Umsetzung
A3.2-1	Werkzeug zur Instanziierung und Bearbeitung von MOHITO Modellen	Muss	Durch Eclipse-basierte IDE zur Modellierung erfüllt.
A3.2-2	Generator zur Abbildung von MOHITO Modellen auf plattformspezifische Laufzeitartefakte	Muss	Erfüllt.
A3.2-3	Nutzung der Eclipse-Plattform wo möglich	Muss	Erfüllt.

3.3 Homogene Datenzugriffs-API

Ein wichtiger Aspekt der entstandenen Mohito API war die Plattformunabhängigkeit, universelle Einsetzbarkeit und vereinheitlichte Programmierschnittstelle über Plattformen hinweg, unter Beachtung der Möglichkeiten zur Berücksichtigung von Plattformspezifika und Plattformeinschränkungen. Dieser Aspekt wurde während des kompletten Entwicklungszyklus des Mohito Frameworks beachtet und das Ergebnis erfüllt alle zu Anfang gestellten Anforderungen, wie in der nachstehenden Tabelle dargestellt.

	Anforderung	**Gewichtung**	**Anmerkung zur Umsetzung**
A3.3-1	Vereinheitlichte Programmierschnittstelle über Plattformen hinweg	Muss	Vollständig erfüllt. Die Schnittstellen, die im Mohito Framework entstanden sind, sind von ihrer Konzeption her plattformunabhängig und somit universell einsetzbar.
A3.3-2	Möglichkeit zur Berücksichtigung von Plattformspezifika und Plattformeinschränkungen	Soll	Vollständig erfüllt. Das Mohito Framework ist erweiterbar, bzw. kann über plattformspezifische Abstraktionsschichten angesprochen werden und damit an Plattformspezifika angepasst werden.
A3.3-3	Datenoperationen mit der Mächtigkeit von CRUD	Muss	Vollständig erfüllt. Das Mohito Framework bildet alle CRUD Operation ab.
A3.3-4	Use Case spezifische Operationen, die über CRUD hinausgehen	Soll	Vollständig erfüllt. Im Zuge spezifischer Operationen ist die Mohito Query Language entstanden, die komplexe Datenanfragen an den Datenhaltungsstack zulässt.
A3.3-5	Berücksichtigung der Schnittstellenqualität	Soll	Vollständig erfüllt. Die Qualität der Mohito API wurde durch immer wiederkehrende Code-Reviews durch das gesamte Konsortium und durch Unit Tests sichergestellt.

3.4 Laufzeitmechanismen

In diesem Abschnitt wird die im Rahmen des MOHITO Projekts erreichte Umsetzung der im Ergebnisdokument 2.1 (Konsortium, D2.1 Anwendungsszenarios und Anforderungsanalyse, 2013) erfassten und wohldokumentierten Anforderungen an die vom MOHITO Framework bereitzustellenden Laufzeitmechanismen tabellarisch dokumentiert. Hier steht die umgesetz-

te Funktionalität des Frameworks im Mittelpunkt. Die im Projekt definierten Anwendungsszenarios, die auf den Laufzeitmechanismen aufsetzen und die Praktikabilität der entwickelten Lösung aufzeigen, finden sich in Kapitel 2 Anwendungsszenarios .

	Anforderung	**Gewichtung**	**Anmerkung zur Umsetzung**
A3.4.1-1	Unterstützung für Caching von Datensätzen	Muss	Erfüllt. Vom Server abgerufene Datensätze werden bei entsprechender Konfiguration durch das MOHITO Framework lokal beim Client persistiert. Nachfolgende Zugriffe werden dann aus dem lokalen Speicher des Clients bedient, sofern der Datensatz dort vorhanden ist. Andernfalls wird er vom Server abgerufen.
A3.4.1-2	Unterstützung für Prefetching von Datensätzen	Muss	Erfüllt. Vollständiges und selektives Prefetching ist durch den Abruf von Listen ohne bzw. mit Filterbedingungen möglich, die bei entsprechender Konfiguration des MOHITO Frameworks lokal beim Client persistiert und im Offline-Fall lokal abgerufen werden. Dies schließt Einzeldatensätze (Listen der Länge eins) ein.
A3.4.1-3	Unterstützung für Synchronisation von Datensätzen zwischen Client und Server	Muss	Erfüllt.
A3.4.1-4	Offline-Funktionalität auf Client verfügbar	Muss	Erfüllt.
A3.4.1-5	REST Unterstützung	Muss	Erfüllt.
A3.4.1-6	Datenaustausch über strukturierte Datentypen in JSON oder XML Notation	Muss	Erfüllt (JSON).
A3.4.1-7	Verfügbarkeit für eine mobile Plattform (Android oder iOS)	Muss	Erfüllt (Android).
A3.4.1-8	Verfügbarkeit für zwei mobile Plattformen (Android und iOS)	Soll	Nicht erfüllt.
A3.4.1-9	Browserunterstützung	Kann	Nicht erfüllt.
A3.4.2-1	Unterstützung für CRUD-Zugriffe exisitiert	Muss	Erfüllt.
A3.4.2-2	Serverinterne Modifikation übergebener Datensätze wird berücksichtigt	Muss	Erfüllt.
A3.4.2-3	Integration mit dem Berechtigungssystem des Applikationsservers	Muss	Erfüllt. Datensatzzugriffe erfolgen serverseitig mit den Berechtigungen des Client-Benutzers.
A3.4.3-1	Datenmodelländerungen über den Lebenszyklus einer An-	Muss	Erfüllt für statische Änderungen des Datenmodells.

	wendung hinweg unterstützen		
A3.4.3-2	Hinzufügen neuer Attribute zu existierendem Datentyp zur Laufzeit möglich	Muss	Nicht erfüllt. Inkompatibel zum generativen MOHITO-Ansatz.
A3.4.4-1	Versionierte Datensätzen werden unterstützt	Soll	Teilweise erfüllt (serverseitig für CAS Open). Im Demonstrator kommen Zeitstempel zur Erkennung unterschiedlicher Datensatzversionen zum Einsatz.
A3.4.4-2	Erweiterungspunkt für Datensatzversionierung durch MOHITO Runtime existiert	Kann	Aus Komplexitätsgründen nicht umgesetzt.
A3.4.5-1	Clientseitiger Datensatzcache	Muss	Erfüllt (siehe A3.4.1-1).
A3.4.5-2	Cache bedient Lese-/Schreibanfragen falls Server nicht erreichbar	Muss	Erfüllt.
A3.4.5-3	Erzeugen und Löschen von Datensätzen im Cache möglich	Soll	Erfüllt im MOHITO Framework.
A3.4.5-4	Automatisches Verdrängen von Cacheeinträgen (konfigurierbar)	Muss	Teilweise. Es existiert keine Möglichkeit zur automatischen Verdrängung einzelner Cacheeinträgen. Alle zwischengespeicherten Datensätze lassen sich im Blick entfernen.
A3.4.5-5	Datensätze lassen sich durch Markierung vor automatischer Verdrängung schützen (Verdrängung erfolgt durch expliziten Aufruf)	Muss	
A3.4.5-6	Datensätze lassen sich durch Markierung vor clientseitigem Caching schützen	Muss	Nicht umgesetzt.
A3.4.5-7	Cache-Inhalte werden verschlüsselt	Soll	Nicht umgesetzt (siehe A2.1.2-40).
A3.4.5-8	Cache liefert Cache-Statistik und Metadaten zu Cacheeinträgen über eine Schnittstelle aus	Muss	Nicht umgesetzt.
A3.4.6-1	Synchronisation zum Datenabgleich zw. Cache und Server möglich	Muss	Erfüllt.
A3.4.6-2	Synchronisationskonflikte werden erkannt	Muss	Erfüllt.
A3.4.6-3	Sync.-konfliktbehandlung ist automatisch und manuell möglich	Muss	Erfüllt. Automatische Konfliktbehandlung im Client wurde realisiert. Die Notwendigkeit einer manuellen Konfliktbehandlung wird vom MOHITO Framework durch Auswertung der Ergebnisse des Synchronisationslaufs (Rückgabe einer nichtleeren Liste von in Konflikt stehenden Datensätzen) fest-

			stellen.
A3.4.6-4	Manuelle Konfliktbehebung auf mobilen Clients für einzelne Datensätze	Muss	Nicht relevant da keine Aufgabe des MOHITO Frameworks (vgl. A3.4.6-2).
A3.4.6-5	Manuelle Konfliktbehandlung für mehrere Datensätze über Desktop/Notebook möglich	Soll	Nicht erfüllt (keine serverseitige Konfliktbehandlung).
A3.4.6-6	Konnektivität des Clients wird erkannt	Muss	Erfüllt. Das MOHITO Framework lässt sich auf Basis der Konnektivität des Client konfigurieren und richtet sein Verhalten daran aus. Die Ermittlung der Konnektivität selbst ist keine Aufgabe des MOHITO Frameworks.
A3.4.6-7	QoS der Internetverbindung wird bei Synchronisation berücksichtigt	Kann	Nicht erfüllt.
A3.4.6-8	Start des Sync.-vorgangs bei bestehender Verbindung wahlweise automatisch oder manuell	Muss	Erfüllt. Das MOHITO Framework erlaubt die Umsetzung beider Strategien, im manuellen Fall nur innerhalb einer auf ihm aufbauenden Anwendung.
A3.4.6-9	Manuelle Synchronisation einzelner Datensätze bei manuellem Start des Sync.-vorgangs möglich	Muss	Erfüllt. Das MOHITO Framework erlaubt die Umsetzung dieser Strategie innerhalb einer auf ihm aufbauenden Anwendung.
A3.4.6-10	Synchronisationsreihenfolge von Datensätzen über Prioritäten steuerbar	Soll	Nicht erfüllt.
A3.4.6-11	Verbindungsabbruch bei laufender Synchronisation wird geeignet behandelt	Muss	Erfüllt. Eine anschließende Ausnahebehandlung durch die auf dem Framework aufsetzende Anwendung ist möglich.
A3.4.7-1	Datensatzauswahl für Prefetching über austauschbare Strategie möglich	Muss	Teilweise erfüllt. Prefetching-Strategien lassen sich auf Basis der Criteria API des MOHITO Frameworks leicht umsetzen, werden aber vom Demonstrator nicht unterstützt.
A3.4.7-2	Prefetching-Strategie mit manueller Datensatzauswahl existiert	Muss	
A3.4.7-3	Prefetching-Strategie mit automatischer Datensatzauswahl existiert	Soll	

3.5 Anbindung von Bestandstechnik

Ein wesentlicher Bestandteil des Mohito Projektes beschäftigte sich mit der Anbindung von Bestandssoftware an den Mohito-Stack, als auch der Migration bestehender Softwarelösungen hin zum Mohito Framework. Beiden Aspekten wurden während der Entwicklung Rechnung getragen. So wurde sowohl die Anbindung des MML der B2M als auch der CAS Open

Plattform im Rahm des Projektes realisiert. Des Weiteren wurden während der Entwicklung der Demonstratoren die wichtigsten Aspekte der Migration von Bestandssoftware beleuchtet und in den Dokumenten L5.1 (Klatt, Gailus, Hübsch, & Schwichtenberg, 2013) und L5.2 (Schwichtenberg, Hübsch, Gailus, & Groenda, 2014) detailliert erörtert. Ein ebenso wichtiger Aspekt war die Integration der bestehenden Werkzeugkette, die im verwandten Modagile Mobile Projekt entstanden war und die Anbindung der Modagile Mobile Datenhaltung an Mohito. Abschließend kann festgestellt werden, dass all diese Anforderungen berücksichtigt und entsprechend umgesetzt wurden. Im Folgenden ist eine tabellarische Übersicht der Anforderungen und ihrer Umsetzung zu finden.

	Anforderung	**Gewichtung**	**Anmerkung zur Umsetzung**
A3.5-1	Anbindung an den MML	Soll	Die clientseitige Referenzanwendung wurde vollständig auf dem MML aufgesetzt und der MML wurde entsprechend der analysierten Anforderungen weiterentwickelt, um als Serverkomponente und Referenzimplementierung im Rahmen des Mohito Projekts dienen zu können.
A3.5-2	Anbindung an die Modagile Mobile Datenhaltung und Integration in die Werkzeugkette	Soll	Bei der Entwicklung der Modellierungs-Werkzeugkette wurde vollständig auf den bestehenden Modagile Mobile Werkzeugen und deren Quellcode aufgesetzt und dieser entsprechend weiterentwickelt. Somit ist eine perfekte Integration in das bestehende Modagile Mobile Projekt gewährleistet und Mohito kann als Datenhaltungsschicht von Modagile Mobile genutzt werden.
A3.5-3	Anbindung an CAS Open	Soll	Die clientseitige Referenzimplementierung des xRM Demonstrators nutzt die REST Schnittstelle des CAS Open Servers und ist gegen das auf Basis von CAS Open realisierte Produktivsystem CAS PIA lauffähig. Die Integration der Serverseite wurde weiterhin durch eine Modell-zu-Code Transformation umgesetzt, die aus MOHITO Datenmodellen vom CAS Open Server verwendete deklarative Datentypbeschreibungen erzeugt.

3.6 Nicht-technische Anforderungen

Die entstandene Lösung und das Tutorial sind ohne Kauf proprietärer Produkte durch Dritte verwendbar. Die Lösung ist als Open Source verfügbar und ihre Weiternutzung möglich.

	Anforderung	**Gewich-tung**	**Anmerkung zur Umsetzung**
A3.6-1	Bereitstellung eines Referenz-stacks als Open Source	Soll	Erfüllt.

4 Literaturverzeichnis

Eclipse Foundation. (2007 йил 18-06). *Eclipse documentation* . Retrieved 2014 йил 12-06 from http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.emf.doc%2Ftutorials%2Fclibmod%2Fclibmod.html&cp=17_1_0

Klatt, B., Gailus, E., Hübsch, G., & Schwichtenberg, A. (2013). *L 5.1: Migrationsstrategie für Bestandsanwendungen zur MOHITO Referenzarchitektur.*

Konsortium, M. (2013). *D2.1 Anwendungsszenarios und Anforderungsanalyse.*

Konsortium, M. (2013). *L 8.2: Finale Version der Generatortemplates und des Modellinterpreters für die unterstützten Plattformen.*

Konsortium, M. (2013). *L7.2: Dokumentation verfeinerte DSL.*

Konsortium, M. (2014). *L4.2: xRM Demonstrator.*

Konsortium, M. (2014). *L4.3: mobiler Demonstrator.*

Mohito, K. (2013). Dokumentation der Anwendungsszenarien, der Anforderungen und des Stands der Technik.

Schwichtenberg, A., Hübsch, G., Gailus, E., & Groenda, H. (2014). *L5.2 Migrationsstrategie für die sanfte Überführung von Bestandssoftware zu modellbasierten Multi-Plattformanwendungen mittels MOHITO.*