Eine generische Softwarebibliothek zur Web-Umsetzung des Datenschutzkonzepts A des TMF e.V.

Institut für Medizinische Informatik
und Biomathematik
Dipl.-Inf. Martin Lablans
Dipl.-Inf. René Brüntrup
Prof. Dr. med. Frank Ückert
Domagkstraße 11
48149 Münster

Email: lablans@imfl.de
Fon: +49 (0) 251 – 83-58210
Fax: +49 (0) 251 – 83-58219

1. Zweck und Funktionalität

Zentrale Forderung des Datenschutzkonzepts A der Technologie- und Methodenplattform für die vernetzte medizinische Forschung e.V. (TMF) ist die organisatorische und physikalische Trennung der gespeicherten Datenklassen, etwa der identifizierenden (IDAT) und medizinischen (MDAT) Kontextdaten. Deren Zusammenführung soll dabei allein auf dem Rechner des berechtigten Benutzers geschehen, was in heutigen Webbrowsern mit Bordmitteln aufgrund technischer Beschränkungen bzw. Sicherheitsmerkmalen wie etwa der Same-Origin-Policy nicht ohne Weiteres möglich ist.

Die vorliegende Softwarebibliothek unterstützt den Entwickler bei der praktischen Umsetzung der Datentrennung und Datenzusammenführung mithilfe der folgenden Funktionalitäten:

- Handhabung von temporären Identifikatoren (TempIDs)
    - Erzeugung
        - zufallsgenerierte TempIDs für übergebene IDs (zum Beispiel Patienten-IDs)
        - transparente Wiederverwendung bereits erzeugter TempIDs
        - Verwaltung und Ablauf der Gültigkeit
    - Registrierung beim anderen Server
    - clientseitige Auflösung
        - Verwendung allein von Bordmitteln gängiger Browser ausreichend (JavaScript)
        - Interpretation von HTML-Attributen erlaubt Verwendung bestehender Template-Engines bei minimalem Scripting-Aufwand für Entwickler
- Authentifizierung
- Kommunikation zwischen den Servern durch gemeinsame Sitzungen (vgl. 3.1)
    - Transfer beliebiger Nutzdaten zwischen den Servern ohne Einbeziehung oder Kenntnis des Webbrowsers

Um den Einstieg in die Entwicklung einer Webapplikation mit Datentrennung zu erleichtern, enthält die Bibliothek außerdem drei einfache Beispielimplementierungen, die verschiedene Ansätze zeigen.

Was diese Bibliothek nicht leistet:

- Die Nutzung der Bibliothek entbindet nicht von Lektüre und Verständnis des zugrundeliegenden Datenschutzkonzepts[1].
- Um der Generik willen ist diese Bibliothek kein vollständiges Applikations-Framework, sondern unterstützt allein bei üblicherweise durch die Datentrennung verursachten Problemstellungen. Insbesondere sind nach wie vor Programmierkenntnisse nötig.

[1] vgl. Reng CM, Debold P, Specker C, Pommerening K. *Generische Lösungen zum Datenschutz für die Forschungsnetze in der Medizin*. Medizinisch Wissenschaftliche Verlagsgesellschaft 2006.

2. Entstehung und Veröffentlichung

Eine erste Web-Umsetzung des TMF-Datenschutzkonzepts A erfolgte im Rahmen des Bildserverprojekts „MDPE“ für den HIT-Studienverbund, gefördert durch die Deutsche Kinderkrebsstiftung. Die hier vorliegende Bibliothek beruht auf einer Weiterentwicklung im Rahmen des Projekts „KernPäP“ zur Dokumentation in der ambulanten pädiatrischen Palliativmedizin, gefördert durch die Alfried-Krupp-von-Bohlen-und-Halbach-Stiftung. Die Veröffentlichung erfolgt in Zusammenarbeit mit der TMF, in deren Arbeitsgruppe Datenschutz auch das zugrundeliegende Datenschutzkonzept erarbeitet wurde.

Hinter dieser Bibliothek steht ein Gedanke der freien, gemeinschaftlichen Fortentwicklung. Deshalb erfolgt die Veröffentlichung unter der MIT-Lizenz (für Wortlaut vgl. 6), die u.a. kostenlose Verwendbarkeit auch in kommerziellen Projekten, Nachprüfbarkeit des Programmcodes und Erweiterbarkeit sowie Modifizierbarkeit der Bibliothek durch die Nutzergemeinde gewährleistet. In diesem Sinne wird darum gebeten – auch wenn die Lizenz dazu nicht verpflichtet – freiwillig:

- die Bibliothek als hier vorliegendes Gesamtpaket nicht zum Download bereitzustellen, sondern auf die von der TMF und den Urhebern bereitgestellten Angebote hinzuweisen,
- den Autoren Rückmeldung über die Nutzung zu geben, was unabhängig vom konkreten Projekt eine wertvolle Evaluation darstellt und
- etwaige entwickelte Korrekturen, Verbesserungen und Erweiterungen in die Bibliothek zurückfließen zu lassen, etwa durch Zusendung von Patches an die ursprünglichen Autoren.

3. Grundlegende Funktionsweise

Um einen Überblick über die Funktionalität der Bibliothek zu erhalten, wird in den folgenden beiden Kapital ein einfaches Beispiel verwendet: Es soll eine Applikation entwickelt werden, die es einem Arzt erlaubt sich anzumelden, auf seine Patienten zuzugreifen und zu jedem davon einen medizinischen Datensatz zu speichern. Die Applikation soll gemäß TMF-Datenschutzkonzept A zwei Server verwenden: Der IDAT-Server speichert die Anmeldedaten sowie die identifizierenden Patientendaten und auf dem MDAT-Server werden die medizinischen Informationen abgelegt. Eine Patienten-ID, kurz PID, wird verwendet um diese Daten zu verknüpfen, also jedem Patienten auf IDAT-Seite seine Patientendaten auf MDAT-Seite zuzuordnen. Außerdem soll der Arzt die Möglichkeit haben, sich eine Liste der Patienten mit den neuesten medizinischen Daten anzeigen zu lassen.

Die folgenden beiden Abschnitte erläutern grundlegende Implementierungskonzepte und Abweichungen von der TMF-Beispielimplementierung. Die Abweichungen werden anhand des obigen Beispiels motiviert. Sämtliche Abweichungen wurden von den Autoren innerhalb der TMF-Arbeitsgruppe Datenschutz ausführlich besprochen und befürwortet[2].

3.1 Gemeinsame Sitzung

Die vorliegende Bibliothek stellt eine gemeinsame Sitzung zwischen beiden Servern bereit, in der beliebige Schlüssel-Wert-Paare gehalten werden können. Die Datenpaare werden direkt zwischen den Servern, also ohne Einbeziehung des verwendeten Webbrowsers, übertragen. Diese Synchronisation erfolgt automatisch beim Spei-

[2] vgl. Protokoll der Sitzung der TMF-Arbeitsgruppe Datenschutz am 8. April 2010.

chern der Sitzungsdaten, sodass beide Rechner immer auf den gleichen, aktuellen Datensatz zugreifen können. Diese Implementierung bietet mehrere Vorteile, die im Folgenden beschrieben werden.

Im TMF-Konzept ist die Übertragung einer Berechtigungsliste („Information über Zugangsrestriktionen“) zusätzlich zu jeder erzeugten TempID vorgesehen. Häufig sind die Berechtigungen aber für viele oder alle Patienten identisch, da sie sich aus der Rolle des Benutzers ergeben. Vor allem aus Leistungsgründen ist es daher ratsam eine mehrfache Übertragung derselben Berechtigungsliste zu vermeiden. Das lässt sich durch das Eintragen der Berechtigungen in die gemeinsame Sitzung einfach erreichen.

In den gemeinsamen Sitzungen können durch Eintragen entsprechender Schlüssel-Wert-Paare auch die temporären und zugehörigen echten IDs übertragen werden. Diese haben dann keine separaten Gültigkeitszeiten, sondern behalten ihre Gültigkeit für die gesamte Sitzung. Gegenüber der Fassung des Konzepts aus dem Jahr 2006 ergeben sich daraus wiederum zwei entscheidende Vorteile. Zum einen werden temporäre IDs, soweit einmal erstellt, ohne erneute Übertragung wiederverwendet. Zum anderen wird eine ganze Klasse von Anwendungsfehlern ausgeschlossen, die sich dadurch ergeben können, dass ein Teil der temporären IDs nicht länger gültig ist. Je nach Projekt kann für den Datenschutz allerdings auch relevant sein, dass TempIDs durch regelmäßiges Aufrechterhalten der Sitzung prinzipiell beliebig lange haltbar sind. Es obliegt dem Entwickler, Sitzungen gegebenenfalls früher zu schließen.

Die Beispielapplikation enthält ein typisches Problem aus der Praxis, bei der eine allgemeine Übertragung der Berechtigungsliste notwendig ist: „Zu welchen Patienten liegen die neuesten medizinischen Daten vor?“. Streng nach Implementierungsempfehlung müsste für jeden Patienten, der je vom Arzt behandelt wurde, vom IDAT-Server eine TempID erzeugt werden. Dieser Satz an TempIDs würde an den Arbeitsplatzrechner übertragen, der unter Angabe des vollständigen TempID-Satzes vom MDAT-Server sein angefordertes Ergebnis erhielte. Mit einer gemeinsamen Sitzung kann der Arbeitsplatzrechner die Anfrage direkt an den MDAT-Server stellen, an dem er über die gemeinsame Sitzung ebenfalls angemeldet ist. Die Informationen über die vom Arzt zugreifbaren Patienten wurden im Rahmen der Rechteliste über die gemeinsame Sitzung zur temporären Haltung an den MDAT-Server übertragen, so dass der Server direkt mit den TempIDs der zuletzt behandelten Patienten antworten kann.

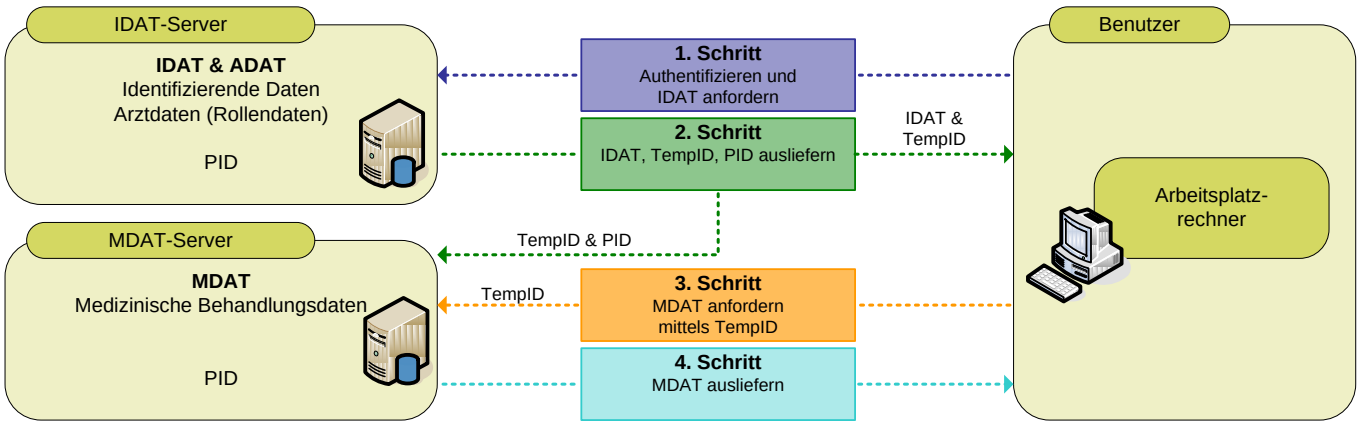

Abbildung 1: Anlegen und Nutzen einer TempID (nach Beispielimplementierung im TMF-Konzept, Modell A).

Als weitere Konsequenz aus der Verwendung einer gemeinsamen Sitzung ergibt sich die Möglichkeit, TempIDs nicht nur auf dem IDAT-, sondern auch auf dem MDAT-Server zu erzeugen (vgl. nächster Abschnitt).

3.2 Rückpfeil

Ursprünglich war die Erstellung der TempIDs allein auf dem IDAT-Server vorgesehen, der diese dann dem MDAT-Server übermittelt (vgl. Abbildung 1). In bestimmten Fällen kann es jedoch notwendig sein, dass auch der MDAT-Server diese Identifikatoren erzeugt und seinerseits an den IDAT-Server sendet. Das erlaubt eine „Rückauflösung“ identifizierender Daten aus MDAT-Antworten.

Betrachtet man erneut obige Fragestellung „Zu welchen Patienten liegen die neuesten medizinischen Daten vor?“, so erhält man als Antwort vom MDAT-Server Personeninformationen, welche nicht mit tatsächlichen Namen versehen sein können, da sie als identifizierendes Merkmal dort nicht vorliegen. Streng nach Konzept darf der MDAT-Server jedoch auch keine TempID anlegen; das darf nur der IDAT-Server. Dieser müsste also vorab TempIDs für sämtliche infrage kommenden Personen erstellen und an MDAT-Server und Browser übermitteln – an letzteren sogar mit allen potentiell benötigten identifizierenden Daten. Das ist datenschutztechnisch diskutabel und zudem aufgrund der Datenmengen unpraktikabel.

In der vorliegenden Softwarebibliothek ist es auch dem MDAT-Server erlaubt TempIDs zu erzeugen. Unter deren Angabe kann der Arbeitsplatzrechner notwendige identifizierende Daten vom IDAT-Server abrufen. So besteht im obigen Beispiel die Antwort vom MDAT-Server aus einer Liste von TempIDs, die sich dann über den „Rückpfeil“ auflösen lassen (vgl. Abbildung 2). Dabei wird die Berechtigung weiterhin durch den IDAT-Server geprüft, dem ja die Rechtetabellen vorliegen. Die TempID hat im „Rückpfeil“ also nur identifizierenden Charakter.

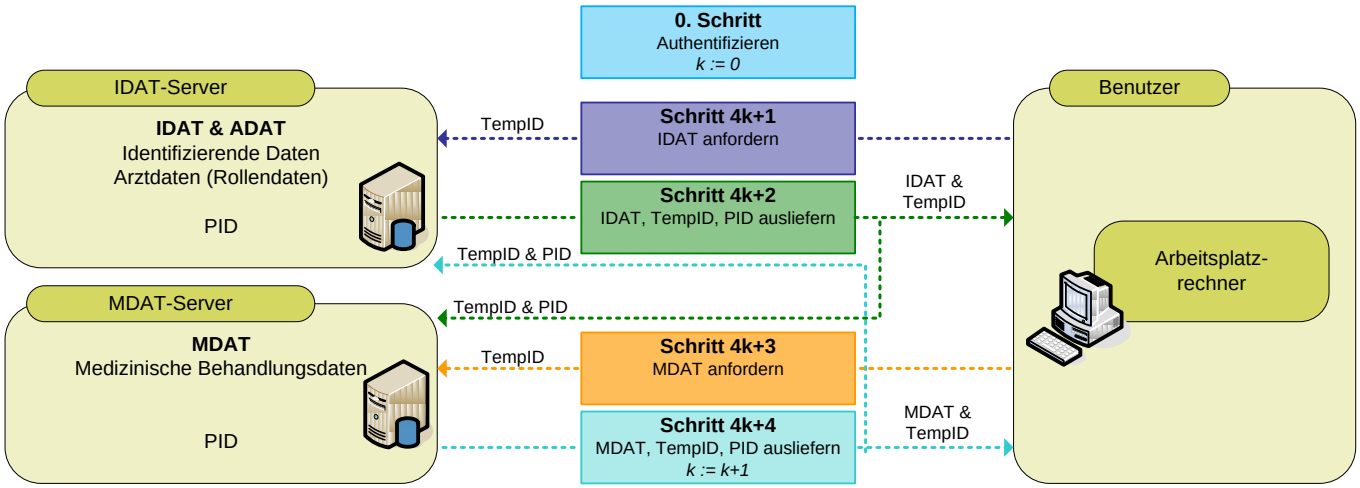

Abbildung 2: Die Erstellung von TempIDs durch den MDAT-Server erlaubt eine schleifenartige Abfrage beider Server. *k* zählt den Schleifendurchlauf.

4. Verwendung

Im Folgenden wird die Verwendung der Bibliothek anhand einer schrittweisen Implementierung der Beispielapplikation unter zur Nutzung der wichtigsten Funktionen demonstriert. Informationen zu den weiteren Funktionen der Bibliothek und den Demoapplikationen befinden sich in der mitgelieferten Datei README.html.

Anmeldung

Zunächst muss der Arzt authentifiziert und eine gemeinsame Sitzung aufgebaut werden. Der Browser sendet die Anmeldedaten des Arztes an den IDAT-Server, wo die Authentifizierung stattfindet. Bei korrekten Anmeldedaten wird dort eine Sitzung mit einer eindeutigen ID angelegt und die Rechteliste darin gespeichert.

```
$sessionId = createUniqueId();
setcookie("idat_session_id", $sessionId);
$remoteSession = remoteSessionFactory->create($sessionId);
$remoteSession->set("userRights", "darf_alles");
$remoteSession->save();
```

Quellcode 1: Anmeldung (IDAT). Nach Authentifizierung wird auf IDAT-Seite eine gemeinsame Session initiiert.

Hierzu wird eine neue Session erstellt (Zeilen 1-3), in der beliebige Schlüssel-Wert-Paare gespeichert werden können. Diese Funktion wird in Zeile 4 genutzt, um die Nutzerrechte unter dem Schlüssel „userRights" in der Session zu speichern. Abschließend wird die Session gespeichert (Zeile 5). „Speichern" bewirkt hier nicht nur eine lokale Speicherung, sondern automatisch auch einen Abgleich der Sitzung mit dem MDAT-Server. Der Anwender ist damit am IDAT-Server angemeldet und dem MDAT-Server ist die Sitzung bekannt. Um die Anmeldung auch beim MDAT-Server zu vervollständigen, genügt es den Benutzer dort dieser Session zuzuordnen. Dazu wird dem Browser die `$sessionId` übertragen, der diese an den MDAT-Server übermittelt und auch dort ein Sitzungscookie erhält[3]:

```
$remoteSession = remoteSessionFactory->load($sessionId);
if (!$remoteSession || $remoteSession->get("userRights") !== "darf_alles")
 return false;
setCookie("mdat_session_id", $sessionId);
```

Quellcode 2: Anmeldung (MDAT). Der MDAT-Server erkennt die gemeinsame Sitzung wieder und kann sie dem Benutzer zuordnen.

Bei nachfolgenden Anfragen können jetzt sowohl IDAT als auch MDAT die Sitzung anhand des Cookies laden. Beispiel für IDAT:

```
if (!isset($_COOKIE[„idat_session_id"])) return null;
return remoteSessionFactory->load($_COOKIE[„idat_session_id"]);
```

Quellcode 3: Laden der etablierten Sitzung (IDAT; MDAT analog).

Erzeugung und Verwendung von TempIDs

Mithilfe der etablierten Sitzung können nun temporäre IDs (TempIDs) erzeugt und verwendet werden. Der angemeldete Arzt wird zunächst eine Liste von Patienten erhalten, die auf dem IDAT-Server wie folgt erstellt werden kann:

```
$patients = getPatients();
foreach ($patients as $patient) {
 $patient->tempId = $remoteSession->getTempId($patient->id);
 unset($patient->id);
}
$remoteSession->save();
```

Quellcode 4: Erzeugung von TempIDs für PIDs (IDAT).

[3] Das zweite Cookie ist erforderlich, da der MDAT-Server – anders als bei Inhalten der gemeinsamen Sitzung – nicht auf die Cookies des IDAT-Servers zugreifen kann (und andersherum). Um das zu verdeutlichen, werden hier trotz gleichen Inhalts serverspezifische Namen verwendet.

Die Liste `$patients` enthält nun TempIDs anstatt PIDs und kann daher an den Client übertragen werden. Nach der Auswahl eines Patienten können die medizinischen Daten unter Angabe der TempID vom MDAT-Server abgerufen werden. Auf dem Server wird die TempID wieder zur ursprünglichen PID aufgelöst:

```
$id = $remoteSession->getId($tempID);
$patientData = getPatientData($id);
```

Quellcode 5: Rückgewinnung der PID aus einer TempID (MDAT).

Holen und Auflösen identifizierender Daten

Zuletzt wird eine Übersicht über zuletzt behandelte Patienten implementiert. Diese Anfrage setzt eine Erstellung von TempIDs durch den MDAT-Server und Auflösung mithilfe des „Rückpfeils“ zum IDAT-Server voraus (vgl. 3.2). Dazu wird folgende Anfrage an den MDAT-Server gestellt:

```
jQuery.getJSON(mdatUrl + „/api/getLatestPatients?physician=abt…0Lc”, …);
```

Die Antwort auf diese Anfrage enthält lediglich TempIDs und keine identifizierenden Daten, etwa

```
[“1od…7du”, “dns…jks”, “m91…pol”]
```

Um diese Daten anzuzeigen, sollen sie nun gegen die tatsächlichen Patientennamen ausgetauscht werden. Um diese häufig wiederkehrende Aufgabe zu vereinfachen, enthält die Bibliothek den sogenannten TempID-Resolver, der bei Aufruf sämtliche DOM-Elemente nach den beiden Attributen `data-subject` und `data-id` durchsucht, diese via Nachfrage beim anderen Server auflöst und die Elemente entsprechend mit Daten füllt. Aus den oben erhaltenen Daten kann z.B. folgender HTML-Code erzeugt werden:

```
<span data-subject=”patientName” data-id=”1od…7du” />
<span data-subject=”patientName” data-id=”dns…jks” />
<span data-subject=”patientName” data-id=”m91…pol” />
```

Alternativ könnte der MDAT-Server natürlich auch direkt diese Ausgabe liefern. Nach dem Einfügen des Codes in die Seite reicht ein einfacher Aufruf des Resolvers, der die TempIDs sammelt und in einer einzigen Anfrage an den IDAT-Server zu identifizierenden Daten auflöst. Diese Daten werden dann direkt in die Seite eingetragen:

```
tempIdResolver.resolve();
```

5. Weiterentwicklung; Berücksichtigung neuer technologischer Entwicklungen

Mit ständiger Weiterentwicklung von Browsern, Webservern und anderen beteiligten Werkzeugen muss eine Softwarebibliothek Schritt halten können. Die Bibliothek berücksichtigt zum Zeitpunkt ihrer ersten Veröffentlichung jüngste Entwicklungen. So existiert etwa eine „Browserweiche“, die transparent für Entwickler und Nutzer „Cross-Origin Resource Sharing“[4] nutzt, falls der genutzte Browser dies schon zulässt. Aber auch neue Entwicklungen werden durch den modularen und leicht anpassbaren Ansatz begünstigt.

[4] vgl. Entwurf des W3C-Konsortiums unter http://www.w3.org/TR/2009/WD-cors-20090317/

Weitere Entwicklungen werden durch den modularen und leicht anpassbaren Ansatz begünstigt und sind aufgrund des frei verfügbaren Quellcodes sowie der gewählten Lizenz auch durch andere Entwickler als die ursprünglichen Urheber möglich (vgl. 2 und 6).

6. Anhang: MIT-Lizenz

Die folgende Lizenz gilt für den im Rahmen dieser Bibliothek ausgelieferten Quellcode.