

Planung & Entwicklung einer Datenbank- bankanwendung für das Forschungs- datenmanagement

Max Brauer

September 2021

Inhaltsverzeichnis

1	Einleitung	3
2	Anforderungen	3
2.1	Technische Anforderungen	3
2.1.1	System	3
2.1.2	Backups	4
2.2	Wahl der Programmiersprachen	4
2.2.1	Bedienoberfläche der Software	4
3	Sicherheit	4
3.1	Grundsätze der IT-Sicherheit	5
3.1.1	Vertraulichkeit	5
3.1.2	Integrität	6
3.1.3	Verfügbarkeit	7
3.1.4	Authentizität	7
3.1.5	Verbindlichkeit	7
3.1.6	Zurechenbarkeit	7
3.1.7	Resilienz	7
3.2	Datenschutz	7
3.2.1	Schutz vor Fremdzugriff	7
3.2.2	Schutz der Betroffenen	7
3.3	Technische Basis	8
4	Technisches System	8
4.1	Überblick	8
4.1.1	Front-End (UI)	9
4.1.2	Back-End (Server)	9
4.1.3	Datenbank	10
4.1.4	Zusammenhalt	10
4.2	Externe Apis	10
4.2.1	DDI Standard	10
4.3	Anmeldung	10
5	Umsetzung	11
5.1	Stolpersteine	12
5.1.1	WebAuthn	12
6	Tests	12
7	Rollout	12
8	Ausblick	12
9	Abschlussbetrachtung	13
10	Anhang	14
10.1	Entity Relationship Diagramm	14
10.2	Projektsteckbrief	15

1 Einleitung

Diese Arbeit wurde in Zusammenarbeit mit dem Max-Planck-Institut für ethnologische Forschung erstellt. Das Ziel ist, das Forschungsdatenmanagement mit Hilfe aktueller Techniken und Software auf einen Stand zu bringen, der vergleichbar mit anderen Forschungseinrichtungen oder -gebieten ist.

Die Forscher in diesem Institut begeben sich, bedingt durch ihre Tätigkeit, auf weltweite Reisen, um mit einer großen Vielfalt an Personen Interviews zu führen und Daten zu sammeln. Hauptsächlich werden diese Interviews in Papierform (handschriftliche Notizen auf einen Block) oder als einfache Audioaufnahme dokumentiert. Letzteres wird händisch als Textdatei transkribiert und alles zusammen in einer großen Sammlung an Word-Dokumenten abgespeichert und archiviert.

Eine weitere Verarbeitung mit diesen Dokumenten erfolgt dann nur noch über das Programm Microsoft Word und/ oder mit den Ausdrucken. Diese Dokumente erfahren häufig keine richtige Versionierung (Änderungen sind nicht mehr nachvollziehbar) und spätestens beim Teilen entstehen Duplikate die später nur sehr aufwändig händisch vereinigt werden.

Ziel war es, diese Sammlung von Daten in ein einheitliches Verwaltungssystem zu überführen und zu organisieren, um die Arbeit mit diesen zu erleichtern und perspektivisch eine automatische Teilverarbeitung in Zukunft zu ermöglichen.

2 Anforderungen

TODO:

- Inhalt (Was?)
- Technische Sachen
 - System
 - Speicherdauer (Fristen?)
 - Architektur
- Auswahl der Programmiersprachen, Datenbanken, Schnittstellen
- DDI Format

2.1 Technische Anforderungen

2.1.1 System

Die Forscher sollen die Software auf ihren Forschungsreisen nutzen und da ist die Wahrscheinlichkeit sehr groß, dass es dabei zweitweise keinen Internetzugriff gibt. Von daher müssen alle Daten offline verfügbar sein. Dennoch können online Backups erstellt werden, sobald eine Internetverbindung wieder aufgebaut wird.

Außerdem kann man von sehr großen Datenmengen ausgehen. Die Forscher werden auf ihren Reisen eine größere Anzahl an Bildern und Videos aufnehmen. Von daher muss das Zielgerät einen größeren Speicherplatz für die Anwendung bereitstellen. Für die externen Geräte (z.B. Kamera) müssen Anschlüsse vorhanden sein, um Daten übertragen zu können.

Des weiteren muss für den Verschlüsselungsprozess auch die benötigte Rechenleistung zur Verfügung stehen.

Aus diesen Anforderung ergibt sich als Zielsystem ein Laptop, da die meisten günstigeren

Mobiltelefone den Anforderungen nur bedingt nachkommen können und der Sicherheitsaspekt nur umständlich garantiert werden kann.

Das Max-Planck-Institut stellt seit Jahren den Forschern für ihre Reise einen relativ aktuellen Windowslaptop zur Verfügung, weshalb primär für dieses Zielsystem entwickelt wird.

TODO:

- Wie sieht es mit App-Entwicklung aus? Kommen Apps überhaupt in Frage? Bekommt man das DBMS überhaupt auf dem Handy zu laufen? Was wären Alternativen?

2.1.2 Backups

Es ist gefordert, ein Backup von allen Daten erstellen zu können. In diesem Fall soll dies in ein Archivierungssystem namens Keeper geschehen.

TODO:

- Herausforderung?

2.2 Wahl der Programmiersprachen

2.2.1 Bedienoberfläche der Software

Für die Oberfläche wurde die Programmiersprache Elm gewählt, da sie einige Vorteile bringt, die einem die Entwicklung und das Testen erleichtert und man damit dennoch schöne Oberflächen bauen kann. So wird diese Programmiersprache damit beworben, dass sie keine Laufzeitfehler hat und der Compiler eine nützliche Fehlermeldung gibt, damit man ein sauberes Programm erhält.

Elm ist eine funktionale Programmiersprache ähnlich zu Haskell und wird in JavaScript übersetzt, damit dies in einem beliebigen modernen Browser ausgeführt werden kann. Von den Vorteilen, die einem Elm bietet, konnte ich mich schon selbst überzeugen und habe auch gelernt mit den Vor- und Nachteilen, die Elm hat, umgehen zu können.

Als große Vorteile lassen sich aufzählen:

- Keine Laufzeitfehler
- Einfaches Refactoring. Typen und Namen lassen sich beliebig ändern und der Compiler stellt sicher, dass man wieder ein gültiges Programm erhält.
- Typsicher. Ich weiß mit welchen Daten ich zu tun habe und was ich mit diesen anstellen kann.
- Eine große Auswahl an Bibliotheken auf package.elm-lang.org.

Als Nachteile gibt es:

- Nicht jede Funktion des Browsers wird in Elm nativ unterstützt. Dazu ist man gezwungen selber JavaScript-Code zu schreiben und dies mittels "Ports" anzubinden.
- Bestehende JavaScript-Bibliotheken arbeiten architekturbedingt eher schlecht mit Elm zusammen.

3 Sicherheit

Bei der Sicherheit müssen eine Reihe von Gesetzen und Richtlinien befolgt werden. Unter den Gesetzen zählen das Bundesdatenschutzgesetz (BDSG) und die europäische General

Data Protection Regulation (GDPR). Der Einfachheit halber wird sich folgend auf das BDSG bezogen, da es die deutsche Implementierung der GDPR ist.

Zudem gibt es eine Richtlinie der Europäischen Kommission [**eu-regulation-ethics-dataprotection**] welche außerdem auf die Sicherheit und den Schutz der Forschungsdaten eingeht.

TODO:

- Grundsätze der IT-Sicherheit (Unterpunkte hier eingliedern)
 - Authentizität
 - Vertraulichkeit
 - Verfügbarkeit
 - Integrität
- Datenschutz
- Technische Basis ableiten
 - Warum
 - Vorteile
 - Alternativen (Nachteile)

3.1 Grundsätze der IT-Sicherheit

In der IT selbst wurden Schutzziele [**eckert-it-sicherheit2012**] definiert. Diese werden hier einzeln aufgeführt und mit den entsprechenden Gesetzen und Normen verglichen.

3.1.1 Vertraulichkeit

Die Daten selbst dürfen nur von autorisierten Nutzern gelesen und bearbeitet werden. Dies gilt auch für den Zugriff auf gespeicherte Daten oder die Übertragung dieser.

Für eine Autorisierung stehen einem eine große Liste an Möglichkeiten zur Auswahl. Hier ein paar Beispiele:

- Benutzername + Passwort
- Auth-Token (z.B. ID-Karte mit Chip und/oder NFC, USB-Sticks)
- Biometrische Daten wie Gesichtserkennung oder Fingerabdrucksensor
- externe Anbieter mittels LDAP oder OAuth
- physische Liste mit Einmalpasswörtern (z.B. die TAN Liste die früher von Banken genutzt wurde)
- Anmeldecodes per SMS oder Email (meist zur Verifizierung als 2. Faktor genutzt)
- Codes über Apps Externer (z.B. Google Auth, Microsoft Authenticator)

Es wird empfohlen mindestens zwei dieser Möglichkeiten zu verbinden (Zwei-Faktor-Authentisierung [**bsi-2fa**]), um einen möglichst guten Schutz erhalten.

Den Zugriff auf die gespeicherten Daten kann man mit folgenden Möglichkeiten erreichen:

- physisches Gerät mit Daten vor unbefugten Zugriff schützen: z.B. Laptop nicht stehen lassen, Gerät nicht weitergeben
- Datenspeicher vor Zugriff schützen: Dies kann man z.B. mit den Rechten des Betriebssystems erreichen.
- Daten vor Zugriff schützen: z.B. den kompletten Datenspeicher verschlüsseln und nur autorisierten Nutzern ermöglichen diesen Bereich zu entschlüsseln

Die sichere Übertragung der Daten geht vergleichsweise einfacher mit einer verschlüsselten Verbindung, auch wenn es hier ein paar Hürden gibt. So soll auf ein etabliertes System (wie TLS) gesetzt werden ¹. Aber auch hier muss auf darauf achten, dass die verwendeten Verschlüsselungsmethoden noch aktuell sind (z.B. MD5 und SHA-1 gelten mittlerweile als veraltet) und die verwendeten Zertifikate noch gelten.

Zertifikate, solange diese von einer vertrauenswürdigen Stelle signiert sind, sind ein probates Mittel um den Schlüsselaustausch und die Authentizität der Gegenseite zu gewährleisten. Hier empfiehlt es sich unter Umständen sogar das Rootzertifikat in der Anwendung mitzuliefern und nicht auf die installierten des Betriebssystems zu verlassen, da Nutzer jederzeit unwissentlich ein kompromittiertes installieren können.

3.1.2 Integrität

“Integrität bezeichnet die Sicherstellung der Korrektheit (Unversehrtheit) von Daten und der korrekten Funktionsweise von Systemen”
[bsi-grundschutz2021]

Es gibt eine große Vielzahl an Faktoren, welche die Integrität von Daten beeinträchtigen können. Dafür hat das BSI in seinen Grundschutzkompendium von 2021 einige Gefahrenquellen [bsi-grundschutz2021] aufgelistet, welche die Integrität von Daten mitunter beeinträchtigen können.

Eine kleine Auswahl ist:

- **Wasser** [bsi-grundschutz2021]. Durch Wasser können Computer abstürzen und die ungespeicherten Daten aus dem Arbeitsspeicher können verloren gehen und im schlimmsten Fall lokale Dateien beschädigen. Auch kann Wasser Datenträger komplett unbrauchbar machen, wodurch die Daten auch als verloren und unvollständig gelten.
- **Informationen oder Produkte aus unzuverlässiger Quelle** [bsi-grundschutz2021]. Nicht nur, dass die Datensätze selbst unvollständig oder fehlerhaft durch den Forscher eingegeben werden können, können auch andere Software oder Daten die Integrität beeinträchtigen. Dies kann z.B. durch Viren geschehen. Aber auch fehlerhafte Daten, welche über eine Schnittstelle importiert werden sollen, können die Integrität beeinträchtigen.
- **Manipulation von Hard- oder Software** [bsi-grundschutz2021]. Die Daten müssen irgendwo gespeichert werden. Es gibt keine Garantie, dass der Nutzer mutwillig oder unabsichtlich sich Zugang zu den gespeicherten Daten an dem System vorbei beschafft und dann Dateien löscht, hinzufügt oder bearbeitet. Im schlimmsten Fall ist die Integrität derart beschädigt, dass die Daten unbrauchbar werden.
- **Fehlfunktion von Geräten oder Systemen** [bsi-grundschutz2021]. Ein Bestandteil oder das komplette Gerät kann jederzeit komplett ausfallen oder unzuverlässig arbeiten. Dadurch kann es passieren, dass Daten unvollständig oder fehlerhaft gespeichert werden und erst beim nächsten Auslesen festgestellt wird, dass die Integrität beeinträchtigt wurde.

TODO:

- Einpflegen von BSI G 0.46

¹Die beliebte Messaging-App Telegram benutzt ein eigenes Verschlüsselungssystem MTProto für die Kommunikation mit ihren Servern. Leider gab es immer wieder Datenlecks die systematisch bedingt durch das Protokoll sind. [mtpsym-telegram-mtproto-weakness]

3.1.3 Verfügbarkeit

Systemausfälle müssen verhindert werden und die Daten sollen nach einen vorher vereinbarten Zeitrahmen wieder verfügbar sein.

Hierzu zählt eine Backupstrategie, bei der ein früherer Stand wiederhergestellt werden kann, falls der aktuelle Datenstand verloren geht.

3.1.4 Authentizität

Die Daten müssen auf Echtheit und Vertrauenswürdigkeit geprüft werden können.

3.1.5 Verbindlichkeit

Ein unzulässiges Abstreiten durchgeführter Handlungen ist nicht möglich.

3.1.6 Zurechenbarkeit

Eine durchgeführte Handlung lässt sich den Verantwortlichen jederzeit zuordnen.

3.1.7 Resilienz

Das System muss Widerstandsfähig gegen Ausspähungen, irrtümliche oder mutwillige Störungen oder absichtlichen Schädigungen (Sabotage).

3.2 Datenschutz

3.2.1 Schutz vor Fremdzugriff

Die Daten befinden sich auf physischen Geräten (Laptop, Festplatte, USB-Stick, ...) welche mit dem Forscher weltweit mitgenommen werden. Dabei kann es sein, dass die Datenträger in fremde Hände gelangen können. Um hierbei die Daten selbst zu schützen ist es zwingend erforderlich die Daten so zu schützen, dass sie selbst mit vertretbarem Aufwand nicht les- oder manipulierbar sind.

TODO:

- Verschlüsselung der Datenbank ?
- Verschlüsselung der Dateien?
- Festplattenverschlüsselung, Luks, BitLocker?
- One-Time-Pad (Einmal Passwörter)?
- Authorisierung beim Start der GUI?
- 2-Faktor-Authentifizierung?

3.2.2 Schutz der Betroffenen

In der Datenbank können Patienteninformationen oder persönliche Meinungen und Weltanschauungen vorhanden sein, welche nicht nachträglich mit dem Interviewten wieder verknüpft werden dürfen. Selbst für den Forscher dürfen diese Zusammenhänge nicht mehr nachträglich (alleinig über die Datenbank) gezogen werden. Dadurch wird auch das Datenbankschema maßgeblich beeinflusst.

TODO:

- Europäische Datenschutzgrundverordnung DSGVO (GDPR)

- Bundesdatenschutzgesetz BDSG
- Landesdatenschutzgesetze?
- DSG der betroffenen Bürger?

3.3 Technische Basis

TODO:

- Warum
- Vorteile
- Alternativen (Nachteile)

4 Technisches System

TODO:

- Überblick
- Einzelkomponenten
 - Front-End
 - Back-End
 - Datenbank
 - Zusammenhalt
- Vor- und Nachteile bei ausgewählten Sachen, Vergleiche
- Api zu DDI (und/oder andere Standards)
 - Warum? Vorteile, Nachteile
- Anmeldung
 - Sicherheit
- Software-/Hardwarearchitektur
 - 4-Tier: Entwicklung, Staging, Test, Produktion

Unterrubrik:

- Metadatenstandards
 - Hergang zu DDI
 - * Gibt es Schnittstellen, Hergang
 - DDI

4.1 Überblick

Die gesamte Anwendung ist in mehrere Module geteilt, welche für sich abgeschlossen und auch austauschbar sind. Sie sprechen über eine einfache Api miteinander und lassen sich separat voneinander testen.

Hier wird ein lokaler HTTP Webserver genutzt, welcher nur Anfragen von localhost entgegen nimmt, verarbeitet und die Antworten zurückliefert. Der Nutzer kann dann eine beliebige Anwendung (in den meisten Fällen ein moderner Webbrowser wie Firefox oder Chrome) nutzen und diesen Server ansprechen. In den folgenden Fällen wird vom diesen Modul als Back-End oder auch Server gesprochen.

Hinter dem Webserver wird eine lokale verschlüsselte Datenbank genutzt, wo die komplette Verwaltung im Prozess des Webserver eingebettet ist. Dazu wird eine Open Source Bibliothek genutzt, die sich um die komplette Verwaltung dazu kümmert.

Des weiteren gibt es das Front-End welches aus einer Webseite besteht, welche von einem beliebigen modernen Webbrowser dargestellt werden kann. Mit dieser Oberfläche wird der Nutzer hauptsächlich kommunizieren und von den Vorgängen im Hintergrund sollte dieser eigentlich nichts mitbekommen. Im folgenden wird vom Front-End auch von der Oberfläche oder auch UI gesprochen.

4.1.1 Front-End (UI)

Für das Front-End wurde eine moderne HTML Oberfläche gewählt. Diese hat den Vorteil, dass sich diese auch später ohne großen Aufwand auf andere Plattformen oder Systeme übertragen lässt. (Webbrowser sind fast überall verfügbar.) Außerdem hat sich das Web mittlerweile so weit entwickelt, dass viele Office Tätigkeiten sich auch heute schon komplett im Browser erledigen lassen (z.B. Dokumente schreiben, Emails lesen, einfache Videobearbeitung, ...) und man auf Spezialanwendungen verzichten kann.

Als Programmiersprache für die UI habe ich die Sprache Elm gewählt. Diese ist relativ noch neu und nicht so weit wie ihre Konkurrenten (z.B. JavaScript oder TypeScript) verbreitet. Dafür bietet sie mir ein paar Garantien, welche für eine schnelle und saubere Entwicklung unwiderstehlich sind:

- Ein statisches Typsystem. Es steht zu jederzeit fest welche Daten welchen Typ haben. Es gibt kein Casten oder dubiose untypisierte Objekte. Ich kann zu jeder Zeit bei einer Struktur ein Feld hinzufügen, entfernen oder den Typ ändern und der Compiler führt mich an der Hand, bis ich an allen Stellen im Code diese Änderung berücksichtigt habe.
- Funktional und ohne Seiteneffekte. Aus gleichen Eingaben kommen immer die gleichen Ausgaben. So brauche ich Funktionen nur einmal zu testen und solange ich an den Eingabedaten nichts ändere, bekomme ich auch immer das Gleiche heraus. Ich muss nicht mehr beachten, dass eine Änderung an einer Stelle eine andere Stelle kaputt machen würde.
- Es gibt keine Laufzeitfehler. Alle Prüfungen muss ich schon vorher im Code erledigen und der Compiler hilft mir auch dabei alle Fälle zu betrachten. Somit gibt es in der Ausführung nie den Fall, dass da etwas unerwartetes passiert (wie z.B. Null-Fehler).

Zudem bietet man mir hilfreiche Bibliotheken welchen einen die Entwicklung noch schneller voran bringen lässt.

Was Elm noch besonders macht im Vergleich zu anderen Systeme wie Vue oder React, dass die resultierenden Builds besonders klein und schnell sind. Dies erreicht es dadurch, dass es unbenutzten Code entfernen oder bestehenden transformieren kann, so dass es kleiner und schneller wird. Auch von den gängigen Minifizierern kann man profitieren, da sämtliche sonst unsicheren Transformationen auf einmal sicher sind.

Das Styling wird durch handgeschriebenes CSS gemacht. Hier wird kein Framework genutzt.

4.1.2 Back-End (Server)

Der Server ist eine kleine C# Anwendung, welche sich um alles Wichtige im Hintergrund kümmert. Sie nimmt alle Anfragen von der Oberfläche entgegen und informiert diese über Änderungen. Dann kümmert es sich um die Verwaltung der Datenbanken und der verschlüsselten Dateien. Hier ist der komplette Sicherheitsaspekt gelagert.

4.1.3 Datenbank

Eine Datenbank enthält all ihre Einstellungen, Zugangsberechtigungen, Dateien und eingebene Daten und Metadaten des Forschers. Diese wird verschlüsselt auf der Festplatte des Endgeräts hinterlegt und besteht in den meisten Fällen aus mehreren Dateien. Die Schlüssel selbst werden vom Server erzeugt.

4.1.4 Zusammenhalt

All diese Module werden über verschiedene Apis zusammengehalten und darüber wird auch kommuniziert.

Der Server und die UI kommunizieren hauptsächlich über eine einzelne WebSocket-Verbindung. Das hat den Vorteil, dass die Authentifizierung nur einmal am Anfang erledigt werden muss und danach kann sich gegenseitig vertraut werden, solange die Verbindung nicht abbricht (z.B. wenn der Nutzer die Seite im Browser neu lädt). Außerdem können jederzeit Nachrichten vom Server zur UI und auch anders herum sendet werden, und so schneller auf neue Ereignisse reagieren.

Für Datei-Up- und Downloads wird zusätzlich eine kleine REST-API genutzt, damit zum einen hierfür die Verbindungskapazität der WebSocket-Verbindung nicht ausgelastet wird und zum anderen die Einbettung in die Oberfläche einfacher geschieht.

Der Server und die Datenbank kommunizieren über eine Open-Source-Bibliothek, welche im Prozess des Servers angesiedelt ist. Über die Api der Bibliothek wird dann die Datenbank verwaltet. Es findet keine Inter-Process-Kommunikation statt - alle Daten sind sofort beim Server verfügbar und können nicht mit einfachen Mitteln ausgespäht werden.

4.2 Externe Apis

Um den Zugang der Daten zu anderen Anwendungen zu ermöglichen soll die Anwendung Schnittstellen bereitstellen.

4.2.1 DDI Standard

In der ethnologischen Forschung haben sich die Datenaustauschformate der DDI Alliance etabliert. Diese stellt auf ihrer Webseite verschiedene Schemas für XML, JSON und andere Dateiformaten bereit, in der man seine Daten ex- und importieren kann.

TODO:

- Research über Aufbau

4.3 Anmeldung

Die zu speichernden Daten haben einen sehr hohen Schutzbedarf (#Belege) und müssen dementsprechend verschlüsselt gespeichert und übertragen werden und brauchen eine Zugriffskontrolle. Das BSI verlangt hierfür eine Zwei-Faktor-Authentifizierung.

Für die Anwendung wurde eine Zwei-Faktor-Authentifizierung ausgewählt, welche auf Wissen (Passwort) und Besitz (FIDO-Key) basiert. Die Authentifizierung erfolgt in folgenden Schritten:

1. Der Nutzer öffnet die Oberfläche in seinem Browser und wird nach Benutzername und Passwort gefragt.

2. Der Server prüft, ob eine Datenbank diesen Nutzer hinterlegt hat. Wenn nein gibt es eine Fehlermeldung und die Authentifizierung wird abgebrochen.
3. Dann wird geprüft, ob der Wert von $\text{SHA256}(\text{Passwort} + \text{Salt})$ mit dem gespeicherten Wert übereinstimmt. Der Salt ist ein zufälliger Wert, welcher bei der Erstellung der Datenbank angelegt wurde. Falls es hier ein Fehler gab, wird dies angezeigt.
4. Die Oberfläche fragt über die WebAuthn Schnittstelle des Browser (ist in jeden modernen Browser implementiert) nach dem FIDO Key. Das ist ein kleiner spezieller USB Stick, welcher eingesteckt werden muss.
5. Der FIDO Key bekommt den Wert von $\text{SHA256}(\text{Passwort})$ und soll diesen mit seinen lokalen privaten Key signieren. Die Signatur ist immer gleich, wenn die Eingabe gleich ist.
6. Der Browser liefert die Signatur an die Oberfläche und diese an den Server.
7. Es wird geprüft, ob $\text{SHA256}(\text{Signatur})$ mit den gespeicherten Wert übereinstimmt. Wenn nicht gibt es wieder eine Fehlermeldung und ein anderer FIDO-Key wird verlangt.
8. Die Datenbank wird geöffnet.

Es können in einer Anwendung mehrere Datenbanken hinterlegt werden, wo bei jeder Datenbank der Nutzer ein anderes Passwort oder FIDO-Key nutzen kann. Am Anfang wird mit jeder Datenbank verglichen und nach und nach werden die noch gültigen Datenbank ausgesiebt, welche noch damit angemeldet werden können. Sobald zu einen Zeitpunkt keine Datenbank mehr möglich ist, so wird dies als Fehler angegeben. Am Ende können mehrere Datenbanken gleichzeitig authentifiziert werden (sofern Nutzernamen, Passwort und FIDO bei allen gleich sind). Datenbanken, welche gerade nicht geöffnet werden konnten, können auch noch nachträglich geöffnet werden. Dazu wird der Anmeldeprozess wiederholt und schon offene Datenbanken ignoriert.

Das Neuanlegen einer Anmeldeöglichkeit (z.B. bei Neuerzeugung einer Datenbank oder Hinzufügen eines weiteren Nutzers) werden folgende Schritte abgehandelt:

1. Zuerst wird geprüft, ob Zugang überhaupt besteht. Bei neuen Datenbanken ist dies implizit. Bei bestehenden muss man sich erneut anmelden, da der Server den Entschlüsselungskey nicht permanent im RAM hält
2. Der Nutzer muss einen Benutzernamen und Passwort angeben und bestätigen.
3. Nutzer wird von der Oberfläche nach einen FIDO-Key gefragt.
4. Es wird der Public-Key und eine Prüfung abgefragt.
5. Der Server prüft das. Bei Misserfolg wird dies angezeigt.
6. Die Prüfsummen für die Anmeldung werden erzeugt und bei der Datenbank parallel hinterlegt.
7. Datenbank wird verbunden, sofern noch nicht geschehen.

Des weiteren wird bei der Anmeldung nicht der eigentliche Key für die Datenbank erzeugt. Stattdessen wird für jede Anmeldeöglichkeit der eigentliche Key für die Datenbank separat verschlüsselt und ist nur mit der Signatur aus der Anmeldung entschlüsselbar. Dadurch ist es auch relativ einfach möglich mehreren Nutzern Zugang zur gleichen Datenbank zu geben, ihnen zu erlauben ihre Passwörter zu ändern oder den Zugang wiederherzustellen, falls der mal verloren gegangen ist.

5 Umsetzung

TODO:

- Stolpersteine, Probleme
 - Welche, Warum, Wie behoben
 - Technologie basiert, Produkt basiert, ...

5.1 Stolpersteine

Leider gab es bei der Umsetzung des Projekts ein paar Stolpersteine, wo ich jetzt hier versuche einige aufzulisten und zu erklären, was hier das Problem war.

5.1.1 WebAuthn

WebAuthn ist ein relativ neuer Standard. So neu, dass es zwar schon in jeden modernen Browser unterstützt wird, es aber derzeit keine offizielle Spezifikation gibt (nur Entwürfe). Des weiteren ist diese jetzt auch nicht so verbreitet, dass man sofort auch die nötige Technik parat hat, um das zu testen.

Zuerst konnte ich das nur an meinem Mobilgerät testen, da die Fingerabdruckfunktion gleichzeitig als Hardwaretoken gilt. Dies lässt sich aber nicht auf Laptops übertragen (wofür das Programm eigentlich sein soll), weshalb ich auf die FIDO-USB-Sticks gestoßen bin. Diese sollen von Browsern und am Laptop unterstützt werden.

TODO:

- Tests, Hardwaretoken fehlt noch

6 Tests

TODO:

- automatisierte Tests
 - Front-End, Integration, Test Suite, Browser-Test (Puppetier, ...)
 - Back-End, Modultests, Funktionen
- Vor- und Nachteile, Was deckt man ab, Grenzen, unentdeckte Fehlerquellen
- Temporale Tests, weitere Testarten
- Abschätzung der Nutzungsdaten in der Zukunft

7 Rollout

TODO:

- Pipeline, Container, Installskript, Webseite, ...
- Installation

8 Ausblick

TODO:

- Nachnutzung
- Wartung
- Update
- Backup (Referenz zu Sicherheit)

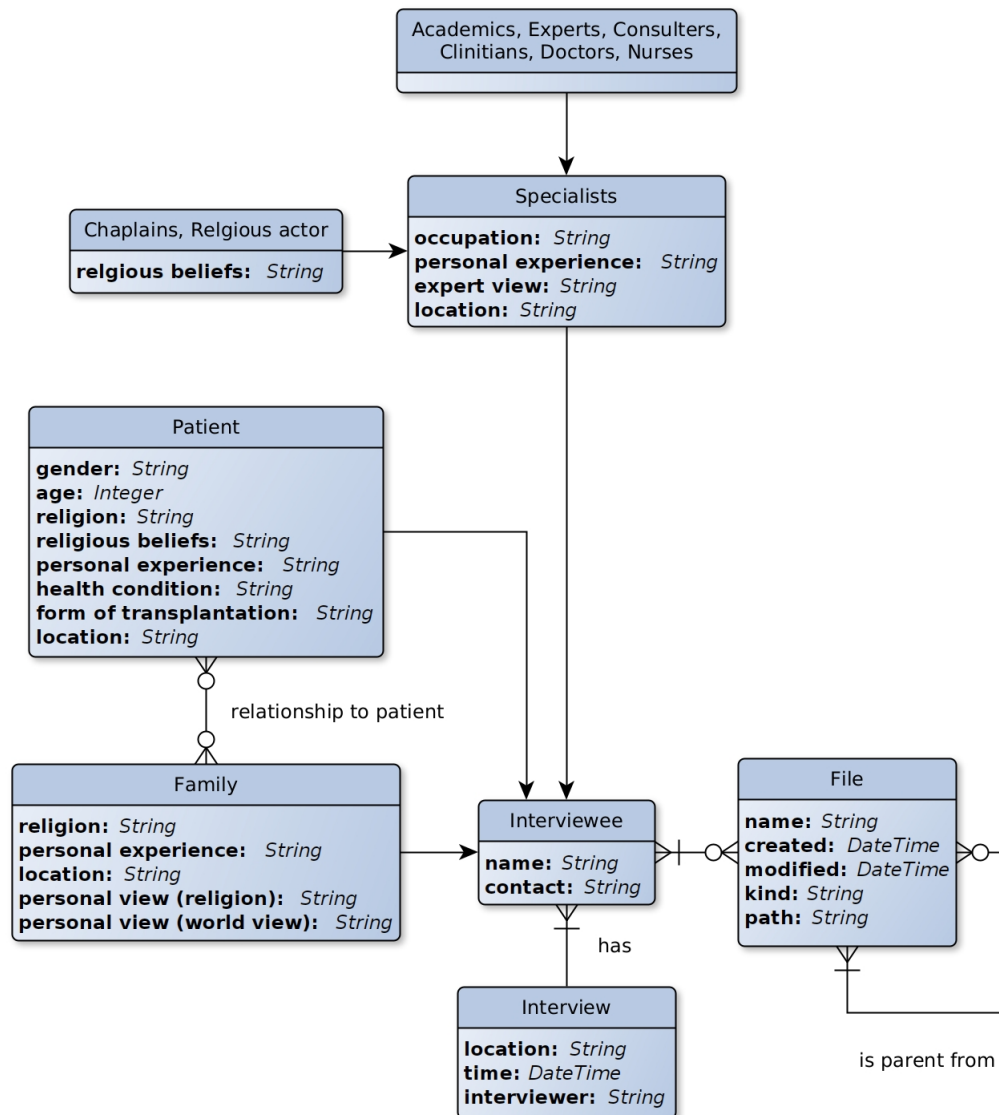
9 Abschlussbetrachtung

TODO:

- gezogene Schlüsse
- positive, negative Erkenntnisse

10 Anhang

10.1 Entity Relationship Diagramm



10.2 Projektsteckbrief

11 Literatur

Projekttitel	Planung & Entwicklung einer Datenbankanwendung für das Forschungsdatenmanagement		
Projektnummer	2021-SWD-19526		
Projektleitung	Max Brauer		
Auftraggeber	Sebastian Ehser, Max-Planck-Institut für ethnologische Forschung		
Projektnutzen	Verwaltung von Forschungsdaten		
Projektumfeld			
Projektinhalt / -ziele	<p>Erstellen eine Datenbankanwendung zur Aufnahme von Forschungsdaten für Forschende eines ethnologischen Instituts. Diese Anwendung soll den Forschenden auf seinen Reisen begleiten und die Arbeit mit seinen Daten erleichtern. Dabei soll ein besonderes Augenmerk auf die Sicherheit gelegt werden.</p> <p>Als Nichtziel wurden zusätzliche Relationen zwischen den Interviewten (außer den vorgeschriebenen) festgelegt.</p> <p>(prototypisch Start mit Raza und das Ziel das auf das gesamte Institut umzusetzen)</p>		
Projektnutzen	<ul style="list-style-type: none"> • Vereinfachung in der Arbeitsweise der Forschenden durch zentrale und strukturierte Speicherung und Darstellung von Daten • Sicherung der Daten durch Backups und Zugriffskontrollen • Weniger Papierverbrauch bei der Arbeit mit den Daten 		
Klärungs-/ Unterstützungsbedarf	<ul style="list-style-type: none"> • Genaues Modell der zu speichernden Daten • Aktuelle Arbeitsweise der Forschenden 		
Start / Ende	Oktober 2021 - Januar 2022		
Zwischentermine	<ul style="list-style-type: none"> • Anmeldung der Bachelorarbeit: offen • Vorstellen der Zwischenergebnisse: im 2-4 Wochen Rythmus • Verteidigung der Arbeit: 5-6 Monate nach Anmeldung der Arbeit • Alphaversion: • Betaversion: • Releaseversion: ähnlich zur Verteidigung 		
Aufwand	Software: 112-150 Stunden	Schriftliche Arbeit: 300 Stunden	Gesamt: 450 Stunden
Beteiligte	<ul style="list-style-type: none"> • Max Brauer (Entwickler und Verfasser der Arbeit) • Sebastian Ehser (Auftraggeber, Projektberater MPI) • Christian Kieser (technischer Berater MPI) • Farah Raza (Bedarfsträgerin) 		
Risiken	<ul style="list-style-type: none"> • Krankheit oder sonstige Ausfälle • Software wird nicht rechtzeitig fertig • Software entspricht nicht oder nicht komplett den Wünschen der Kunden 		