Duale Hochschule Baden-Württemberg Mannheim

Wintersemester 2022

Webanwendung für die Speicherung und Verwaltung medizinischer Daten

Eine Projektgruppenarbeit im Modul
Security by Design

betreut von
Dr. Jürgen Schneider

erstellt von
David Fambach,
Jonas Grohe und
Vincent Hundeloh

im Rahmen des Kurses
TINF20CS1

im Bearbeitungszeitraum
10.10.2022 bis 12.12.2022

Inhaltsverzeichnis

1 Ziel und Kontext 1
1.1 Regulatorische Vorschriften 1
2 Architekturüberblick 2
2.1 Architekturentscheidung: Frontend 2
2.2 Architekturentscheidung: Web Framework 3
2.3 Architekturentscheidung: Programmiersprache 3
2.4 Architekturentscheidung: Datenbank 4
3 Bedrohungsanalyse 5
3.1 Schutzziele 5
3.2 Sicherheitsmaßnahmen 7
3.3 Risikoregister 8
4 Konzepte 8
4.1 Konzept: Schützen von Data-at-rest 8
4.2 Konzept: Schützen von konfigurierten Geheimnissen produktiver Systeme 11
5 Anhang 13
5.1 Externe Inhalte 13
5.2 Darstellungen 14

Darstellungsverzeichnis

Darstellung 1: Architekturüberblick (abstrakt) 14
Darstellung 2: Architekturüberblick (technisch) 15
Darstellung 3: Architekturüberblick mit Schutzobjekten und Vertrauensgrenzen 16
Darstellung 4: Architekturüberblick mit Schutzobjekten und Sicherheitsmaßnahmen 17
Darstellung 5: Entscheidungsmatrix Architekturentscheidung Frontend 18
Darstellung 6: Entscheidungsmatrix Architekturentscheidung Web Framework 19
Darstellung 7: Entscheidungsmatrix Architekturentscheidung Programmiersprache 21
Darstellung 8: Entscheidungsmatrix Architekturentscheidung Datenbank 22

1 Ziel und Kontext

Ziel ist die Bereitstellung einer Webanwendung, die es Patienten ermöglicht, Gesundheitsdaten zu verwalten und mit Ärzten und anderen Personen zu teilen, also lesenden Zugriff einzuräumen. Dadurch wird sichergestellt, dass Patienten und Ärzten stets aktuelle und vollständige Unterlagen zur Verfügung stehen. Ein besonderer Fokus liegt auf dem Schutz der Vertraulichkeit und Integrität dieser Daten.

1.1 Regulatorische Vorschriften

Für die Entwicklung und den Betrieb einer solchen Anwendung sind verschiedene deutsche und europäische Vorschriften von Bedeutung. Im folgenden werden einige Implikationen ausgewählter bestehender und zukünftiger Vorschriften für den Entwicklungsprozess einer solchen Anwendung, die Anwendung selbst und den Betrieb derselben untersucht.

Die europäische Datenschutz-Grundverordnung (Verordnung EU 2016/679) enthält Bestimmungen zur Verarbeitung personenbezogener Daten mit dem Ziel die Rechte natürlicher Personen im Bezug auf diese Daten zu stärken. Weil sich die DSGVO nur auf die Verarbeitung personenbezogener Daten bezieht, betrifft sie nur den Betrieb der Anwendung direkt. Allerdings erfordert der DSGVO-konforme Betrieb einer Anwendung zur Verwaltung von Gesundheitsdaten notwendigerweise auch bestimmte Eigenschaften der Anwendung selbst, sodass die meisten Vorschriften der DSGVO sowohl als Anforderung an die Anwendung als auch an deren Betrieb verstanden werden müssen. Die DSGVO fordert insbesondere, dass …

- weil es sich bei den verarbeiteten personenbezogenen Daten um Gesundheitsdaten handelt, eine ausdrückliche Einwilligung der betroffenen Person vor der Verarbeitung eingeholt wird (Art. 9 Abs. 1, 2 DSGVO).
- Rahmenbedingungen für die Einwilligung eingehalten werden (Art. 7, 8 DSGVO).
- Informationspflichten eingehalten werden können (Art. 13, 15 DSGVO).
- betroffenen Personen ermöglicht wird, bestimmte Betroffenenrechte auszuüben (Art. 16-20 DSGVO).
- risikobasiert technische und organisatorische Maßnahmen zum Schutz der Verarbeitung personenbezogener Daten sowie deren Integrität und Vertraulichkeit (Art. 5 Abs. 1 lit. f DSGVO, Art. 24 Abs. 1 DSGVO, Art. 32 DSGVO).
- Voreinstellungen der Anwendung datenschutzfreundlich sind (Art. 25 Abs. 1, 2 DSGVO).
- keine personenbezogenen Daten erfasst werden, die keinem eindeutigen Zweck dienen oder nicht länger erforderlich sind oder deren Zweck über das für die Anwendung notwendige Maß hinausgeht (Art. 5 Abs. 1 lit. b, c, e).

Die vorgenannten Forderungen konzentrieren sich auf Aspekte, die bereits bei der Entwicklung der Anwendung zu berücksichtigen sind. Für den Betrieb der Anwendung sind weitere Aspekte relevant, unter Anderem die Benennung von Kontaktpersonen und einem Datenschutzbeauftragten oder die Information von Aufsichtsbehörden bei Datenschutzvorfällen.

Weil für die Verarbeitung von Gesundheitsdaten bereits eine Einwilligung des Betroffenen eingeholt werden muss, wird auch für personenbezogene Daten, die nicht unter Art. 9 DSGVO fallen, eine Einwilligung als Rechtsgrund der Verarbeitung eingesetzt.

Darüber hinaus ist ein angemessener Schutz von Vertraulichkeit, Integrität und Verfügbarkeit sensibler Daten für Entwickler und Betreiber einer Anwendung, deren Zweck der Umgang mit diesen Daten ist, von erheblicher Bedeutung, weil Datenpannen insbesondere in diesem Bereich erhebliche Rufschädigungen herbeiführen können, die unter Umständen auch geschäftsgefährdend sein können. Je nach Rechtsform eines Unternehmens existieren verschiedene Vorschriften, die die Geschäftsführung verpflichten, auf die Erkennung oder Behebung dieser Probleme hinzuwirken (vgl. bspw. § 91 Abs. 2 AktG).

In der Zukunft können zusätzliche Vorschriften relevant werden, beispielsweise durch den vorgeschlagenen EU Cyber Resillience Act, der Sicherheitsvorgaben für Produkte mit digitalen Komponenten macht. Für EHR-Systeme nach der vorgeschlagenen European Health Data Space Regulation, was unter anderem

Lösungen zur Verwaltung von Gesundheitsdaten umfasst, wird Konformität zu einem Anforderungskatalog gefordert (vgl. Art. 24 Abs. 4 Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022). Dieser Anforderungskatalog betrifft sowohl den Entwicklungsprozess der Anwendung als auch die Anwendung selbst als auch den Betrieb der Anwendung. Er fordert insbesondere, dass …

- der Entwicklungsprozess eine Bedrohungsanalyse umfasst (Art. 10 Abs. 3 Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022).
- der Entwicklungsprozess in Anbetracht bestehender Risiken angemessen ist (Anhang 1 Abs. 1 Nr. 1 Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022).
- der Entwicklungsprozess zur Behandlung von Schwachstellen fähig ist (Anhang 1 Abs. 2 Nr. 2 Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022).
- der Entwicklungsprozess angemessene Sicherheitstests vorsieht (Anhang 1 Abs. 2 Nr. 3 Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022).
- für den Entwicklungsprozess und den Betrieb der Anwendung eine Minimierung der Angriffsoberfläche und eine Minimierung der Auswirkungen von Sicherheitsvorfällen (Anhang 1 Abs. 1 Nr. 3 lit. h, i Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022).
- für die Anwendung eine Freiheit von Schwachstellen zur Auslieferung (Anhang 1 Abs. 1 Nr. 2 Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022).
- für die Anwendung …
  - sichere Standardeinstellungen,
  - einen Schutz gegen nicht autorisierten Zugriff,
  - einen Schutz der Vertraulichkeit von verarbeiteten Daten, unabhängig davon, ob sie personenbezogen sind, während der Speicherung und Übertragung,
  - einen Schutz der Integrität dieser Daten,
  - eine Datenminimierung für personenbezogene und nicht personenbezogene Daten,
  - einen Schutz der Verfügbarkeit der Kernfunktionen sowie einen Schutz gegen DoS-Angriffe,
  - eine Minimierung der von dem Produkt für andere Dienste ausgehenden Gefahren,
  - ein Logging von sicherheitsrelevanten Ereignissen und
  - die Fähigkeit zu Sicherheitsaktualisierungen (Anhang 1 Abs. 1 Nr. 3 lit. a-g, j, k Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022).

Diese Aspekte sind derzeit noch nicht verbindlich, aber werden in der Zukunft wahrscheinlich mindestens in ähnlicher Form vom Gesetzgeber eingefordert werden. Neben den genannten rechtlichen Vorgaben gibt es weitere Gründe, Sicherheitsaspekte bei der Entwicklung verstärkt zu berücksichtigen, unter Anderem die oben genannte Gefahr der Rufschädigung bei Pannen, aber auch eine Verringerung der notwendig werdenden Korrekturmaßnahmen können sowohl für Softwarehersteller als auch -betreiber eine Aufwands- und damit eine Kostenreduktion darstellen.

2 Architekturüberblick

Die abstrakte Struktur der Anwendung ist in Darstellung 1. Anhand der in den folgenden Abschnitten beschriebenen Architekturentscheidungen ergibt sich der in Darstellung 2 gezeigte Technologieüberblick.

2.1 Architekturentscheidung: Frontend

Bereich: Eingesetzte externe Komponenten
Thema: Frontend Framework
Nummer: AD1
Architekturentscheidung: Verwendung von Vue als Frontend Framework

Problemstellung	Es ist ein Frontend Framework auszuwählen, welches ausgewählte Anforderungen bestmöglich erfüllt
Annahmen	
Motivation	Ein passendes Frontend erleichtert die Programmierarbeit und vermindert

	nötige Sicherheitsanpassungen.
Alternativen	Option 1: React Option 2: Vue Option 3: Angular
Entscheidung	Verwende Option 2: Vue
Begründung	Siehe Entscheidungsmatrix
Implikationen	Keine
Abgeleitete Anforderungen	Keine
Zugehörige Entscheidungen	Keine

Die Entscheidungsmatrix zu dieser Architekturentscheidung ist als Darstellung 5 angehängt.

2.2 Architekturentscheidung: Web Framework

Bereich: Eingesetzte externe Komponenten
Thema: Web Framework
Nummer: AD2
Architekturentscheidung: Verwendung von Django als Web Framework

Problemstellung	Auswahl eines Webframeworks
Annahmen	- Webframework basierend auf der Programmiersprache Python
Motivation	
Alternativen	Option 1: Django (Makroframework) Option 2: Flask (Microframework) Option 3: Bottle
Entscheidung	Verwende Django
Begründung	Siehe Entscheidungsmatrix
Implikationen	Keine
Abgeleitete Anforderungen	Keine
Zugehörige Entscheidungen	Architekturentscheidung Programmiersprache: Python

Die Entscheidungsmatrix zu dieser Architekturentscheidung ist als Darstellung 6 angehängt.

2.3 Architekturentscheidung: Programmiersprache

Bereich: Eingesetzte externe Komponenten
Thema: Programmiersprache
Nummer: AD3
Architekturentscheidung: Verwendung von Python als Programmiersprache

Problemstellung	Es ist eine Programmiersprache auszuwählen, welches ausgewählte Anforderungen bestmöglich erfüllt.
Annahmen	Keine
Motivation	Ein passende Programmiersprache erleichtert den Programmierprozess und schont Ressourcen .
Alternativen	Option 1: Phyton Option 2: Java Option 3: C++
Entscheidung	Verwende Option 1: Python
Begründung	Siehe Entscheidungsmatrix
Implikationen	Keine
Abgeleitete Anforderungen	Keine
Zugehörige Entscheidungen	Keine

Die Entscheidungsmatrix zu dieser Architekturentscheidung ist als Darstellung 7 angehängt.

2.4 Architekturentscheidung: Datenbank

Bereich: Eingesetzte externe Komponenten
Thema: Datenbank
Nummer: AD4
Architekturentscheidung: Verwendung von Postgres als Datenbankmanagementsystem (DBMS)

Problemstellung	Es ist ein Datenbankmanagementsystem auszuwählen, das geignet ist, die durch die Anwendung anfallenden Daten zu verwalten. Dabei ist nur eine Datenbankinstanz zu verwenden, die aber auch nur von einer Instanz des Dateiablagedienstes verwendet wird.
Annahmen	- Python wird als Programmiersprache für den Dateiablagedienst eingesetzt - Es wird eine einzige, zentrale Datenbankinstanz eingesetzt. - Nur der Dateiablagedienst muss regulär in der Lage sein, auf die Datenbank zuzugreifen und von diesem Dienst ist nur eine Instanz gleichzeitig aktiv. - Die Datenbankinstanz muss als Docker-Container lauffähig sein.
Motivation	Sicheres Ablegen von Daten mit validierter Struktur.
Alternativen	MySQL (relational) PostgreSQL (relational) MongoDB (dokumentenorientiert) Apache CouchDB (dokumentenorientiert) SQLite (relational) Microsoft SQL Server (relational) Oracle Datenbank (relational)
Entscheidung	Verwende PostgreSQL
Begründung	Weil es sich bei der Aufgabe um ein Beispielprojekt handelt, sollten Lizenzkosten dringend vermieden werden. Daher werden nur kostenlose Alternativen betrachtet. Weil Aus demselben Grund wird DBaaS nicht weiter betrachtet. Die abzulegenden Datensätze haben sehr ähnliche Strukturen. Das gilt für Benutzerprofilinformationen und Metainformationen zu Dateien. Von Benutzern hochgeladene Dateien sind nicht näher definierte, aber tendenziell große Binärdaten. Für die ersteren beiden Datentypen sind relationale Datenbanken aufrauend ihrer Struktur besonders geeigenet, da sie anders als dokumentenorientierte Datenbanken wirksam die korrekte Struktur erzwingen. Objektorientierte und Key-Value-Datenbanken sind insbesondere für beliebige Binärdaten weniger geeignet sind und liefern auch für die anderen Datentypen keine besonderen Vorteile. Daher werden nur relationale und dokumentorientierte Datenbanken weiter betrachet. Alle betrachteten Datenbanksysteme können in einem Docker-Container betrieben werden. Für SQLite würde dies bedeuten, im Container des Dateiablagedienstes abgelegt zu werden. Das ist hinnehmbar, weil nur eine Instanz des Dateiablagedienstes notwendig ist. Gegen SQLite spricht der im Vergleich zu den anderen Datenbanksystemen stark eingeschränkte Funktionsumfang und die fehlende Eignung für Binärdateien. Von den übrigen relationalen Datenbankmanagementsystemen ist für PostgreSQL das meiste Vorwissen vorhanden und PostgreSQL schneidet sowohl laut [1] als auch erwartungsgemäß hinsichtlich Performance am besten ab.
Implikationen	- Die relationale Datenstruktur muss entweder bei Installation angelegt

	werden oder durch die Anwendung vor der Verwendung erzeugt werden. - Die Unterstützung für große Binärdaten steht für das ablegen beliebiger Dateien zur Verfügung - Es muss ein PostgreSQL-Datenbankadapter für den Dateiablagedienst eingesetzt werden. - Das Ablegen von nicht strukturierten oder unvollständigen Datensätzen ist außerhalb der rohen Binärdaten nicht möglich.
Abgeleitete Anforderungen	F22, F23, F24, NF10
Zugehörige Entscheidungen	Keine

Die Entscheidungsmatrix zu dieser Architekturentscheidung ist als Darstellung 8 angehängt.

3 Bedrohungsanalyse

Betrachtet werden folgende Schutzobjekte:

Nr.	Bezeichnung	Bemerkung
A1	Webserver	
A2	Dateiablagedienst	
A3	Datenbankserver	
A4	Konfigurationsdaten Webserver	
A5	Logdaten Webserver	
A6	Konfigurationsdaten Dateiablagedienst	
A7	Logdaten Dateiablagedienst	
A8	Gesundheitsdaten	
A9	Persönliche Daten	
A10	Anmeldedaten für Benutzer	
A11	Konfigurationsdaten Datenbank	
A12	Logdaten Datenbank	
A13	Statische Webseiteninhalte	
A14	Anmeldedaten für Webserveradministratoren	
A15	Anmeldedaten für Anwendungsadministratoren	
A16	Anmeldedaten für Datenbankadministratoren	
A17	Sitzungsdaten für Benutzer	
A18	Sitzungskennzeichen für Benutzer	
A19	Logserver	
A20	Konfigurationsdaten Logserver	

Darstellung 3 verortet die Schutzobjekte im Architekturüberblick.

3.1 Schutzziele

Die folgende Tabelle ordnet Schutzobjekten anhand ihrer Nummer relevante Schutzziele, Vertraulichkeit (Vtr.), Integrität (Int.) und Verfügbarkeit (Vrf.) zu. Wird für mehrere Schutzziele eine Angabe gemacht, ist das Schutzziele mit der kleineren Angabe wichtiger.

Nr.	Vtr.	Int.	Vrf.	Bemerkung
A1		1.	2.	Arbeitet das System nicht wie vorgesehen, ist also die Integrität des Systems verletzt, kann das zum einen zum Verlust der Verfügbarkeit führen. Ist einem Dritten eine Manipulation des Systems möglich, hängen die Auswirkungen davon ab, welche Arten der Manipulation möglich ist. Im schlimmsten Fall kann die

Nr.	Vtr.	Int.	Vrf.	Bemerkung
				Kontrolle über das System erlangt werden, wodurch auch die Vertraulichkeit aller Daten verloren geht, die das System passieren, ohne dass zusätzliche Schutzmaßnahmen bestehen, beispielsweise durch eine Verschlüsselung mit einem Schlüssel, der auf dem kompromittierten System nicht vorliegt. Weil von einem solchen kompromittierten System auch Gefahr für andere Systeme ausgehen kann, fordert der vorgeschlagene EU Cyber Resillience Act eine entsprechende Absicherung (Anhang 1 Abs. 1 Nr. 3 lit. g Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022). Ist das System nicht Verfügbar, kann die Anwendung nicht verwendet werden. Auch ein Schutz der Verfügbarkeit wird durch den vorgeschlagenen EU Cyber Resillience Act gefordert (Anhang 1 Abs. 1 Nr. 3 lit. f Proposed EU Cyber Resilience Act i. d. F. v. 15. September 2022).
A2		1.	2.	s. Bemerkung zu A1
A3		1.	2.	s. Bemerkung zu A1
A4	3.	1.	2.	Die Integrität der Konfigurationsdaten ist im Wesentlichen ähnlich zur Integrität des entsprechenden Systems, nur auf einer niedrigeren Ebene. Eine Integritätsverletzung kann einen Ausfall zur Folge haben und die Folgen einer möglichen Manipulation durch einen Dritten ist abhängig vom Umfang der Möglichkeiten der Manipulation Vertraulichkeit ist für Konfigurationsdaten nicht notwendigerweise relevant, weil sich viele Konfigurationsparameter durch das Verhalten der Anwendung nach außen zeigen. Allerdings liefert eine Offenlegung der Konfiguration potenziellen Angreifern gegebenenfalls zusätzliche Informationen über das System und seine Arbeitsweise, weswegen eine Wahrung der Vertraulichkeit wünschenswert ist. Bei der hier vorgenommen Bewertung wird angenommen, dass die Konfigurationsdaten keine Schlüssel oder Ähnliches umfassen, für die Vertraulichkeit dringend geboten ist, noch vor Integrität und Verfügbarkeit (vgl. auch das Konzept "Schützen von konfigurierten Geheimnissen produktiver Systeme").
A5	3.	1.	2.	Eine Integritätsverletzung bei Protokolldaten führt dazu, dass sie ihren Zweck nicht mehr erfüllen können. Eine Integritätsverletzung hat daher für die betroffenen Bereiche dieselben Folgen wie eine Verfügbarkeitsverletzung, wenn die Integritätsverletzung festgestellt werden kann. Eine unbemerkte Integritätsverletzung kann zu falschen Annahmen führen und ist daher ungünstiger als der Verlust der Verfügbarkeit. Die Vertraulichkeit von Protokolldaten ist insofern von Bedeutung, dass die Protokolle unter Umständen ein Nutzungsverhalten aufzeigen können, sowie Aufschluss über das Verhalten des Systems geben, der möglicherweise Angriffe auf das System erleichtert. Es wird angenommen, dass keine sensiblen Daten in Protokollen auftauchen, beispielsweise Schlüssel zu kryptografischen Verfahren. Wäre dies der Fall, müsste die Vertraulichkeitsanforderung entsprechend höher ausfallen. Eine Vertraulichkeitsverletzung führt nicht dazu, dass das Protokoll unbrauchbar wird.
A6	3.	1.	2.	s. Bemerkung zu A4
A7	3.	1.	2.	s. Bemerkung zu A5
A8	1.	1.	2.	Bei den Gesundheitsdaten handelt es sich um sensible persönliche Daten, für die daher eine hohe Vertraulichkeitsanforderung besteht. Gleichzeitig muss die Integrität insoweit sichergestellt sein, dass keine unbemerkte Manipulation möglich ist, damit Patienten und Ärzten nur korrekte Informationen bereitgestellt werden. Eine Verletzung der Verfügbarkeit ist nicht in gleichem Maß kritisch, ist aber für Benutzer äußerst unpraktisch und sollte daher vermieden werden. Die DSGVO fordert für diese Daten, dass Vertraulichkeit und Integrität gewahrt werden und dass hierzu geeignete Maßnahmen ergriffen werden (Art. 5 Abs. 1 lit. f

Nr.	Vtr.	Int.	Vrf.	Bemerkung
				DSGVO, Art. 24 Abs. 1 DSGVO, Art. 32 DSGVO).
A9	1.	2.	3.	Bei den persönlichen Daten der Benutzer handelt es sich ebenso um sensible persönliche Daten. Dementsprechend besteht auch hier eine hohe Vertraulichkeits- und Integritätsanforderung. Allerdings wird die Vertraulichkeit hier über die Integrität gestellt, weil eine Verletzung der Integrität nachträglich korrigiert werden kann, eine Verletzung der Vertraulichkeit jedoch nicht. Beides ist jedoch auch hier wichtiger als die Wahrung der Verfügbarkeit. Die Forderung der DSGVO nach Vertraulichkeits- und Integritätsschutz besteht auch für diese Daten (Art. 5 Abs. 1 lit. f DSGVO, Art. 24 Abs. 1 DSGVO, Art. 32 DSGVO).
A10	1.	2.	2.	Die Vertraulichkeit von Anmeldedaten/Sitzungsdaten ist von hoher Wichtigkeit, weil Dritte sie gegebenenfalls dazu einsetzen können, die Identität des Benutzers anzunehmen, zu dem sie gehören. Ein Integritäts- oder Verfügbarkeitsverlust führt lediglich dazu, dass der Inhaber sie nicht verwenden kann.
A11	3.	1.	2.	s. Bemerkung zu A4
A12	3.	1.	2.	s. Bemerkung zu A5
A13		1.	2.	Die Integrität der statischen Webseiteninhalte ist ist von hoher Bedeutung, weil sie auch die Anwendungslogik beinhalten. Eine gezielte Manipulation der Anwendungslogik kann dazu führen, dass sich Benutzer an einem gefälschten Server authentisieren oder dass Eingabedaten abgegriffen werden. Die Verfügbarkeit dieser Daten ist für den Betrieb der Anwendung erforderlich, weil ansonsten ein Aufruf der Webseite nicht möglich ist. Vertraulichkeit ist nicht erforderlich, weil die Seite so aufgebaut ist, dass sie an beliebige nicht authentifizierte Benutzer übertragen werden kann.
A14	1.	2.	2.	s. Bemerkung zu A10
A15	1.	2.	2.	s. Bemerkung zu A10
A16	1.	2.	2.	s. Bemerkung zu A10
A17		1.		Die Sitzungsdaten umfassen lediglich technische Daten über die Sitzung, den Sitzungsbeginn, die für die Sitzung aufgewendete Bandbreite und die Anzahl der verarbeiteten Anfragen. Sie dienen dazu, eine übermäßige Belastung der Anwendung automatisch erkennen zu können. Damit die Daten sinnvoll zu diesem Zweck eingesetzt werden können, muss ihre Integrität sichergestellt sein. Eine besondere Anforderung an die Verfügbarkeit besteht nicht, weil die Anwendung auch ohne diese Daten weiter funktioniert, sei es, indem sie die Daten neu anlegt oder indem sie Sitzungen bei Verlust beendet und einen erneuten Verbindungsaufbau anfordert. Auch an die Vertraulichkeit besteht im Vergleich zu den übrigen Schutzobjekten keine besondere Anforderung.
A18	1.	2.	2.	s. Bemerkung zu A10
A19		1.	2.	s. Bemerkung zu A1
A20	3.	1.	2.	s. Bemerkung zu A4

3.2 Sicherheitsmaßnahmen

Die folgenden allgemeinen Sicherheitsmaßnahmen werden ergriffen. Darstellung 4 verortet diese Sicherheitsmaßnahmen, sofern dies nützlich ist, im Architekturüberblick.

Nr.	Maßnahme	Bemerkung
SC1	Verwende TLS in Version 1.2 oder höher	vgl. BSI TR-02102-2, vgl. R3
SC2	Verwende SSH in Version 2 oder höher	vgl. BSI TR-02102-4
SC3	Eingabevalidierung	vgl. R4, R5
SC4	Authentifizierung	vgl. R1

Nr.	Maßnahme	Bemerkung
SC5	Setzen von HTTP-Headern zur Mitigation von XSS	vgl. R5
SC6	Autorisierung	vgl. R2
SC7	Schutz gegen übermäßige Speicherverwendung durch Nutzer	vgl. R8
SC8	Schutz gegen Sitzungsübernahme und -vorhersage	vgl. R5
SC9	Restriktion des Zugriffs auf konfigurierte Geheimnisse	vgl. R10, vgl. Konzept „Schützen von konfigurierten Geheimnissen produktiver Systeme“
SC10	Verwendung von TLS-Serverzertifikaten	vgl. R14
SC11	Verwendung kryptografisch starker Hashfunktionen, wenn kryptografische Stärke relevant ist, beispielsweise SHA-256, SHA-384 oder SHA-512, in effizienter Implementierung, die im vorgesehen Anwendungsfall zu konfigurierende Parameter, insbesondere Salz, Rechenzeitaufwandsfaktor und Speicherplatzaufwandsfaktor, bereits in ihrer Konstruktion berücksichtigt, sofern vorhanden.	vgl. BSI TR-02102-1
SC12	Verwendung kryptografisch starker Blockchiffren, wenn kryptografische Stärke relevant ist, beispielsweise AES-128, AES-192, AES-256 in einem dem Zweck angemessenen Operationmodus	vgl. BSI TR-02102-1
SC13	Verwendung kryptografisch starker asymmetrischer Verschlüsselungsverfahren	vgl. BSI TR-02102-1

3.3 Risikoregister

Das Risikoregister liegt gemäß Aufgabenstellung in einem gesonderten Dokument bei.

4 Konzepte

4.1 Konzept: Schützen von Data-at-rest

Der Schutz der Vertraulichkeit der über die Plattform geteilten Daten ist von besonderer Bedeutung. Zunächst muss für jede Kommunikation zwischen den Anwendungskomponenten sowie für jede Kommunikation zwischen einem Benutzer und der Anwendung eine Vertraulichkeit und Integrität erzielt werden. Vertraulichkeit bedeutet, dass die von Benutzern übertragenen Dateien und Befehle sowie vom Webserver zurückgegebene Dateilisten und Dateien nicht gegenüber an der Kommunikation nicht beteiligten Dritten offengelegt werden. Integrität bedeutet, dass die oben benannten Daten gegen die Manipulation durch einen solchen Dritten zu schützen sind. Zu unterscheiden sind hier schwache Integrität und starke Integrität. Erstere fordert, dass eine unbefugte Manipulation nicht in einer Weise möglich ist, dass sie den Zugriffsberechtigen nicht jederzeit auffällt. Zweitere fordert, dass eine unbefugte Manipulation nicht möglich ist. Für Kommunikationsverbindungen im Internet ist existieren im Allgemeinen allerdings stets solche Dritte, die in der Lage sind, die Kommunikation vollständig zu unterbrechen und so die starke Integrität zu verletzen. Inhärent ist es also für eine Kommunikation durchs Internet im Allgemeinen nicht möglich, starke Integrität wirksam zu erzielen. Daher wird für Kommunikationsverbindungen durchs Internet für diese Anwendung nur schwache Integrität gefordert, weil damit zumindest für ungestört übertragene Daten eine Manipulationsfreiheit verifizierbar ist. Um auf Kommunikationsverbindungen die Vertraulichkeit und schwache Integrität zu sichern, wird für alle Verbindungen TLS eingesetzt.

Nicht alle Teile der Kommunikation erfordern notwendigerweise Vertraulichkeit und schwache Integrität. Beispielsweise ist für die Übertragung HTML-, CSS- und JavaScript-Ressourcen der Webseite, die den allgemeinen Seitenaufbau vorgeben und explizit keinen darzustellenden Nutzinhalt umfassen, keine Vertraulichkeit der Übertragung nötig. Das ist offensichtlich, weil diese Ressourcen öffentlich für jeden

zugreifbar abgelegt werden. Es ist aber zu beachten, dass in diesem Fall aber ein Integritätsschutz unabdingbar ist. Weil die Kosten des TLS-Einsatzes für den Webseitenbetrieb vernachlässigbar sind, wird das Verfahren grundsätzlich für alle Verbindungen eingesetzt. Webclients wird auf diese Weise außerdem ermöglicht, eine Authentizitätsprüfung für den Webserver durchzuführen, indem das entsprechende Serverzertifikat überprüft wird. Dieselbe Technik wird bei der Kommunikation zwischen den einzelnen Anwendungskomponenten untereinander eingesetzt.

Sobald die von Benutzern geteilten Daten in der Datenbank abgelegt wurden, können auch Maßnahmen zum Erzielen einer starken Integrität getroffen werden. Dafür werden Benutzersitzungen authentifiziert und Autorisierungsprüfungen durchgeführt, bevor Befehle ausgeführt werden, die die zu schützenden Daten betreffen. Durch konsequentes Ablehnen nicht zulässiger Operationen wird für den Datenbankinhalt starke Integrität gegen Angriffe durch Dritte erzielt. Für privilegierte Benutzer, wie unten beschrieben, gilt dies offensichtlich nicht.

Bisher nicht berücksichtigt wurde der Schutz der Vertraulichkeit von Benutzerdaten gegen die unbefugte Einsichtnahme durch privilegierte Benutzer, die zur Konfiguration und Wartung der Anwendung weitreichende Zugriffsmöglichkeiten auf verschiedene Systeme der Anwendung besitzen. Dazu zählen insbesondere der Zugriff auf die rohen Datenbankinhalte sowie der Zugriff auf private kryptographische Schlüssel, die von der TLS-Servern für einen Identitätsnachweis eingesetzt werden. Erstere Zugriffsmöglichkeit ist bereits hinreichend, um beliebige von Benutzern abgelegte Dateien einsehen zu können, sofern keine weiteren Schutzmechanismen dies verhindern.

Problematisch ist in diesem Fall, dass aufgrund der weitreichenden Kontrolle über die relevanten Systeme kryptografische Verfahren nur dann vor dieser Benutzergruppe schützen, wenn diese Verfahren zum Schutz der Daten für den gesamten Zeitraum, in dem sich die Daten auf einem zentralen System der Anwendung befinden, dauerhaft eingesetzt werden und die zum Einsehen der Daten erforderlichen kryptografische Schlüssel auf diesen Systemen nicht zur Verfügung stehen.

Eine solche Absicherung kann nur realisiert werden, indem die Daten vor dem Hochladen durch die Webanwendung verschlüsselt werden und erst zur Einsicht nach dem Herunterladen wieder entschlüsselt werden. Es ist also neben der oben beschriebenen Verschlüsselung durch TLS eine weitere Verschlüsselung der Nutzdaten erforderlich. Der dafür verwendete Schlüssel darf auf den zentralen Systemen der Anwendung nicht zugänglich werden. Gleichzeitig ist es aber für die Freigabe von Dateien für den Zugriff durch andere Nutzer notwendig, dass diesen Benutzern der Schlüssel zur Verfügung steht. Das lässt sich realisieren, indem jedem Benutzer ein asymmetrischer Schlüssel zugeordnet wird. Die öffentlichen Komponenten dieser Schlüssel können über die Anwendung zentral bekanntgemacht werden. Zur Verschlüsselung der Dateien wird dann ein pro Datei gewählter symmetrischer Schlüssel verwendet. Dieser Schlüssel wird bei Änderungen an der Datei oder wenn Freigaben entzogen werden, neu kryptografisch sicher zufällig gewählt. Dieser Schlüssel kann dann jeweils mit den öffentlichen Schlüsseln aller Benutzer, die einen Zugriff auf die Datei erhalten sollen, verschlüsselt. Diese verschlüsselten symmetrischen Schlüssel können ebenso über die Anwendung zentral bereitgestellt werden. Privilegierte Benutzer können dann zwar auf die öffentlichen Schlüssel der Benutzer sowie auf die verschlüsselten symmetrischen Schlüssel zugreifen, aber für deren Entschlüsselung ist der private Schlüssel eines Benutzers, für den die Datei freigegeben ist, nötig.

Problematisch an diesem Ansatz ist, dass Benutzern für dieses Verfahren ein Schlüssel zu einem asymmetrischen kryptografischen Verfahren zugeordnet werden muss. Dieser darf nur dem Benutzer bekannt sein und er darf insbesondere keinem zentralen System der Anwendung bekannt werden. Folglich muss eine externe, also nicht aus der Anwendung stammende, Information als Teil dieses Schlüssels verwendet werden, weil ansonsten über die zentralen Systeme der Anwendung eine Rekonstruktion des Schlüssels möglich wäre. Zu diesem Zweck bieten sich Informationen aus dem Authentisierungsprozess an. Je nach Authentisierungsmethode stehen unterschiedliche Informationen zur Verfügung.

Im Fall einer einfachen Benutzername-Passwort-Authentifizierung kann das Passwort als externe Information gesehen werden. Üblicherweise würde für diese Authentisierungsmethode das Passwort an den Server gesendet, um die Identität nachzuweisen. Damit das Passwort aber nicht auf den zentralen Systemen verfügbar wird, darf es nicht direkt an den Webserver versendet werden. Stattdessen können aus dem Passwort zwei separate Geheimnisse berechnet werden, von denen eins als Ersatz für das Passwort der

Authentisierung gegenüber der Anwendung dient. Das zweite Geheimnis kann dann als symmetrischer Schlüssel für eine vom Server bereitgestellte, verschlüsselte Datei verwendet werden, die den zu schützenden privaten Schlüssel des Benutzers enthält. Auf diese Weise muss der Benutzer lediglich, wie bei der Benutzername-Passwort-Authentifizierung üblich, ein Passwort eingeben, besitzt aber dennoch einen asymmetrischen privaten Schlüssel, der durch die zentrale Anwendung verwaltet wird, aber nicht eingesehen werden kann.

Im Fall eines externen IDP ist eine Unterstützung durch den IDP für das Verfahren erforderlich, weil für ein solches Verfahren nicht notwendigerweise ein Geheimnis liefert, dass dem Client aber nicht dem Server bekannt ist. In der Regel wird durch die Authentifizierung durch einen IDP dem Client und Server ein gemeinsames Geheimnis zur Verfügung gestellt, was aber nicht den oben beschriebenen Anforderungen genügt.
Unabhängig von der verwendeten Authentisierungsmethode kann stets ein privater Schlüssel vom Benutzer zur Verfügung gestellt werden, beispielsweise indem er eine entsprechende Datei auf seinem System ablegt und für die Webanwendung freigibt. Diese Lösung verringert aber die Benutzbarkeit der Anwendung erheblich, weil die Schlüsseldatei für die Verwendung der Anwendung notwendig ist und die vom Benutzer gewählte Authentisierungsmethode nicht mehr alleine für die Anwendung ausreicht. Tatsächlich wäre eine anderweitige Authentisierung in diesem Fall hinfällig, weil der geheime private Schlüssel auch zur Durchführung einer Authentisierung eingesetzt werden könnte.

Es lässt sich also zusammenfassen, sich das Verfahren für beliebige Authentisierungsmethoden umsetzen lässt, aber abhängig von dieser Methode Mehraufwand für den Benutzer bedeuten würde. Insbesondere wird der Einsatz eines IDP, der das Verfahren nicht unterstützt, de facto nutzlos. Um einen solchen IDP einsetzen zu können, müsste entweder der private Schlüssel vom Benutzer zusätzlich bereitgestellt werden oder die Datenverschlüsselung für solche Profile deaktiviert werden. Beide Optionen sind für einen Einsatz in der Praxis nicht tauglich. Ersteres könnte vom Benutzer für eine Mehrfaktorauthentifizierung gehalten werden, und würde auch den Aufwand einer Mehrfaktorauthentifizierung für den Benutzer mit sich bringen, wäre aber schwächer als eine echte Mehrfaktorauthentifizierung, weil für die Dateiverschlüsselung nur einer der Faktoren relevant ist. Die zweite Option, die Verschlüsselung für diese Profile zu deaktivieren, würde alle Vorteile zunichte machen, die der Einsatz des beschriebenen aufwendigen Dateiverschlüsselungsverfahren bringt. Das gelte zwar nur für Benutzer, die die betroffene Authentisierungsmethode verwenden, würde aber dazu führen, dass die Sicherheitsgarantien, die das System liefert, nicht intuitiv verständlich sind. Keine der beiden vorgenannten Lösungsansätze ist also praxistauglich. Gleichwohl kann einer dieser

Ansätze in einem Beispielprojekt Anwendung finden, das sowohl die Einbindung eines externen IDP als auch die Verwendung kryptografischer Verfahren demonstrieren soll.

Es ist außerdem zu berücksichtigen, dass privilegierte Benutzer auch eine Manipulation an der Webanwendung vornehmen können. Eine solche Manipulation könnte zur Offenlegung von kryptografischen Schlüsseln oder von Nutzdaten führen. Dieses Problem entspringt der Tatsache, dass die Logik der Webanwendung als Teil der Webseite mit ausgeliefert wird und der Benutzer diese nicht überprüft. Um dieses Problem zu lösen, wäre eine Integritätssicherung für die Webanwendung erforderlich. Diese müsste aber durch das HTTP selbst und nicht durch die Webanwendung implementiert werden, weil jeder in der Webanwendung integrierte Integritätsschutz selbst anfällig für eine Manipulation durch entsprechend privilegierte Benutzer wäre. Da ein entsprechender Mechanismus aber derzeit nicht existiert, ist dieses Problem im Kern nicht lösbar.

Ein zweiter Angriff, der durch entsprechend privilegierte Benutzer ausgeführt werden kann, ist die Übermittlung gefälschter öffentlicher Schlüssel für Freigabeziele. Vertraut der Benutzer einem solchen gefälschten öffentlichen Schlüssel und gibt die Datei über diesen Schlüssel frei, erhält der Besitzer des privaten Schlüssels Zugriff auf die Datei. Dieses Problem kann vermieden werden, indem der Benutzer beim erstmaligen Hinzufügen eines Arztes den entsprechenden öffentlichen Schlüssel überprüft. Bei der Wiederverwendung kann auf eine erneute manuelle Prüfung verzichtet werden, indem der Benutzer eine mit seinem privaten Schlüssel signierte Liste an vertrauenswürdigen öffentlichen Schlüsseln führt, die dann auch durch die zentrale Anwendung verwaltet werden kann. Allerdings muss sichergestellt sein, dass Einträge aus dieser Liste auch zuverlässig entfernt werden können.

Bei der Verwendung des beschriebenen Verfahrens ist zu berücksichtigen, dass je nachdem, wie der private Schlüssel des Benutzers geschützt wird, der Zugriff auf die Daten verloren gehen kann, wenn der Benutzer bestimmte Authentisierungsmerkmale verliert. Vergisst der Benutzer beispielsweise ein Passwort, aus dem ein Geheimnis zur Verschlüsslung des privaten Schlüssels abgeleitet ist, so kann zwar sein Passwort für den Zugriff auf die Anwendung zurückgesetzt werden, der Zugriff auf die Daten geht aber verloren. Dieses Problem kann reduziert werden, indem der Benutzer bei der erstmaligen Registrierung aufgefordert wird, eine Wiederherstellungsdatei abzulegen. Außerdem ist zu berücksichtigen, dass das Entziehen einer einmal erteilten Freigabe nur eingeschränkt möglich ist, weil derjenige, dem eine Datei zugänglich gemacht wurde, stets eine Kopie dieser Datei speichern kann.

Zuletzt ist anzumerken, dass das beschriebene Verfahren notwendigerweise und erwünschterweise dazu führt, dass die Webanwendung relevante kryptografische Schlüssel verwaltet und diese daher verwundbar für die Offenlegung durch entsprechende webbasierte Angriffe sind. Wird ein solcher Schlüssel kompromittiert, geht der Schutz vor bestimmten Angriffen durch privilegierte Benutzer beziehungsweise Angreifer, die vollständigen Systemzugriff auf die Anwendung erlange haben, verloren. Der eingangs beschriebene Schutz der Kommunikationswege bleibt jedoch trotzdem bestehen.

4.2 Konzept: Schützen von konfigurierten Geheimnissen produktiver Systeme

Für bestimmte Konfigurationsparameter des Systems besteht eine besonders hohe Vertraulichkeitsanforderung, weil durch ihre Offenlegung eine Manipulation oder Imitation des Systems möglich wäre. Das umfasst insbesondere die privaten Schlüssel zu Serverzertifikaten, die für TLS-Verbindungen eingesetzt werden, sowie gemeinsame Geheimnisse zwischen Systemen, die in diesem Kontext zur Authentifizierung von Teilen des Systems dienen, beispielsweise Passwörter zu Datenbankbenutzern, die von der Anwendung verwendet werden.

Die Hinterlegung solcher geheimen Konfigurationsparameter auf den Systemen ist unabdingbar, weil sie für den Betrieb erforderlich sind. Private Schlüssel zu Serverzertifikaten müssen zum TLS-Verbindungsaufbau vorliegen, um einen Identitätsnachweis durchführen zu können. Datenbankpasswörter müssen beim Verbinden mit der Datenbank vorliegen, wenn eine Passwortauthentifizierung durchgeführt wird.

Werden solche geheimen Konfigurationsparameter auf einem System gespeichert, stellt sich die Frage nach ihrer Absicherung gegen unerlaubte Einsichtnahme. Die Absicherung soll so erfolgen, dass die Vertraulichkeitsgarantien möglichst stark sind, während gleichzeitig die oben beschriebene Verfügbarkeitsbedingung erfüllt ist. Wenn geheime Konfigurationsparameter in einer Datei abgelegt werden, dann muss der Zugriff auf diese Datei möglichst weit eingeschränkt werden. Bei benutzerbasierter Zugriffssteuerung für Dateien bedeutet das, nur dem Benutzer, der die entsprechende Anwendungskomponente ausführt, den Zugriff einzuräumen. Auf diese Weise ist der Zugriff auf diesen Benutzer, sowie Benutzer mit der Berechtigung zum Überspringen von Berechtigungsprüfungen begrenzt. Eine stärkere Begrenzung ist durch das Dateisystem alleine nicht möglich. Ein Verschlüsseln der Konfigurationsdatei bringt keinen Mehrwert, außer maximal einer Security by Obscurity, weil die Anwendung in der Lage sein muss, wenn nötig auf die geheimen Konfigurationsparameter zuzugreifen und folglich auch etwaige zur Verschlüsselung verwendete Schlüssel kennen muss, diese also auch entsprechend zugreifbar hinterlegt sein müssen.

Geht man davon aus, dass die Anwendung selbständig, also ohne manuelle Interaktion, in der Lage ist, wenn nötig auf die geheimen Konfigurationsparameter zuzugreifen, dann folgt, dass jeder, der ein beliebiges Programm mit denselben Berechtigungen ausführen kann, auch auf die geheimen Konfigurationsparameter zugreifen kann, indem er dieselben Schritte ausführt wie die Anwendung selbst. Diese Voraussetzung ist für die meisten Betriebssysteme weitgehend deckungsgleich mit dem Zugriff auf eine wie oben beschrieben abgesicherte Datei. Dementsprechend wird das Ablegen der geheimen Konfigurationsparameter in einer solchen Datei als ausreichend gesichert angesehen. Die Verwendung von separaten Dateien für die geheimen und sonstigen Konfigurationsparameter ist wünschenswert, um Administratoren eine einfache und klare Unterscheidung zu ermöglichen.

Es bringt keinen Vorteil, die obere Annahme aufzuheben, dass die Anwendung selbständig in der Lage sein muss, die geheimen Konfigurationsparameter einzulesen. Wird für den Zugriff auf diese

Konfigurationsparameter erst eine manuelle Aktion erforderlich gemacht, erschwert dies zum einen den Betrieb der Anwendung offensichtlich massiv und zum anderen ist kein verbesserter Schutz zu erwarten, weil davon auszugehen ist, dass jeder, der eine wie oben beschrieben abgesicherte Datei einlesen kann, auch in der Lage ist, eine solche manuelle Interaktion abzufangen beziehungsweise die geheimen Konfigurationsparameter aus dem Hauptspeicher auszulesen.

Ein anderer Verbesserungsansatz ist der Einsatz spezieller Hardware, die das Auslesen von Inhalten unterbindet, aber gleichzeitig deren Verwendung erlaubt. Für gemeinsame Geheimnisse bringt dies keinen Vorteil, weil jede Verwendung eines gemeinsamen Geheimnisses dessen Kenntnis erfordert, ein solches also zwingend auslesbar sein muss. Andere Geheimnisse, insbesondere asymmetrische Schlüssel zu kryptographischen Verfahren, profitieren davon allerdings, weil zumindest ein unberechtigtes Kopieren der Schlüssel ausgeschlossen werden kann, wenn auch kein unberechtigtes Verwenden. Da aber entsprechende Hardware für den Einsatz im Projekt nicht verfügbar ist, wird auf ihren Einsatz verzichtet.

Ein Ansatz zur weiteren Regulierung des Zugriffs auf Dateien, die geheime Konfigurationsparameter enthalten, ist die Schaffung einer Berechtigungsstruktur, die es der Anwendung zunächst erlaubt, die geschützte Datei einzulesen, aber dann in einen weniger privilegierten Modus zu wechseln, sodass das nicht mehr möglich ist. So kann eine Anwendung benötigte Dateiinhalte in den Hauptspeicher laden, dann die Berechtigung zum Lesen der Datei fallen lassen und erst dann Funktionen zur Interaktion von außen aktivieren. Dieses Vorgehen bringt nur für vergleichsweise wenige Angriffsszenarien einen zusätzlichen Schutz, erzeugt aber einen im Verhältnis sehr hohen Aufwand, sodass auch diese Technik nicht eingesetzt wird.

Es ist zu berücksichtigen, dass alle oberen Überlegungen lediglich für das Ablegen von geheimen Konfigurationsparametern auf produktiven Systemen gelten, nicht aber für das Erstellen, Transferieren zwischen Systemen oder Vernichten derselben.

Außerdem können Verfahren eingesetzt werden, die zwar nicht den Vertraulichkeitsschutz von geheimen Konfigurationsparametern erhöhen, aber der Nutzen von diesen Informationen für einen Dritten reduziert werden, indem beispielsweise eine Datenbank, die passwortgeschützt ist, so konfiguriert wird, neben dem Passwort auch die Quelladresse des Zugriffs zu prüfen.

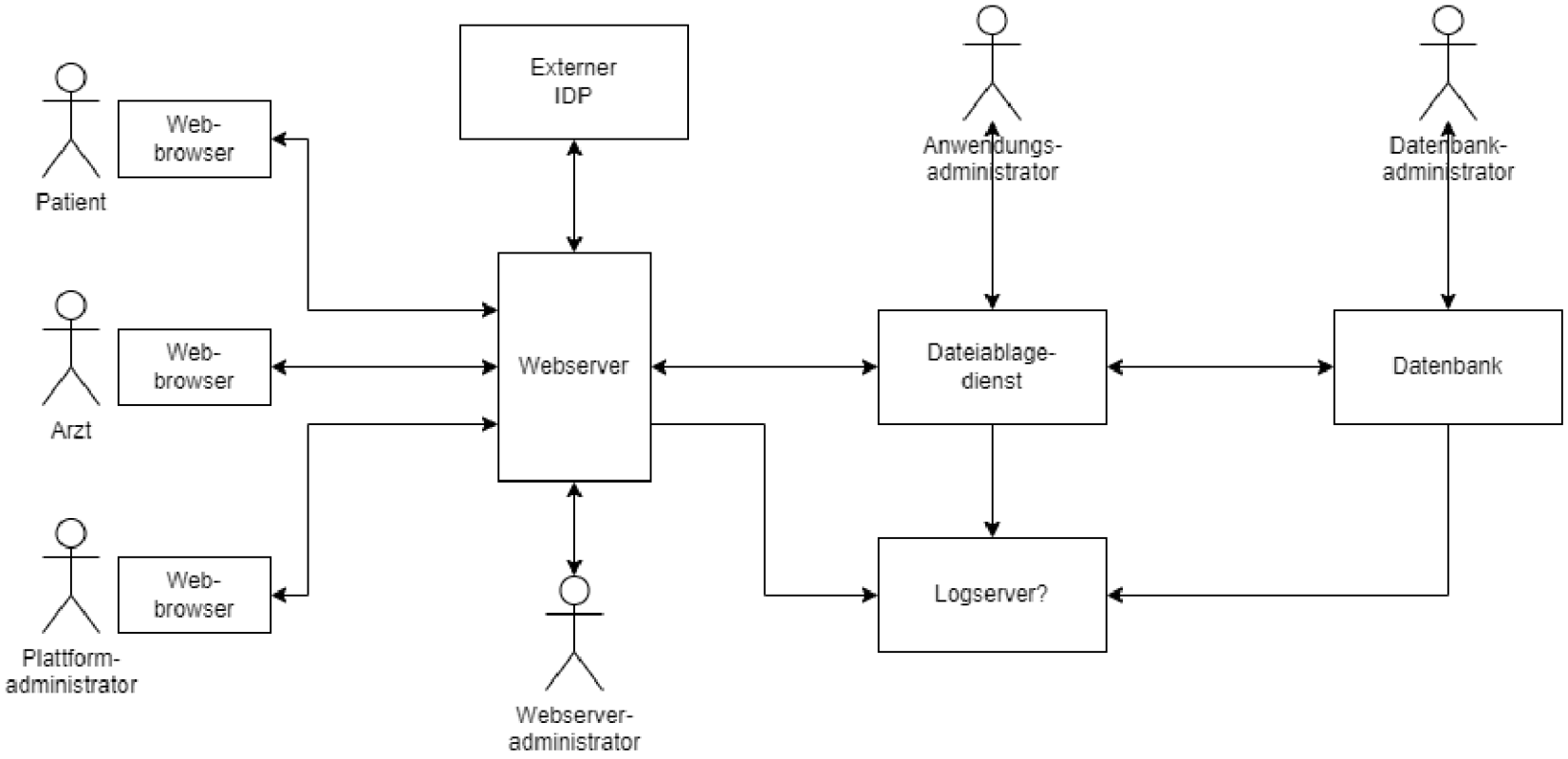
5 Anhang

5.1 Externe Inhalte

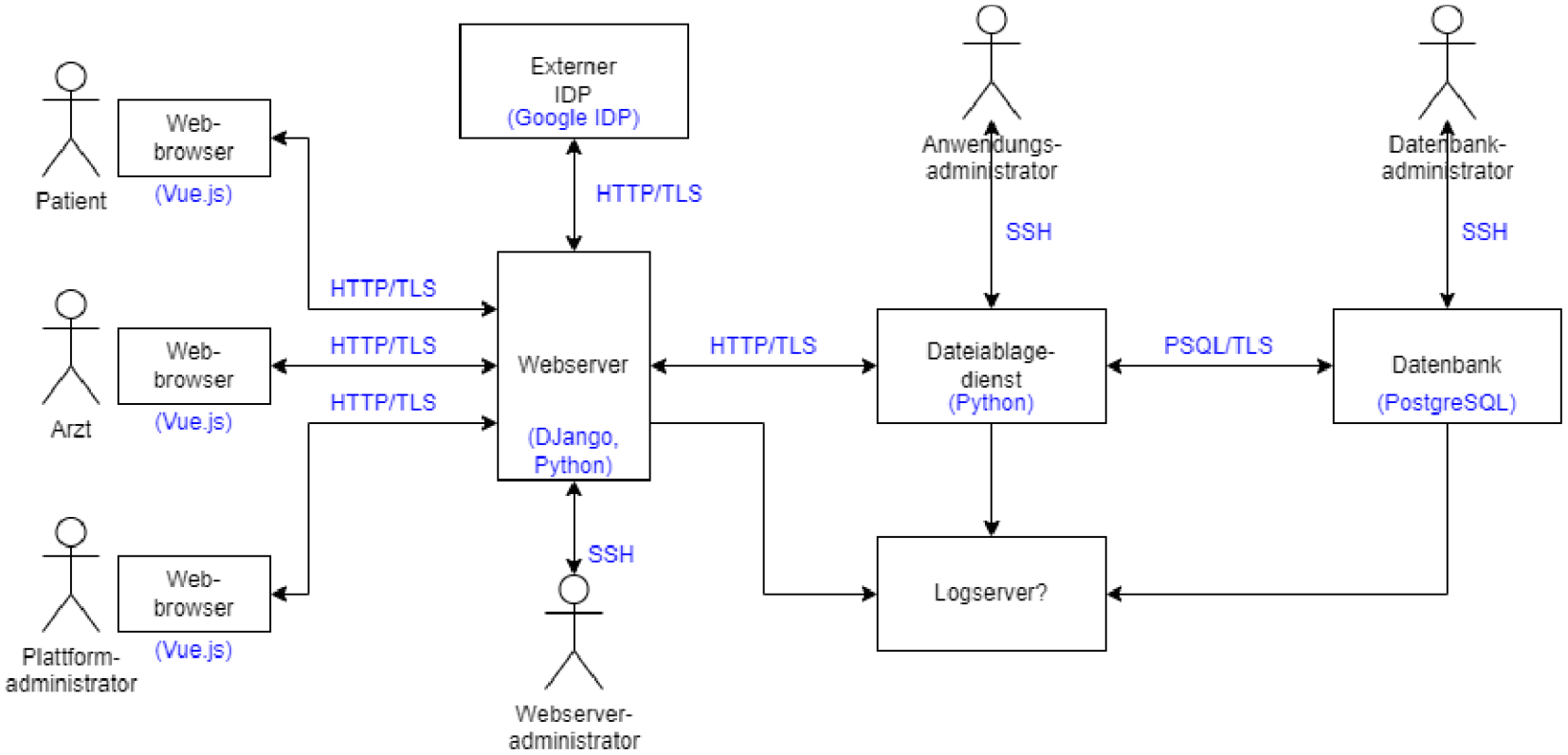
- Risikoregister: `Risikoregister.pdf`

5.2 Darstellungen

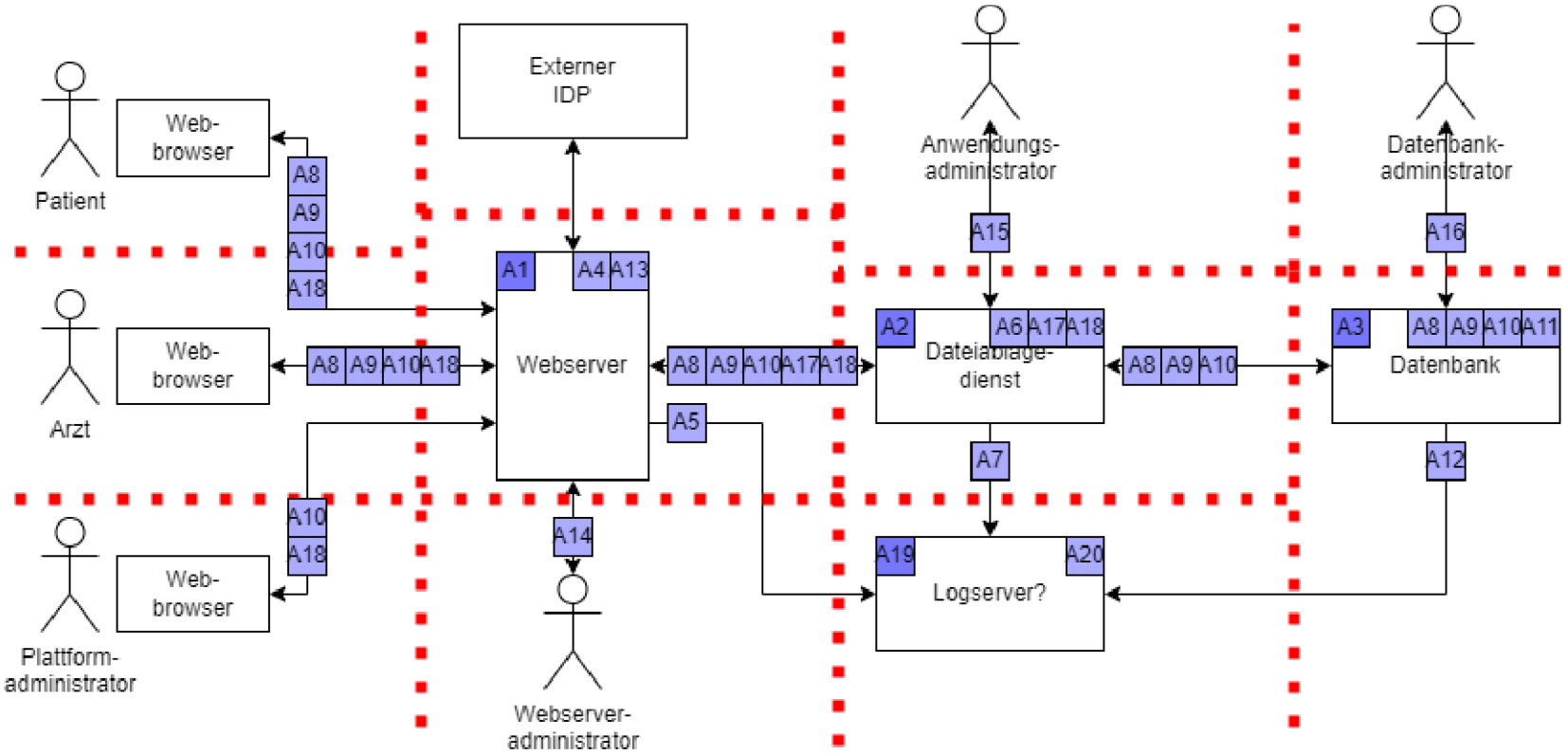
Darstellung 1: Architekturüberblick (abstrakt)



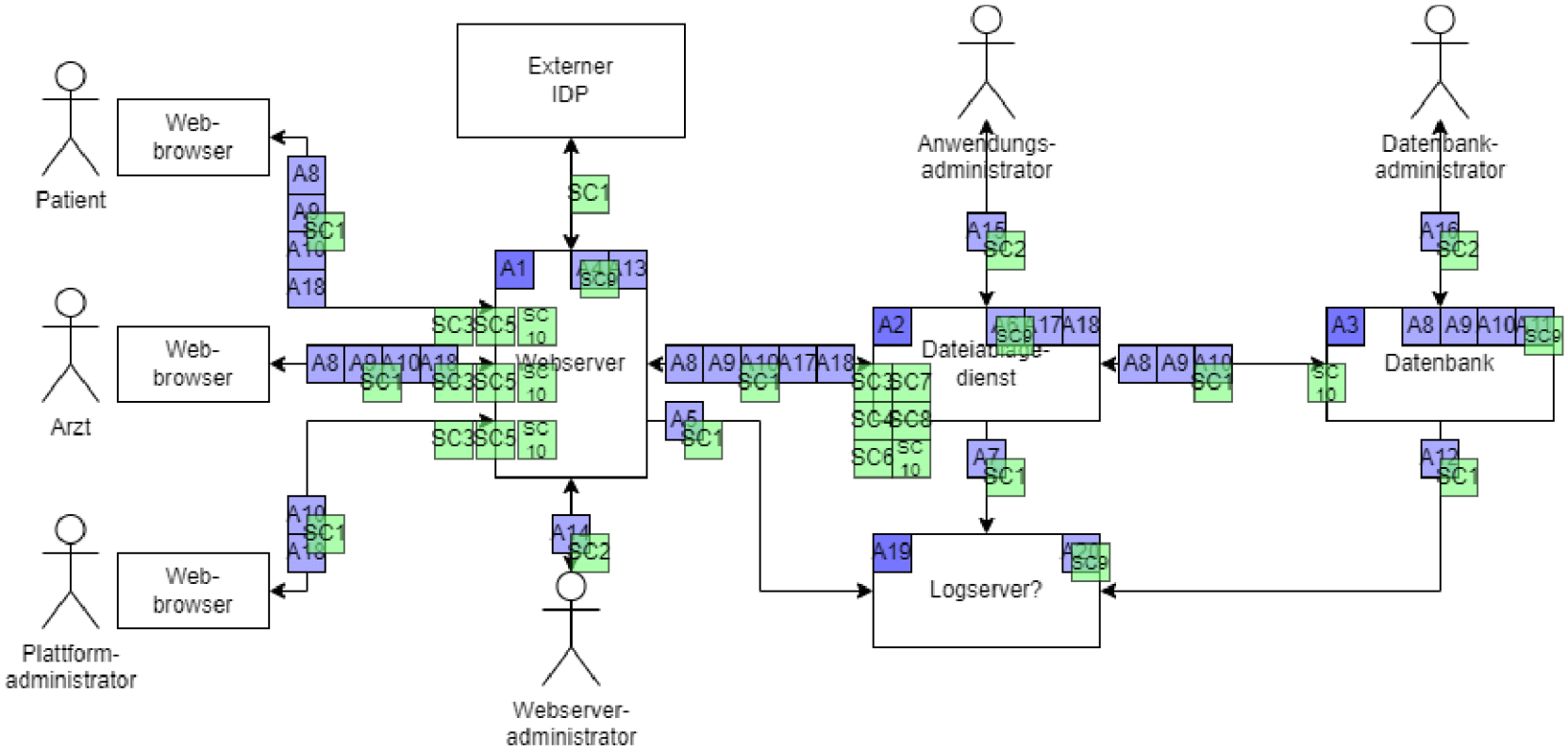
Darstellung 2: Architekturüberblick (technisch)



Darstellung 3: Architekturüberblick mit Schutzobjekten und Vertrauensgrenzen



Darstellung 4: Architekturüberblick mit Schutzobjekten und Sicherheitsmaßnahmen



Darstellung 5: Entscheidungsmatrix Architekturentscheidung Frontend

Kriterien	**Gewichtung**	**React**	**Evaluierung**	**Vue**	**Evaluierung**	**Angular**	**Evaluierung**
Kosten	Ausschluss		++		++		++
Vorhandene Expertise	3		--	keine Expertise, aber detaillierte Best-practise-Liste vorhanden	-	keine Expertise, aber detaillierte Best-practise-Liste vorhanden	-
Performance	2		+		+		O
Fähigkeit zur Entschlüsselung und Anzeige (vgl. Konzept: „Schützen von Data-at-rest“)	3	möglich, aber komplizierter	+		++		++
Benutzerfreundlichkeit	1		+		+		+
Entwicklungsaufwand	1		-		+		+
Resistenz gegen HTML-Injection	3	nur inoffizielle best-pratices	O	Standartmäßige Sicherheit	++	Standartmäßige Sicherheit	++
Resistenz gegen URL-Injection	3	nur inoffizielle best-pratices	O	Standartmäßige Sicherheit	++	Standartmäßige Sicherheit	++
Resistenz gegen JS-Injection	3	nur inoffizielle best-pratices	O	Standartmäßige Sicherheit	++	nur inoffizielle best-pratices	O
Gesamt			**-1 Pkt.**		**24 Pkt.**		**17 Pkt.**

Legende zur Evaluierung:

++: Die Alternative ist bezüglich dem Kriterium sehr positiv zu bewerten. Diese Bewertung entspricht 3 Punkten.
+: Die Alternative ist bezüglich dem Kriterium positiv zu bewerten. Diese Bewertung entspricht 1 Punkt.
O: Die Alternative ist bezüglich dem Kriterium neutral zu bewerten, eine Bewertung der Alternative hinsichtlich dieses Kriteriums ist nicht sinnvoll oder nicht praktikabel oder wurde aus einem anderen Grund nicht vorgenommen. Diese Bewertung entspricht 0 Punkten.
-: Die Alternative ist bezüglich dem Kriterium negativ zu bewerten. Diese Bewertung entspricht -1 Punkt.
--: Die Alternative ist bezüglich dem Kriterium sehr negativ zu bewerten. Diese Bewertung entspricht -3 Punkten. Die Gesamtbewertung ist die Summe der gewichteten Evaluierungen über alle Kriterien, die keine Ausschlusskriterien sind. Alternativen, die bezüglich mindestens einem Ausschlusskriterium negativ bewertet wurden, erhalten keine Gesamtbewertung und werden nicht weiter betrachtet.

Darstellung 6: Entscheidungsmatrix Architekturentscheidung Web Framework

Kriterien	**Gewichtung**	**Django**	**Evaluierung**	**Flask**	**Evaluierung**	**Bottle**	**Evaluierung**
Bibliotheken (z.B. AJAX, Authentifizierung, Authorisation, Caching, Datenvalidierung, etc.)	4		++	zahlreiche gängige Funktionen wie Authentifizierung, Session- und Cookies-Handling, Datenbankanbindung oder Caching sind über vorhandene Python-Libraries integrierbar – Entwickler haben die freie Wahl, welche Bibliotheken und Tools sie für ihre Webanwendungen einsetzen möchten	++		O
Codeausführungsumgebung	2		++	Unterstützung unabhängiger virtueller Entwicklungsumgebungen über das venv-Modul von Python	++		+
konsistente Entwicklungsumgebung	2		++	Unterstützung unabhängiger virtueller Entwicklungsumgebungen über das venv-Modul von Python	++		O
Verbindung zu verschiedenen Datenbanken	1		+		+		+
Logging, Recovery, Inter-System Messaging	3		++		+		O
Unterstützung von anpassbaren Konnektoren und Adaptern	2	durch Python	++	durch Python	++	durch Python	++

Kriterien	**Gewichtung**	**Django**	**Evaluierung**	**Flask**	**Evaluierung**	**Bottle**	**Evaluierung**
Unterstützung von unterschiedlichen Plattformen und Dateiformaten	2		O	Kompatibilität des Frameworks mit zahlreichen PaaS-Diensten in der Cloud	+		O
Datenmapping	1	durch Python	++	durch Python	++	durch Python	++
Automatisches Handling von Ausnahmen und Fehlern	2		O		O		O
Sicherheitsfunktionen	5		++		+		O
Verschlüsselungstools	5	durch Python	++	durch Python	++	durch Python	++
Online Performance Monitoring	1		++		++		+
Unterstützung von Routing Diensten	1		+		++		+
Komplexität	3		O		++		+
Gesamt			**77 Pkt.**		**72 Pkt.**		**32 Pkt.**

Legende zur Evaluierung:

- ++: Die Alternative ist bezüglich dem Kriterium sehr positiv zu bewerten. Diese Bewertung entspricht 3 Punkten.
- +: Die Alternative ist bezüglich dem Kriterium positiv zu bewerten. Diese Bewertung entspricht 1 Punkt.
- O: Die Alternative ist bezüglich dem Kriterium neutral zu bewerten, eine Bewertung der Alternative hinsichtlich dieses Kriteriums ist nicht sinnvoll oder nicht praktikabel oder wurde aus einem anderen Grund nicht vorgenommen. Diese Bewertung entspricht 0 Punkten.
- -: Die Alternative ist bezüglich dem Kriterium negativ zu bewerten. Diese Bewertung entspricht -1 Punkt.
- --: Die Alternative ist bezüglich dem Kriterium sehr negativ zu bewerten. Diese Bewertung entspricht -3 Punkten. Die Gesamtbewertung ist die Summe der gewichteten Evaluierungen über alle Kriterien, die keine Ausschlusskriterien sind. Alternativen, die bezüglich mindestens einem Ausschlusskriterium negativ bewertet wurden, erhalten keine Gesamtbewertung und werden nicht weiter betrachtet.

Darstellung 7: Entscheidungsmatrix Architekturentscheidung Programmiersprache

Kriterien	**Gewichtung**	**Python**	**Evaluierung**	**Java**	**Evaluierung**	**C++**	**Evaluierung**
Kosten	Ausschluss		++		++		++
Vorhandene Expertise	3		++		++		O
Performance	2		+		++		+
Memory Usage	2		+		O		++
Entwicklungsaufwand	3		++		+		-
Gesamt			**22 Pkt.**		**18 Pkt.**		**5 Pkt.**

Legende zur Evaluierung:

- ++: Die Alternative ist bezüglich dem Kriterium sehr positiv zu bewerten. Diese Bewertung entspricht 3 Punkten.
- +: Die Alternative ist bezüglich dem Kriterium positiv zu bewerten. Diese Bewertung entspricht 1 Punkt.
- O: Die Alternative ist bezüglich dem Kriterium neutral zu bewerten, eine Bewertung der Alternative hinsichtlich dieses Kriteriums ist nicht sinnvoll oder nicht praktikabel oder wurde aus einem anderen Grund nicht vorgenommen. Diese Bewertung entspricht 0 Punkten.
- -: Die Alternative ist bezüglich dem Kriterium negativ zu bewerten. Diese Bewertung entspricht -1 Punkt.
- --: Die Alternative ist bezüglich dem Kriterium sehr negativ zu bewerten. Diese Bewertung entspricht -3 Punkten. Die Gesamtbewertung ist die Summe der gewichteten Evaluierungen über alle Kriterien, die keine Ausschlusskriterien sind. Alternativen, die bezüglich mindestens einem Ausschlusskriterium negativ bewertet wurden, erhalten keine Gesamtbewertung und werden nicht weiter betrachtet.

Darstellung 8: Entscheidungsmatrix Architekturentscheidung Datenbank

Kriterien	**Gewichtung**	**MySQL**	**Evaluierung**	**PostgreSQL**	**Evaluierung**	**MongoDB**	**Evaluierung**
Kosten	Ausschluss		+		+		+
Vorhandene Expertise	5		+		++		O
Performance	2	vgl [1]	O	vgl [1]	++	vgl [1]	++
Entwicklungsaufwand	5	Python-Bibliothek vorhanden	+	Python-Bibliothek vorhanden + bekannt	++	Python-Bibliothek vorhanden	+
Struktur entspricht Daten	4		++		++		O
Unterstützung großer Binärdaten	4		-		O		O
Unterstützung paralleler Operationen	3		++		++		++
Gesamt			**27 Pkt.**		**60 Pkt.**		**20 Pkt.**

Kriterien	**Gewichtung**	**Apache CouchDB**	**Evaluierung**	**SQLite**	**Evaluierung**	**Microsoft SQL Server**	**Evaluierung**
Kosten	Ausschluss		+		+		-
Vorhandene Expertise	5		--		+		O
Performance	2	vgl [1]	++		-		O
Entwicklungsaufwand	5	Python-Bibliothek vorhanden	+	Python-Bibliothek + Expertise vorhanden	++		O
Struktur entspricht Daten	4		O		++		O
Unterstützung großer Binärdaten	4		-		--		O
Unterstützung paralleler Operationen	3		++		O		O
Gesamt			**1 Pkt.**		**21 Pkt.**		**n/a**

Kriterien	**Gewichtung**	**Oracle Datenbank**	**Evaluierung**
Kosten	Ausschluss		-
Vorhandene Expertise	5		O
Performance	2		O
Entwicklungsaufwand	5		O
Struktur entspricht Daten	4		O

Kriterien	**Gewichtung**	**Oracle Datenbank**	**Evaluierung**
Unterstützung großer Binärdaten	4		O
Unterstützung paralleler Operationen	3		O
Gesamt			**n/a**

Legende zur Evaluierung:

- ++: Die Alternative ist bezüglich dem Kriterium sehr positiv zu bewerten. Diese Bewertung entspricht 3 Punkten.
- +: Die Alternative ist bezüglich dem Kriterium positiv zu bewerten. Diese Bewertung entspricht 1 Punkt.
- O: Die Alternative ist bezüglich dem Kriterium neutral zu bewerten, eine Bewertung der Alternative hinsichtlich dieses Kriteriums ist nicht sinnvoll oder nicht praktikabel oder wurde aus einem anderen Grund nicht vorgenommen. Diese Bewertung entspricht 0 Punkten.
- -: Die Alternative ist bezüglich dem Kriterium negativ zu bewerten. Diese Bewertung entspricht -1 Punkt.
- --: Die Alternative ist bezüglich dem Kriterium sehr negativ zu bewerten. Diese Bewertung entspricht -3 Punkten. Die Gesamtbewertung ist die Summe der gewichteten Evaluierungen über alle Kriterien, die keine Ausschlusskriterien sind. Alternativen, die bezüglich mindestens einem Ausschlusskriterium negativ bewertet wurden, erhalten keine Gesamtbewertung und werden nicht weiter betrachtet.

Referenzen:

[1]: https://benchant.com/de/ranking/datenbank-ranking (Zuletzt abgerufen: 20.10.2022 19:00 Uhr)