IT-Architekturkompendium des BAMF - inkl. Coding Guidelines

Stand: 15.05.2020

Inhalt

Kontext und Zweck des Dokumentes

Um innerhalb der IT-Architektur die nachhaltige Entwicklung sowie die Konformität zur Geschäfts- und IT-Strategie des BAMF sowie zu den Architekturrichtlinien des Bundes und generell die Handlungs- und Steuerungsfähigkeit der IT zu bewahren und gleichzeitig Flexibilität und Innovation zu fördern, sind verbindliche Vorgaben und verlässliche Rahmenbedingungen unerlässlich.

Hierfür werden in den „Architekturprinzipien“ die zugrundeliegenden allgemeinen Regeln und Richtlinien für die IT-Architektur definiert. Dies stellt keinesfalls einen Ersatz für die Architekturrichtlinie des Bundes dar, sondern enthält Auszüge daraus und Ergänzungen des BAMF dazu. Den Architekturprinzipien werden jeweils auch „Designprinzipien“ zugeordnet als Richtlinien für die Gestaltung aller Entwicklungen innerhalb der IT-Architektur.

Das Architekturkompendium gilt als Grundlage für „Architektur-Prüfungen“. D.h., dass von Beginn eines IT-Projekts an durch die (i.d.R.) technische Projektleitung bzw. den/die verantwortliche(n) Software-Architekten/-in belegt werden soll, dass sich die Konzeption und die geplante Umsetzung des jeweiligen IT-Projekts konform zu den Grundsätzen verhalten. Die Einhaltung der Architektur- und Designprinzipen soll kontinuierlich gewährleistet und überprüfbar gehalten werden. Grundsätzlich sollen nur IT-Projekte gestartet werden, für die die Konformität gewährleistet werden kann. Wenn – in Ausnahmen – geplant ist, von Prinzipien abzuweichen, dann sind die Gründe und Lösungsansätze (ggf. Ausgleichsmechanismen und künftig geplante Anpassungen) darzustellen.

Die Beschreibungen der Architekturprinzipien werden um Definitionen und Beispiele ergänzt, die nicht auf alle IT-Projekte zutreffen. Es kann also bei Architektur-Prüfungen auch eine Konformität mit dem Architekturprinzip festgestellt werden, auch wenn nicht exakt dieselbe Definition/derselbe Anwendungsfall im IT-Projekt dargestellt werden kann. Hier ist die Konformitätsbegründung im einzelnen IT-Projekt entscheidend. Zudem gilt grundsätzlich, dass die bestehende gesamte IT-Architektur (künftig im EAM abgebildet) des BAMF zu berücksichtigen und die jeweils aktuell gegebenen Rahmenbedingungen in die Argumentation einzubeziehen sind.

In den ergänzenden Richtlinien zur Umsetzung der IT-Architektur- und Designprinzipien wird auf konkrete architekturkonforme Umsetzungsaufgaben eingegangen.

Die Coding Guidelines stellen die Regeln für die Softwarekomponenten-Architektur und die Programmierung der einzelnen Komponenten an sich dar.

Strategische Ziele der IT

Basierend auf den Fachaufgaben und den sich daraus ergebenden IT-Anforderungen wird das strategische Ziel einer zügigen, wirtschaftlichen, sicheren und nachhaltigen Umsetzung von neuen Verfahren und Verfahrensänderungen durch eine möglichst skalierbare IT-Unterstützung verfolgt. Dieses strategische Ziel lässt sich in verschiedenen Dimensionen beschreiben:

Nr.	IT-strategische Ziele
GZ01	Das Bundesamt als moderne und agile Behörde.
GZ02	Interne Prozessketten durchgängig digitalisiert.
GZ03	Die IT trägt zu einer hohen Qualität bei.
GZ04	Ressourcen werden effektiv eingesetzt.
GZ05	Die IT-Unterstützung ist sicher und funktioniert, auch in Krisensituationen.
GZ06	Die IT trägt zur Informationssicherheit, Datenschutz und Geheimschutz bei.
GZ07	Die IT wird an den Geschäftsprozessen der Behörde und den Schnittstellen zu externen Partnern ausgerichtet.

IT-Architekturziele

IT-Architekturziele sind die Grundlagen der IT-Architektur und Einflussfaktoren auf IT-Architekturentscheidungen — im Grunde sind es die wichtigsten architekturrelevanten Anforderungen.

Die Architekturziele lassen sich aus der Geschäfts- und IT-Strategie ableiten:

Nr.	Architekturziel	Kurzbeschreibung
AZ01	Konformität	Sicherstellung der Architekturkonformität aller IT-Vorhaben.
AZ02	Transparenz	Architektur-Framework und -Prozesse sind bekannt und verankert.
AZ03	Business/ IT Alignment	Ganzheitliche IT- und Geschäftsstrategieorientierung.
AZ04	Agilität/ Flexibilität	Agile Organisation und Vorgehensweisen. Verbesserte Flexibilität in der Architekturgestaltung und Weiterentwicklung der IT.
AZ05	Innovation	Innovations- und Gestaltungsfähigkeit.
AZ06	Sicherheit	Gewährleistung der Betriebs- und IT-Sicherheit, des Geheimschutzes und des Datenschutzes.
AZ07	Datenqualität	Gewährleistung der Datenqualität, Zugriff und Kommunikation.
AZ08	Wirtschaftlichkeit	Wirtschaftlichkeit der IT und Optimierung des IT Betriebs.
AZ09	Transformation	Gewährleistung der Fähigkeiten des IT-Architekturmanagement.
AZ10	Komplexitäts-Steuerung	Komplexität der IT-Landschaft effektiv und effizient zu verwalten.

AZ01 Konformität

Das Anforderungsmanagement und Projekt-Portfolio-Management sollen mit dem Enterprise Architecture Management kompatibel sein. Die Architekturkonformität aller IT-Vorhaben (zur Weiterentwicklung der IT-Landschaft) ist rechtzeitig zum Projektbeginn sicherzustellen.

AZ02 Transparenz

Die IT-Architektur-Organisation und -Steuerung sollen transparent, das Architektur-Rahmenkonzept und die -Prozesse bekannt und verankert sein. Dazu gehört auch die Umsetzung klarer Verantwortungs- und Entscheidungsrechte.

AZ03 Business/IT Alignment

Es herrscht eine ganzheitliche IT- und Geschäftsstrategieorientierung sowie ein ganzheitliches Geschäfts- und IT-Architekturmanagement, die zu einer umfassenden Betrachtung des Systems BAMF insgesamt führen. Dies unterstützt die Stabilität von Geschäftsprozessen sowie -funktionen und somit die

organisationsübergreifende Geschäftsfähigkeit. Gleichzeitig können Diskrepanzen und Probleme besser adressiert werden.

AZ04 Agilität/ Flexibilität

Die agile Organisation verfolgt die Ziele des effizienten und effektiven Beschäftigten- sowie Ressourceneinsatzes durch erhöhte und verbesserte Flexibilität in der Architekturgestaltung und Weiterentwicklung der IT. Dies bedeutet jedoch nicht ein erhöhtes Maß an Willkür oder Unsicherheit, sondern Handlungsspielraum innerhalb klar definierter Rahmenbedingungen.

AZ05 Innovation

Es sollen Rahmenbedingungen herrschen, in denen Innovation nicht nur möglich, sondern aktiv gefördert wird. Kooperative Modelle der Zusammenarbeit, Gestaltungsfähigkeit und Entwicklung sollen zukunftsfähige Lösungsarchitekturen begünstigen. Dazu gehören auch rollierende Planung, verfügbare Entwicklungsumgebungen und flexible Budgets, damit Innovationen schneller umgesetzt und vorangetrieben werden können.

AZ06 Sicherheit und Privacy by design

Neben dem Unfallschutz, Notfall- und Risikomanagement sind insbesondere die Betriebs-Sicherheit (inkl. Ausfallsicherheit), IT- und Informationssicherheit, der Geheimschutz sowie der Datenschutz zu gewährleisten

AZ07 Datenqualität

Daten sind in ausreichendem Maße gegen Verlust, Manipulationen und andere Bedrohungen zu sichern; die Qualität und Validität der Daten ist durch klare Datenarchitektur, systemtechnisch sowie durch die Sensibilisierung der Daten-erfassenden Beschäftigten zu sichern.

AZ08 Wirtschaftlichkeit

Im Rahmen der effizienten Steuerung der gesamten Organisation sind auch die Wirtschaftlichkeit der IT und die Optimierung des IT-Betriebs anzustreben. Die Kosten der Softwareentwicklung, Wartung und Support sollen auf das erforderliche Maß beschränkt sein. Grundsätzliches Ziel ist die Verbesserung der Kosten-/ Nutzensituation und die Risikominimierung zukünftiger Investitionen.

AZ09 Transformation

Um die Transformation hin zum Zielzustand und Innovationen generell umsetzen zu können, müssen entsprechende IT-Architekturmanagement-Fähigkeiten gegeben sein, mit denen die IT- und Informationssystemarchitektur flexibilisiert und gesteuert werden kann. Kostenstrukturen, Flexibilität, und Entwicklungsfähigkeit dürfen dadurch nicht leiden. Wenn Steuerungsstrukturen verändert werden, ist es beispielsweise wichtig, die Rollen und Verantwortlichkeiten für die angepassten Prozesse zu prüfen und sie ggf. ebenfalls anzupassen. Weiterhin sind Controlling-Systeme von Bedeutung, um Fortschritte der Transformation und ihre Zielerreichung zu messen.

EAM dient als Instrument zur Unterstützung und Führung von Organisationsveränderungen. Dazu sind sowohl die IT- als auch die Fachbereiche zu sensibilisieren.

AZ10 Komplexitätssteuerung

Die Komplexität der IT-Landschaft (Anzahl der Komponenten und Verbindungen) ist durch ein konsequentes Betriebsmodell und ein einziges EAM-Tool effektiv und effizient zu verwalten. Es dient der Übersichtlichkeit und einheitlichen Darstellung der IT-Landschaft, der Modularität und Integrationsfähigkeit. Dabei sollen die fachliche und die technische Betrachtungsweise miteinander verlinkt und ein Austausch zwischen verschiedenen Beteiligten (z. B. BAMF-Fachbereiche, BAMF-IT-Abteilung, ITZBund) effizient ermöglicht werden.

IT-Architekturprinzipien

Die IT-Architekturprinzipien definieren die zugrundeliegenden allgemeinen Regeln und Richtlinien für die IT-Architektur.

- Sie bieten eine solide Grundlage für die Erstellung von **IT-Architektur- und Planungsentscheidungen**, Rahmenbedingungen, Verfahren und Standards sowie die Lösung von widersprüchlichen Situationen.
- Sie werden ausgewählt, um die **Ausrichtung der IT-Architektur** an den Geschäftsstrategien und die **Umsetzung der Zielarchitektur** zu gewährleisten. Jedes Architekturprinzip steht in Beziehung zu den jeweiligen Geschäfts- und Architekturzielen.
- IT-Architekturprinzipien müssen **klar nachvollziehbar und klar artikuliert** werden, um Entscheidungsfindung zu leiten. Sie müssen verständlich, robust, wenige in der Zahl, aber möglichst vollständig, konsequent einzuhalten und dauerhaft/ stabil definiert sein.
- Sie sollen **zukunftsorientiert** sein sowie **von der Leitung unterstützt** und verteidigt werden.

Den IT-Architekturprinzipien werden jeweils die Designprinzipien zugeordnet.

Art des Prinzips	Nr.	IT-Architekturprinzipien
Strategische Prinzipien *Business Principles*	AP01	**Einhaltung von Architektur- und Dokumentationsrichtlinien** Einhaltung von Architektur Richtlinien, Prinzipien und Standards des BAMF und des Bundes, sowie einheitliche Dokumentationssprachen für Übersichtlichkeit und Steuerbarkeit
	AP02	**Security by Design** Ganzheitliche Sicherheitskonzeption bzw. Datenschutz- und Geheimschutzkonzeption, Geschäftskontinuität
	AP03	**Durchgängige Digitalisierung** Durchgängige Digitalisierung von Prozessketten
	AP04	**Benutzerfreundliche Anwendbarkeit** Sicherstellung von Ergonomie, benutzerfreundlicher Anwendbarkeit und Kommunikation sowie Barrierefreiheit
	AP05	**Medienbruchfreie Kommunikation** Selbstverständnis einer Dienstleistungsbehörde mit medienbruchfreier Kommunikation ggü. Externen, „Service-Provider“: Verfügbarkeit von BAMF-Daten und -Diensten für externe Nutzer (Fachverfahren)

Daten orientierte Prinzipien *Data Principles*	AP06	**Global gültige Datenstrukturen** Global gültige Datenstrukturen (inkl. Stammdaten) sind übertragbar, hinzukommende Datenstrukturen sind global zu definieren und einzuhalten
	AP07	**Durchgängige Datensicherheit** Daten sind gegen Verlust, Manipulationen und andere Bedrohungen zu sichern
Anwendungs-orientierte Prinzipien *Application Principles*	AP08	**Skalierbarkeit** Die Anwendungslandschaft und die Nutzbarkeit einzelner Anwendungen in ihrer Quantität ist skalierbar (horizontal) sowie die einzelnen Anwendungen in ihrer Funktionalitätstiefe skalierbar sind (vertikal)
	AP09	**Agilität fördernde Architektur** Agile Abstimmung der beteiligten Teams, Anwendungen, Infrastruktur und des Betriebs aufeinander
	AP10	**Wiederverwendbarkeit und Homogenität** Bereitstellung und Einsatz wiederverwendbarer Software und Komponenten, Homogenität der IT-Landschaft
	AP11	**Modularisierung und lose Kopplung** Softwarekonstruktion aus syntaktisch identifizierbaren, für die Realisierung der heutigen und künftigen Systemfunktionen geeigneten Bausteinen
	AP12	**Platform-as-a-Service und Software-as-a-Service** Zentrale Bereitstellung und Nutzung von Plattformen und Software als Dienste
Technologisch-orientierte Prinzipien *Technology Principles*	AP13	**Interoperabilität** Interoperabilitätsstandards müssen grundsätzlich angewendet werden, um eine einfache Integration unterschiedlicher Anwendungen und Technologien zu ermöglichen
	AP14	**Kontrolle technischer Vielfalt** Kontrolle technischer Vielfalt und begründeter Einsatz neuer inkl. innovativer Technologien

Designprinzipien

Die Designprinzipien skizzieren die Richtlinien für Entwicklungen innerhalb der Service-orientierten IT-Architektur. Erst durch die umfassende Verwendung aller IT-Architektur- und Designprinzipien entsteht eine den Anforderungen ganzheitlich angemessene IT-Architektur.

Nr.	Designprinzipien	Übergeordnete Prinzipien	Assoziierte Prinzipien
DP01	**Lose Kopplung** Lose Kopplung bedeutet, dass Services voneinander isoliert sind und nur über Schnittstellen (üblicherweise per REST) miteinander kommunizieren. Eine in-Memory-Kommunikation oder im Code hinterlegte Verknüpfungen zu anderen Services sind unbedingt zu vermeiden. Services bilden keine Workflows ab und haben daher auch keine Kenntnis über andere Services (mit Ausnahme deren Schnittstelle) oder den Systemkontext. Dieser Ansatz erlaubt den relativ einfachen Austausch von Services, solange die Schnittstelle gewahrt bleibt.	AP06, AP08, AP11	DP02, DP03, DP04, DP05, DP11
DP02	**Separation of Concerns** Bedingt durch die Microservice-Architektur, ist bei der Implementierung von Services auf den Grundsatz "Separation of Concerns" zu achten, welcher das Prinzip "Single Purpose" einschließt. D.h. jeder Service hat genau eine Aufgabe bzw. Zweck. Jeder Service stellt nur einen kleinen Teil der Gesamtanwendung bereit, wie z.B. eine einzelne UI-Komponente (nicht aber das gesamte UI), Daten lesen oder Daten speichern. Außerdem ist darauf zu achten, dass fachliche und technische Aspekte voneinander getrennt sind.	AP03, AP10, AP11	DP09

DP03	**Dependency Inversion** Die Umkehr der Abhängigkeiten ist ein Prinzip der Objektorientierten Programmierung, formuliert von Robert C. Martin: "Erlaube Abhängigkeiten von Abstraktionen, nicht von Spezialisierungen." Das zu Grunde liegende Konstruktionsprinzip ist hier "Teile und Herrsche": Module auf höheren Ebenen enthalten Module niedrigerer Ebenen und delegieren einen Teil der Arbeit an diese. Zwischen höherwertigen Modulen und niederwertigen Modulen liegt eine Abstraktionsschicht, an die diese Delegation erfolgt. Das höherwertige Modul kennt nur die Abstraktion (Interface) seiner Dienstmodule und nicht das Dienstmodul selbst. Dadurch entsteht ein hierarchisch aufgebautes System, Implementierungsdetails werden entkoppelt und Details letztendlich austauschbar. Bei der Entwicklung technischer Services ist dieses Prinzip der „Dependency Inversion“ anzuwenden, denn es ermöglicht eine Domain-zentrierte IT-Architektur.	AP10, AP11	DP11

DP04	**Event-Driven Kommunikation** Grundlegend muss zwischen Event und Nachricht unterschieden werden. Die Kommunikation über Nachrichten erfolgt asynchron über den NAM der Middleware (siehe auch DP06), ist dediziert und die Zustellung zum Empfänger ist garantiert. Im Gegensatz dazu sind Events flüchtig, können mehrfach vorkommen und die Zustellung ist nicht garantiert. Im Einzelnen heißt das, dass verpasste Events, z.B. durch eine Nichtverfügbarkeit eines konsumierenden Services, nicht nochmals zugestellt werden. Ein Event kann über die Eventplattform auch an mehrere Empfänger gesendet werden (Subscriber). Die "Event-getriebene" Kommunikation erfolgt per „Publish/Subscribe“ Pattern. Dieses Pattern in einer ereignisgesteuerten Architektur ermöglicht ein asynchrones, dynamisches n:n Messaging und entkoppelt somit Absender und Empfänger. Die Entkopplung ist sinnvoll, da der Empfänger nur Events empfängt, wenn er ein bestimmtes Topic subskribiert hat. Der Empfänger kann jederzeit ein Topic subskribieren bzw. beenden. Der Absender stellt Informationen per Events auf der Eventplattform bereit, ohne zu wissen, wer die Informationen konsumiert. Jeder Empfänger verarbeitet diese Events auf eine andere Art und Weise und unabhängig von anderen Services, aber alle arbeiten mit den gleichen Daten.	AP03, AP11, AP12	DP06

DP05	**Datenstrukturzentrische Anwendungen bzw. Services** Bei der Entwicklung von Services und elektronischen Fachprozessen ist auf eine informationszentrierte Kommunikation zu achten. Bei der Kommunikation zwischen Services dürfen nur strukturierte Daten und keine Dokumente/ Dateien (Ausnahme Dokumentendienste) ausgetauscht werden. Nur damit sind die in der Digitalisierungsagenda gesteckten Ziele erreichbar, wird die maschinelle Weiterverarbeitung erheblich vereinfacht und die Infrastruktur aufgrund geringerer Datenmengen entlastet. Datenstrukturen und zugehörige APIs sind laut "JSON LD"-Prinzip selbstbeschreibend und müssen sich auf den Standard XÖV bzw. die BAMF Salvador-Definition beziehen. Falls Dateien/ Dokumente (z.B. PDF) zur Erhaltung der Rechtssicherheit bzw. des Beweiswerts benötigt werden, so können diese parallel zu den (Meta-)Informationen erstellt und an einem geeigneten Ort (z.B. eArchiv, DZA, etc.) abgelegt werden. Hausintern werden nur Referenzen auf diese Dateien zwischen einzelnen Services ausgetauscht. Bei der Kommunikation mit externen Partnern sind diese Referenzen erst unmittelbar vor dem Versand aufzulösen.	AP03, AP05, AP06, AP11	DP12
DP06	**Asynchrone Schnittstellen** Prozessketten sind primär asynchron zu designen, was technisch zu einem nachrichtenbasierten Service-Design führt.	AP03,AP08, AP10, AP11, AP12, AP14	DP08

DP07	**Fehlermanagement & Konsistenzsicherung** Zur Konsistenzsicherung müssen Services und Prozesse auf Fehler adäquat reagieren, protokollieren und eine Kompensationslogik (Recovery Mechanismus) zur Verfügung stellen. Unter Recovery werden sämtliche Präventiv-Prinzipien und -Techniken subsumiert, welche die Zuverlässigkeit der Services und Prozesse erhöhen. Zu den wesentlichen Qualitätsmerkmalen der Recovery-Mechanismen zählen: • das Recovery soll möglichst vollständig sein • muss möglichst schnell geschehen und die Maßnahmen sollten nicht unnötig aufwendig sein, d.h. Größe des Schadens und der Aufwand für das Recovery sollen in einem vernünftigen Verhältnis zueinander stehen • die Recovery-Maßnahmen sollen sich nur auf die beschädigten Teile auswirken. Insbesondere soll transaktionsbezogenes Recovery möglich sein. • unter Verlust von möglichst wenig bereits geleisteter Arbeit • die Recovery-Maßnahmen sollten je nach Bedarf automatisch durchgeführt werden • die Kosten des Recoverys, Qualität und Quantität des Aufwands, sollen möglichst gering sein.	AP03, AP05, AP11	DP08
DP08	**API- und servicezentriertes Schnittstellendesign** Jede Anwendung stellt ihren Funktionsumfang über eine top-down konzipierte Serviceschnittstelle (API) bereit und ist unabhängig vom User-Interface betreibbar. Aspekte, die beim Design berücksichtigt werden müssen, sind: • Abwärtskompatibilität • Versionierung • Vorrangig ressourcenorientiertes Design (REST und HATEOAS-Prinzip) • Semantische Verlinkung mit "Salvador"-Strukturdefinitionen • Schnittstellen-Dokumentation • Fehlerbehandlung gemäß DP07 und Coding Guidelines	AP11, AP13, AP14	DP18

DP09	**Atomares Servicedesign und Idempotenz** Die interne Implementierung eines Services ist atomar und verbirgt sein internes Verhalten (Kapselung). Die Außenschnittstelle ist möglichst idempotent aufrufbar. Ein Microservice muss in der Lage sein, eine Nachricht/ eine Operation mehrfach zu verarbeiten, ohne dass sich der Zustand des Services/ das Ergebnis nochmals ändert, sobald es bereits einmal erfolgreich ausgeführt/ verarbeitet wurde. D.h., der Service muss vor der Aktion prüfen, ob diese bereits durchgeführt wurde, und eine entsprechende (Log-)Meldung ausgeben. Die Service-Schnittstelle muss Doppelaufrufe akzeptieren und darf keine Fehlersituation sowie keine Fehlermeldung generieren.	AP11	DP02
DP10	**End2End Protokollierung und Monitoring (technisches Logging)** Ziel ist eine zyklische Qualitätssicherung der Services und Serviceketten durch eine konsistente und einheitliche Überwachung im Betrieb sowie auch im Support. Alle relevanten Daten und Services müssen in einem Log protokolliert werden, welches eindeutig, nachvollziehbar und plausibel sein muss. Für jede sich ändernde Aktion enthält der Log einen Eintrag über folgende Angaben: • Identifikation der Transaktion • Identifikation des Objekts • Art der Aktion • ggf. Angaben wie Datum, Uhrzeit, Identifikation betroffener Seiten, oder Verweis auf einen vorgehenden Logeintrag • zusätzlich gibt es Einträge für Beginn, Commit und Rücksetzung von Transaktionen und weitere interne Zwecke • jeder Logeintrag erhält eine fortlaufende Nummer	AP02, AP03, AP11, AP14	
DP11	**Rahmenbedingungen für Agilität und durchgängige Digitalisierung** Durchgängige domänenübergreifende Digitalisierung erfordert bei Wahrung der Agilität Maßnahmen in den Bereichen Security, Datenstrukturen, Schnittstellendesign, Servicedesign, Orchestrierung, Konsistenzsicherung, Fehlerbehandlung.	AP02, AP05, AP06, AP07, AP11	DP05-09, DP12

DP12	**Semantisch definierte Datenstrukturen** An Schnittstellen werden semantisch definierte und übergreifend gültige Daten-Typen und Strukturen verwendet.	AP03, AP05, AP06, AP14	DP05
DP13	**Stammdaten und Wertelisten** Stammdaten sind beschreibende und identifizierende Daten (kein Personenbezug), die stets unverändert bleiben. Den Stammdaten zugehörig sind Wertelisten (Codelisten), z.B. Liste aller Staaten. Alle Objektinstanzen erhalten eine eindeutige und übergreifend gültige ID sowie Versionsnummer. Diese Stammdatenstruktur, aber auch die zugehörigen Codelisten werden semantisch an einer zentralen Stelle als Metadaten dokumentiert in einem zentralen Repository. Dieses Repository („Salvador“) befindet sich im BAMF aktuell im Aufbau, auf der Grundlage des XRepository 3.0. Das Datengrid ist für das Daten-Management ein wesentlicher Bestandteil in dieser Architektur. Stammdaten werden im Datengrid anderen Diensten und Applikationen per Service bereitgestellt. Der Zugriff auf die Stammdaten erfolgt ausschließlich lesend.	AP03, AP05, AP06, AP14	DP12
DP14	**Eigenständige Lebenszyklusverwaltung** Komponenten und Services müssen unabhängig konfiguriert, deployed, betrieben und getestet werden können und weisen ein eigenes Lifecyclemanagement mit vorgegebenem Versionierungsschema auf.	AP09, AP10, AP11	DP02, DP08, DP09
DP15	**Nachvollziehbarkeit (fachliches Logging)** Weshalb, durch wen und in welchem Kontext eine Komponente oder Service genutzt wurde, muss nachvollziehbar sein, weshalb die übergreifenden Mechanismen, wie übergreifendes Auditing, Security-Propagation, Betriebslogging, etc. anzuwenden sind.	AP02, AP03	DP 10

DP16	**Authentifizierung** Die Authentifizierung erfolgt anwendungsneutral über das Access-Management (OAM), gekoppelt mit dem zentralen Identitätsmanagement (IDM). Über IDM werden die Rollen, von denen die Rechte eines Benutzers abgeleitet werden, an die Anwendung bzw. den Service übergeben. Das Durchsetzen der Rechte über diese Rollenzuordnung ist Service- und Anwendungs-lokal zu lösen. Die Anmeldeinformation (dn bzw. IDM-Kürzel) wird durch alle Services propagiert (u.a. für Nachvollziehbarkeit notwendig).	AP02, AP03, AP07	
DP17	**Datensicherheit** Service-Kommunikation erfolgt stets über einen sicheren (verschlüsselten) Kanal. Die Datenablage erfolgt bei Bedarf verschlüsselt. Bei der Protokollierung sind Sicherheitstokens und Attribute einer hohen Schutzklasse auszufiltern.	AP02, AP07	DP10, DP16
DP18	**Schnittstellendokumentation** Schnittstellen müssen die Aspekte Input, Output und Fehlerverhalten dokumentieren und sie müssen in einem einheitlichen Format publiziert werden (bei REST: OpenAPI ab Version 3). Alle Datenstrukturen mit externer Sichtbarkeit (CSV, REST, DB, SOAP, gRPC, usw.) werden in einem einheitlichen Format übergreifend nutzbar beschrieben. Die Standardvorgaben von BAMF Salvador sind einzuhalten. Die Aspekte Input, Output und Fehlerverhalten einer Schnittstelle müssen mit den definierten Datenstrukturen verlinkt sein (semantisches Mapping).	AP01, AP10	DP08
DP19	**Projektdokumentation** Die nachhaltige Dokumentation jeder Komponente auf den Ebenen Anforderung, EAM, Entwicklung (Design), Deployment, Betrieb und Nutzung ist sicherzustellen. Als Dokumentationsrepositories werden BAMF Standards etabliert und verwendet. Genau nachzulesen unter: • Konzeptcheckliste • IT-Architekturkompendium des BAMF • BAMF IT - Coding Guidelines	AP01	

DP20	**Einheitliche Oberflächengestaltung** Nutzer-Oberflächen ("User Interfaces") sind gemäß der Vorgaben des einheitlichen BAMF Styleguides und Design Systems zu erstellen. Weitere Informationen über diese Regelwerke befinden sich im Confluence-Auftritt des Bereichs "UI-S": User Interface Standardisierung (UI-S) Übergreifend gültige Regelweke, wie das Corporate Design der Bundesregierung, sind im BAMF Styleguide berücksichtigt. Zudem sind die Sicherheitsrichtlinien (u.a. die speziellen Vorgaben für SPA) einzuhalten.	AP04	
DP21	**Barrierefreiheit** Das GUI-Design folgt von Beginn an den gesetzlichen Vorgaben. Der ungehinderte Zugang zum System muss gewährleistet werden, um es entsprechend der technischen, persönlichen und situationsbedingten Möglichkeiten nutzen zu können. • Bedienelemente sollen tabulierbar sein. • Inhalte müssen skalierbar sein. • Plattformunabhängigkeit bzgl. Endgerät und Display, inkl. Unterstützung einer Braillezeile. • Inhalte sollen hierarchisch angeordnet sein, um Usern den direkten Zugang zur gewünschten Information zu ermöglichen. • Grafiken sollen mit Alternativtexten versehen werden. Die Richtlinien für barrierefreie Webinhalte (WCAG) 2.0 decken einen großen Bereich von Empfehlungen ab, um Webinhalte barrierefrei zu machen. Auf den Web Content Accessibility Guidelines fußt auch die Barrierefreie-Informationstechnik-Verordnung (BITV) 2.0. Die unbedingt einzuhaltenden Kriterien (Checkliste) sind hier zu finden: Barrierefreiheit-Tests	AP04	

DP22	**Design to Test** Testgetriebene Entwicklung ist ein wesentlicher Bestandteil der agilen Softwareentwicklung und wird in diesem Zusammenhang auch als Test First-Ansatz bezeichnet. Von Anfang an soll die Testbarkeit des Systems sichergestellt, aber auch eine vollständige und automatisierte Überprüfung aller Komponenten erreicht werden. Dadurch können Regressionsfehler bereits früh erkannt und mit geringem Aufwand behoben werden. Testbarkeit ist in allen Projektphasen elementares Strukturierungs- und Designkriterium, d.h. in Analyse, Design, Implementierung und Integration. Hierbei entsteht ein sog. TDD-Zyklus. Bei dieser testgetriebenen Designstrategie wird der Test noch vor der eigentlichen Komponente geschrieben, d.h. die Schnittstelle der zu testenden Komponente wird bereits benutzt, bevor diese existiert. Dadurch ist ein erstes Feedback darüber sichergestellt, ob das Design auch anwendbar ist. Die Implementierung des produktiven Codes erfolgt erst, wenn ein Test vorliegt, der dies verlangt. Für das Schreiben eines bestimmten Tests kann es notwendig sein, zuerst ein Refactoring an anderen Tests oder Produktivcode vorwegzunehmen. Jede Aktivität beim testgetriebenen Entwickeln lässt sich einem Abschnitt zuordnen. Insgesamt arbeitet die Softwareentwicklung testgetrieben und ist für Unit- und autarke Komponententests verantwortlich, wobei diese die wesentlichen Prinzipien einer kontinuierlichen Designverbesserung sowie Test-First beinhaltet.	AP04, AP10	

DP23	**Betreibbarkeit** Der Fokus jeder Komponente liegt auf einfacher Betreibbarkeit. Es werden die BAMF-übergreifend gültigen Mechanismen für Logging, Property-Management (Konfigurations-Server und Service Registry) bzw. Konfiguration, Metrics/Diagnostic (metrics-lib, check_mk, zipkin, Prometheus) angewandt. • Automatische Installierbarkeit: Automatisierte Installation, um den geringen Aufwand und die exakte Reproduzierbarkeit für die Installation einer Umgebung sicherzustellen. • Konfiguration wird separat verwaltet und erlaubt eine unabhängige Bearbeitung durch das Betriebsteam. • Externe Infrastrukturänderungen müssen sich auf das technische Verhalten auswirken, dürfen aber nicht das fachliche Verhalten eines Dienstes beeinflussen. • Übertragbarkeit in andere Umgebungen (Dev, Test, Abn, Prod): Dasselbe Artefakt muss auf allen Umgebungen unverändert installierbar sein. Eine externe Konfigurationsquelle stellt die einzige Quelle von Unterschieden zwischen den Umgebungen dar. • Austauschbarkeit: Durch vorher definierte und dokumentierte Schnittstellen kann die Software jederzeit durch eine andere Software ersetzt werden. • Analysierbarkeit (Logging) → Logging/Tracing: Dies erfordert das Nutzen der bereitgestellten Methoden zum Protokollieren, Metriken sammeln, um damit das Fehlverhalten und Betriebszustände zu jedem Zeitpunkt zu diagnostizieren. • Nachvollziehbarkeit (Tracing) → Logging/Tracing: Der Programmablauf muss vom initialen Request in der gesamten Aufrufkette durch die Administration nachvollziehbar sein.	AP14	DP10

Ergänzende Richtlinien zur Umsetzung der IT-Architektur- und Designprinzipien

Architekturvorgabe „Nutzung der sog. Middleware Dienste“

Die BAMF PaaS Middleware (PaaS = Platform as a Service) erleichtert die architekturkonforme Implementierung von skalierbaren cloudfähigen Software-Systemen unter Berücksichtigung der Bundescloud-Vorgaben. Sie stellt einen Baukasten für die effiziente Entwicklung architekturkonformer Anwendungen dar. Dieser Baukasten enthält eine Reihe konfigurierter und eigen entwickelter Komponenten, welche bei der Entwicklung von BAMF-Anwendungen einzusetzen sind.

Zentrale Middleware-Dienste

Authentifizierung (oAuth2/OpenID Connect) und Autorisierung

Das Architekturprinzip "Security By Design" verlangt die Sicherstellung der Integrität, Vertraulichkeit, Verfügbarkeit und Authentizität der Informationen. Mit oAuth2 Authentifizierung und Autorisierung wird sichergestellt, dass es sich beim Zugriff auf Informationen tatsächlich um eine autorisierte Person (Identitätsnachweis) oder um ein berechtigtes System handelt. Die Integrität, Vertraulichkeit, Verfügbarkeit und Authentizität der Informationen muss zudem mit weiteren Mitteln wie Verschlüsselung, Signatur, HA, usw. umgesetzt werden.

Microservices, Client-Server- und Web-Anwendungen müssen sich mittels oAuth2-Protokoll an einem zentralen Authentifizierungsserver u.a. per REST-Endpoint authentifizieren und erhalten daraufhin die in der IDM-Infrastruktur des BAMF hinterlegten Rollen-Informationen, die zur Autorisierung der Nutzer herangezogen werden können. Services und Systeme müssen sich separat mit speziell ausgestellten Client-Tokens authentifizieren. Sowohl Anwender als auch Systeme erhalten über den Authentifizierungsserver des BAMF IDM-Systems nach der Anmeldung ein zeitlich begrenztes Security-Token ausgestellt, das im Standard-Format JWT über die sichere TLS-Verbindung zwischen allen Service/Anwendungen ausgetauscht wird.

Dokumentendienste

Für die Verwaltung von Dateien und Dokumenten stehen zwei zentrale Dienste bereit, die abgesichert u.a. auch über REST-Schnittstellen angesprochen werden können:

- **Datei-Zwischenablage (DZA)** ermöglicht die sichere, hochverfügbare und performante Zwischenspeicherung von Dateien beliebigen Formats, die sich im Entstehungs- und Bearbeitungsprozess befinden. Die DZA soll das unnötige Versenden von Dokumenten sowie die damit verbundene Netzbelastung und den ungeschützten Zugriff vermeiden. Als Zugriffsschnittstelle wird eine abgesicherte REST-API bereitgestellt, damit der Zugriff von beliebigen Systemen programmiersprachenunabhängig ermöglicht wird. Für JAVA Systeme wird zusätzlich eine Client-Bibliothek bereitgestellt, die eine einfache Nutzung aller Funktionen erlaubt.
- **Dokumenten-Dienste (DD)** stellen verschiedene Services mit REST-API zur sicheren Ablage und Verwaltung von Dokumenten im elektronischen Archiv (eArchiv) und Langzeitarchiv (LZA) zur

Verfügung. Dies sind i.d.R. Dokumente, die einen fachlich, verbindlichen Stand erreicht haben, Teil einer Akte sind und Format-Restriktionen genügen müssen (PDF/A).

Nachrichtendienste

Der Nachrichtendienst (NAM) stellt umfangreiche Funktionen für nachrichtenbasierte, asynchron arbeitende Anwendungen bereit und stellt die garantierte Auslieferung von Nachrichten sicher. **Der Nachrichtendienst MARiS Adapter (NAM-Adapter)** stellt REST Services per Adapter bereit, mit denen eine einfache Serviceanbindung an das bestehende Fachverfahren MARiS und Übergabe von Dokumenten in MARiS-Akten ermöglicht wird.

Bei der Verwendung der NAM Plattform gilt grundsätzlich:

- Nachrichten dürfen nur als application/json asynchron übertragen werden.
- Dateien und Dokumente dürfen nur als Referenz zur DZA/Dokumentendienste übergeben werden.
- Die URL für die asynchrone Rückantwort darf nur als Dienstname aus der Service Registry per Header übergeben werden.
- Die Auslieferung einer Nachricht wird garantiert, wobei der Empfänger mit einer Mehrfachauslieferung aufgrund der cloud-native Anforderung umgehen können muss.

Eventgrid

Die BAMF Architekturprinzipien fordern eine lose gekoppelte Anwendungslandschaft - die Gefahr einer Sternintegration durch die vielen Microservices soll damit vermieden werden - was u.a. auch eine asynchrone Kommunikationsinfrastruktur verlangt.

Anwendungen und Microservices müssen in die Lage versetzt werden ihre fachlichen Zustandsänderungen an die Event Plattform zu melden, damit diese die parallel betriebenen Anwendungen informiert und diese adäquat mit ihren implementierten Aktionen darauf reagieren können. Das Eventgrid arbeitet mit dem Standard https://cloudevents.io/ und erlaubt damit einen cloud- und produktagnostischen Einsatz von Events mit dem langfristigen Ziel eine hybride Multicloud-Strategie zu unterstützen.

Datengrid

Zur Verbesserung der Datenqualität und Unterstützung der Architekturprinzipien stellt das **Datengrid (DGR) Konzept** einen serviceorientierten Zugriff auf Stammdaten sowie Datenperspektiven aus Fachanwendungsdatenbanken bereit. Stammdaten dürfen im Normalfall somit nicht mehr per Daten-Schnittstellen dupliziert und in den lokalen Fachdatenbanken erneut verwaltet werden, sondern müssen über die abgesicherten Datengrid-Services bezogen werden. Die Stammdatenstrukturdefinition bzw. Wertelisten inkl. semantischer Beschreibung und die nachgelagerte Belieferung ins Datengrid erfolgt aus dem zentralen Salvador-Repository.

Fachagnostische Services (FPaaS)

Tresorservice (TRS)

Anwendungen, Anwender und Dienste müssen ihre Geheimnisse (Secrets) wie Passwörter, Zertifikate, Private-Keys, etc. sicher und verschlüsselt speichern und die Zugriffsrechte auf diese Daten verwalten können. Zudem müssen Zugriffe auf Geheimnisse auditfähig und Zugriffsrechte müssen auch wieder entziehbar sein.

Zugriffstoken haben eine Gültigkeit und können in ihrer Nutzungsanzahl beschränkt werden. Ein Anwendungsfall ist die Erzeugung eines Zugrifftokens auf ein Dokument per Dokumentservice, bei dem der Zugriff für einen definierten Zeitraum oder die Anzahl der Zugriffe eingeschränkt wird. Mit Ablauf des Token oder zu häufiger Nutzung wird automatisch der Zugriff entzogen, womit in Einzelfällen ein komplexes Rechtemanagement und Verwaltung der Zugriffsrechte überflüssig wird.

Der Tresordienst (TRS) wird als Basisdienst der BAMF PaaS Middleware sowohl per REST API als auch als Client-Library angeboten.

Maskierungsservice (MAS)

Für Anwendungen, die IDs oder andere Inhalte maskieren (unkenntlich machen) müssen, was z.B. beim technischen Logging von personenbezogene Daten notwendig ist, wird der Maskierungsservice (MAS) bereitgestellt.

Der MAS stellt primär einen Algorithmus zur sicheren Maskierung von Werten und IDs. Durch das Zusammenschalten dieser mit zusätzlichen Strings, auch SALT genannt, und Anwenden von sicheren Hash- Algorithmen sind Werte nicht mehr zu den ursprünglichen Werten zurückzuführen. Der Dienst erzeugt Hash-Werte auf unterschiedliche Weise, je nach gewünschtem Algorithmus und unter Angabe von benutzerspezifischen SALT oder MAS-SALT (MAS-Systemeigener SALT). Es können einzelne Werte (String), rohe Daten (Byte-Array), oder Wertelisten (String-Array) in einem Aufruf verarbeitet werden. Zur sicheren Ablage der SALTs soll der Tresor zum Einsatz kommen.

Zudem wird der Service genutzt, um Passwörter sicher gemäß OWASP Standard (Open Web Application Security Project) zu hashen, damit nur der Hash-Wert, aber nicht das Passwort in „plaintext“ abgelegt wird.

ID-Service

Der ID Service ermöglicht die Generierung von eindeutigen IDs, die verwendet werden können, um Daten im gesamten BAMF System zu identifizieren und nachzuverfolgen. Solche eindeutigen IDs können beispielsweise für die eindeutige Logdatenkennzeichnung oder auch als Schlüsselwerte für Datenbanktabellen dienen, bevor die Daten in die Tabelle eingefügt werden.

Die Verwendung des ID Services ermöglicht die Trennung der ID-Generierung von der Verwendung, was bei Datenbank-Sequenzen nicht der Fall ist. Dadurch wird gewährleistet, dass durch Änderungen an der zugrundeliegenden Datenbanktechnologie bzw. dem Speicherort die Art oder Identität der Daten nicht beeinflusst wird.

ID-Mapper

Bestandssysteme, die vor Einführung des zentralen Datenstruktur- und Stammdaten-Managements implementiert wurden, sowie gekaufte Softwareprodukte, verwenden oft lokal generierte Stammdaten

IDs, die systemübergreifend nicht eindeutig sind und sich damit zu den ID's im später geschaffenen zentralen (XÖV-) Standard unterscheiden. Wenn eine Datenmigration aus Konsistenzgründen nicht immer möglich ist, bedarf es eines Mapping-Dienstes. Dieser ID Mapping Service ist fachagnostisch und erlaubt eine Übersetzung der lokalen IDs zu den übergreifenden ID's.

Als Beispiel könnte in einem Altsystem das Land Italien mit der ID=5 referenziert sein, extern wird jedoch zum Datenaustausch der XÖV-Standard verwendet und damit muss die ID=5 zur ID=137 übersetzt werden.

IP-Cat

Neben der Standard-Netzsicherheit des ITZBund führt BAMF laut Schalenmodell auf Anwendungsebene eine weitere Zugriffskontrolle bzw. Protokollierung in Abhängigkeit des Zugriffswegs (z.B. Mobilarbeitsplätze) ein. Anfragen aus definierten Netzsegmenten werden erkannt und der Zugriff auf ausgewählte Anwendungen verweigert. Beispielsweise können mobile SINA Rechner nur ausgewählte Anwendungen oder Funktionen verwenden, wohingegen die ans BAMF LAN angeschlossen SINA Rechner dieselben Anwendungen mit identischen Anwenderkennung in vollem Umfang nutzen können. Der IP-Cat Service liefert auf unterster Ebene den Sicherheitsgates und den Anwendungen diese Information zur weiteren Verarbeitung.

Logging-Service

Der Logging-Service stellt einen zentralen Dienst zur Ablage und Auswertung von technischen Logdaten bereit. Der Service kann von Systemadministratoren mit gesonderten Berechtigungen genutzt werden, um Fehler zu analysieren und Optimierung im technischen Ablauf vorzunehmen. Um ein technisch, über die Kette aller eingebunden Komponenten durchgängiges Tracing zu erhalten, wird der OpenTracing Standard (https://opentracing.io/) in Verbindung mit Standardframeworks wie z.B. Log4J, ELK eingesetzt. Das Instrumentieren und damit das Durchschleusen der dafür notwendigen IDs erfolgt transparent durch Einbinden von separaten Hilfsmitteln und Frameworks wie z.B. Spring Sleuth für Java-Anwendungen.

Konfigurationsserver

Der Konfigurations-Server erlaubt die anwendungsexterne Verwaltung von umgebungsspezifischen Einstellungen. Er bietet einen Schutz vor unberechtigter Manipulation, Änderungen lassen sich nachvollziehen und er bietet Verschlüsselung von Parametern soweit notwendig. Technologisch wird u.a. ein separates GIT-Repository als Unterbau verwendet, weshalb die Anwendungsparameter wie bisher dateibasiert (Property- und YAML-Files), jedoch separat zum Deploymentartefakt dem Konfigurationserver bei der Installation übergeben werden müssen.

Spring Framework Service

BAMF abstrahiert die verwendeten Technologien und die Umgebungsabhängigkeiten durch Einsatz des Spring Frameworks und der Spring Dienste. Spring Dienste wie z.B. der Konfigurationservice-Client KoS, Authentifizierungs-Client oAuth2-Client, werden mit dem Entwurfsmuster Dependency Injection in die Anwendungen eingewoben und damit können sie anwendungsübergreifend im BAMF identisch behandelt werden. Die Entwicklung des funktionalen Codes ist damit unabhängig von seiner Umgebung geschrieben. Dies erhöht neben der Qualität auch die Entwicklungseffizienz. Im Spring Framework mit u.a. Spring Boot und Spring Cloud wird eine Vielzahl von Sub-Framework und Runtimekomponenten genutzt.

Cloud-Architektur - Cloud Native

Im BAMF gilt eine Anwendung als Cloud-Native, wenn diese für die Cloud-Computing-Architektur, unter Anwendung der Architekturvorgaben, wie z.B. Microservice- und API-driven-Design, unter konsequenter Nutzung der umgebenden Cloud-Infrastruktur-Dienste konzipiert und umgesetzt ist.

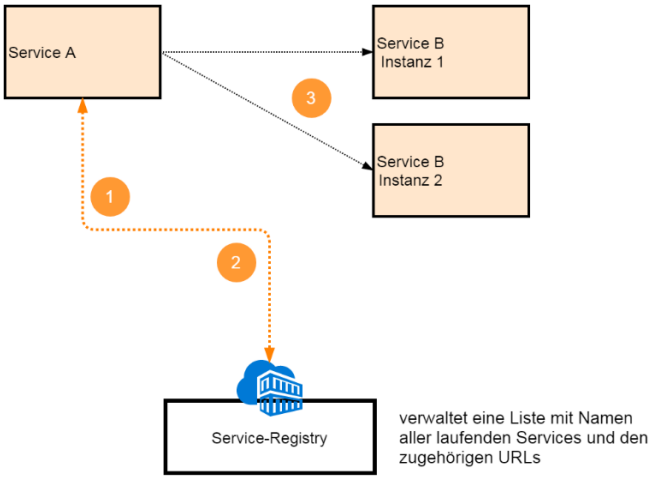
Die geschaffene Microservice-Anwendung und die dafür genutzten Produkte und Services lassen sich in einem herstellerunabhängigem Cloud-Ökosystem kohärent verwalten und betreiben. Querschnittsaspekte wie Administration, Logging, Monitoring, Deployment, Skalierbarkeit, Security, usw. werden aktiv von der Cloud-Plattform über einheitliche Schnittstellen genutzt. Microservices implementieren diese Querschnittsaufgaben nicht selbst, sondern vertrauen auf die externe Bereitstellung und delegieren diese dort hin. Zudem darf ein Microservices kein Ressourcen-Management (CPU, RAM, HD, MEM) betreiben, sondern muss davon ausgehen, dass die Cloud-Umgebung ausreichend Ressourcen bereitstellt bzw., skaliert und diese bei Nichtnutzung wieder freigeben. Die Auswahl der möglichen Cloud Services und Tools richtet sich an den Empfehlungen der Coud Native Compute Foundation (CNCF) aus.

Service Discovery

In einer dynamischen Umgebung (VMWare Cluster, Bundescloud, usw.) müssen Services aus Gründen der Ausfallsicherheit und Skalierbarkeit mehrfach auf unterschiedlichen Knoten einer Umgebung gestartet werden können. Zur Absicherung kurzfristig auftretender Lastspitzen muss dies ggf. auch kurzfristig erfolgen können.

Aus diesem Grund ist eine statische Konfiguration von Services zu URLs in Anwendungen nicht sinnvoll. Vielmehr wird eine dynamische Konfiguration benötigt, die es allen Services ermöglicht, andere Services zur Laufzeit zu finden.

Prinzip des Service-Discovery Services: Mit einer Service-Registry werden Microservices unter einem logischen Namen registriert, anschließend lassen sie sich über diesen Namen ansprechen. Damit können Services dynamisch auf unterschiedlichen Instanzen laufen, ohne dass ein aufrufender Service die tatsächliche URL des Services wissen muss.



- Service A hat selbst kein Wissen, wo Service B läuft. Der Aufruf des Service B erfolgt daher wie folgt:
    1. Service A fragt bei Service-Registry nach, wo Service B zu finden ist.
    2. Die Registry gibt i.d.R. eine Liste mit Adressen aller laufenden Instanzen des Service B zurück.
    3. Service A ruft gemäß Loadbalancing-Regeln eine der Instanzen des Service B auf.

- Startet ein Service neu, ruft er zuerst die Service-Registry auf und registriert sich, damit er von anderen Services gefunden werden kann.

- Ein Service benötigt daher lediglich die Information unter welcher Adresse die Service-Registry der Umgebung (Entwicklung, Test, Staging oder Produktion) läuft.

Architekturvorgabe IDM/IAM-Anbindung

Die Architektur- und Designprinzipien des BAMF umfassen die Vorgabe, dass die im BAMF eingesetzten Systeme an das zentrale IDM/IAM (Identity and Access Management) des BAMF angebunden werden müssen.

Das BAMF-IDM/IAM ist ein (abstraktes) System, das für alle an BAMF-Geschäftsprozessen beteiligten Personen

- **Autorisierung** und
- **Authentifizierung**

durch geeignete Schnittstellen unterstützt. Für durch BAMF direkt oder indirekt verwaltete Identitäten unterstützt es zusätzlich durch geeignete Oberflächen die

- **zentrale und delegierte Administration von Benutzerdaten** sowie
- **Selfservice-Operationen** (z.B. Passwortänderung).

Andere Identitäten werden über eine **Federation** aus anderen vertrauenswürdigen Quellen übernommen. Ebenso stehen per **Federation** BAMF-Identitäten in übergreifend genutzten Produkten zur Verfügung.

Dies dient insbesondere dem Architekturprinzip "AP02 Security-by-Design".

Alle Softwareprodukte, die BAMF-Geschäftsprozesse unterstützen, müssen sich daher in das BAMF-IDM/IAM (bzw. ein per **Federation** Assoziiertes) integrieren und **sollen** Benutzer-Credentials nicht selbst verarbeiten, insbesondere kein eigenes Login bereitstellen.

Konkret legt das Designprinzip "DP16 Authentifizierung“ fest:

Die Authentifizierung erfolgt anwendungsneutral über das Identitätsmanagement (IdM). Das Durchsetzen der (Zugriffs-) Rechte ist Service- und Anwendungs-lokal zu lösen. Die Anmeldeinformation wird durch alle Services propagiert (u.a. für Nachvollziehbarkeit notwendig).

Das BAMF-IDM/IAM umfasst die Benutzerverwaltung (beinhaltet die Verwaltung der User und Rollen, Protokollierung sowie Provisionierung) und das Access Management (Zugangskontrolle und Login).

Im Accessmanagement soll der Standard oAuth2 mit JWT Tokens genutzt werden. Der OpenID Connect Standard erweitert oAuth2 und stellt die Brücke zur Benutzerverwaltung (Infos zum User und dessen Profildaten) dar.

Zweck und Ziele, die mit der IDM-Anbindung erreicht werden:

- Die IDM-Anbindung unterstützt die Architekturziele des BAMF und erfüllt die Architektur- und Designprinzipien.
- Einheitliche Handhabung aller IT-Applikationen hinsichtlich Benutzerverwaltung, IDM und IAM
- Benutzerfreundlichkeit: ein einziges Passwort für alle an IDM angebundenen Fachanwendungen, „Single Sign On“
- Vereinheitlichung der Benutzerverwaltung -> es sind keine separaten Benutzerdatenbanken zu pflegen
- Einfachere Möglichkeit der Anbindung an Drittsoftware (per Vertrauensstellung oder gemeinsamer Autorisierungsbasis)
- Zentrale Durchsetzung von abgestimmten Security-Policies
- Zentrale Verwaltung und Auditierung der Zugriffsrechte
- Durch die Zentralisierung und die Standardisierung (IT-Architektur) wird Handlungssicherheit und Steuerungsfähigkeit gewonnen
- Bedingt durch die Einführung von feingranularen eigenständigen Microservices ist eine durchgängige Sicherheit per standardisiertem Token notwendig
- Umsetzung der vom Informationssicherheitsbeauftragten (ISB) geforderten Sicherheitsprotokollierung für die Nachvollziehbarkeit bei Sicherheitsvorfällen
- Möglichkeit, neue Login-Mechanismen einfach zu etablieren, wie z.B. SmartCard, Zwei Faktoren Authentifizierung, usw.
- Möglichkeit, Identitäten mittels Federation aus anderen vertrauenswürdigen Quellen zu übernehmen.

Zentrale Stammdatenverwaltung

Gemäß Architekturprinzip **AP 06** (Global gültige Datenstrukturen: Global gültige Datenstrukturen sind übertragbar, hinzukommende Datenstrukturen sind global zu definieren) und auf Basis der Designprinzipien **DP 12** (Semantisch definierte Datenstrukturen: An Schnittstellen werden semantisch definierte und übergreifend gültige Daten-Typen und Strukturen verwendet.) und **DP 13** (Stammdaten und Wertelisten: Stammdaten und Wertelisten sind übergreifend und eindeutig zu definieren. Sie werden im Datengrid anderen Diensten und Applikationen per Service bereitgestellt und sind daraus zu beziehen.) sind die IT-Anwendungen im BAMF verpflichtet, den zentralen Stammdatenservice „Salvador" zu nutzen.

Als Stammdaten werden hierbei die vergleichsweise statischen Datenstrukturinformationen und Code-/Wertelisten bezeichnet; wohingegen Nutzdaten bzw. personenbezogene Daten nicht zu den hier gemeinten Stammdaten zählen. Stammdaten können von mehreren Anwendungen verwendet werden und sind im Kontext analytischer Auswertungen oft die Kriterien, nach denen ausgewertet wird. Veränderungen werden über eine Stammdatenhistorie aufgezeichnet und vorgehalten.

In der bislang aufgebauten Softwarelandschaft – in den diversen IT-Fachanwendungen – des BAMF wird eine Vielzahl von Datentypen und Wertelisten verwendet. Diese werden aktuell im Rahmen einer Stammdaten-Konsolidierung vereinfacht (Beseitigung von Redundanzen) und schrittweise, soweit möglich vereinheitlicht (Beseitigung unnötiger Heterogenität). Die zentrale Stammdatenverwaltung „Salvador" wird dabei das Instrument, durch das die definierten grundlegenden Datentypen und Wertelisten vorgehalten und dokumentiert werden. „Salvador" wird auf der Basis der Software der OSS-Webapplikation XRepository 3.0 der KoSIT bis ins Jahr 2020 entwickelt und soll folgende Anforderungen erfüllen:

- In der zentralen Stammdatenverwaltung sollen Datenstrukturinformationen, Datentypen und Wertelisten sowie ggf. weitere Arten von Komponenten validiert, gespeichert, visualisiert und zugreifbar gemacht werden
- Das System soll die Dokumentation von Ableitungsbeziehungen (Datentypen auf Anwendungsebene werden von Kernkomponenten oder von Kerndatentypen abgeleitet) gestatten. Es soll nachvollziehbar machen, welche Komponente (Werteliste) in welchen Kontexten (Datentypen) zum Einsatz kommt.
- Die Stammdatenverwaltung soll auf XÖV-Technik basieren, d.h. die Artefakte werden als UML-Modelle editiert und durch die Stammdatenverwaltung eingelesen (Austauschformat XMI für UML-Klassenmodelle). Auch die weiteren Komponenten der XÖV-Methodik (z.B. XÖV-UML-Profil zur Auszeichnung von Klassen zur Gestaltung von Datentypen und Wertelisten) sollen eingesetzt werden wie im XÖV-Rahmen definiert (vgl. XÖV-Handbuch in der aktuellen Version 2.1).

Daraus ergibt sich folgender Zielzustand:

- In der Datenübermittlung werden Stammdaten und Codelisten eingesetzt, um die für einen bestimmten Übermittlungskontext relevanten Sachverhalte eindeutig zu bezeichnen und in einem einheitlichen Format zu übermitteln.
- Die Stammdatenstruktur und Codelisten werden semantisch an einer zentralen Stelle als Metadaten dokumentiert (zentrales Repository).
- Jedes Stammdatum wird an nur einer zentralen Stelle verwaltet à SSOT (syn.: Single Source of Truth = einzige Quelle der Wahrheit/der gültigen Stammdatendefinition). Sie bilden eine Bibliothek

(Repository), deren Daten und ID‘s von allen lokalen Software-Instanzen eingebunden und genutzt werden können.

- Alle Objektinstanzen in den SSOT erhalten eindeutige und übergreifend gültige ID und Versionsnummern.
- Es existiert ein konsistenter Datenbestand und die Stammdaten sind nicht redundant abgelegt. Vorhandene Datensilos sind - soweit möglich - in einem Repository aufgelöst. Sofern eine IT-Fachanwendung systemtechnisch bedingt (noch) nicht umgestellt werden kann, sind die Stammdaten einer solchen Anwendung im Repository repliziert und erhalten neben den internen und nur lokal gültigen ID’s eine übergreifend gültige ID. An den replizierten Orten werden die Stammdaten nur als „readonly“ Werte gekennzeichnet und demzufolge dort nicht modifiziert.
- Wertelisten (Codelisten), wie z. B. Staaten, Bundesländer, sind stets im zentralen Repository zu pflegen und über zentrale Datengrid-Services abzurufen. Alle abgelegten und ausgetauschten Datenobjekte werden semantisch gekennzeichnet und mit der semantischen Beschreibung im Repository verlinkt.
- Ausgetauschte Daten und Attribute per API-Services werden mit einer Verlinkung zur semantischen Beschreibung kennzeichnet.
- Stammdaten und Wertelisten werden per Datengrid-Service den konsumierenden Systemen in der gewünschten Form (Perspektive) bereitgestellt.
- Alle Stammdaten sind eindeutig (eindeutiger Identifier – Schlüssel) und mit Hilfe von Metadaten beschrieben.
- Bei Stammdaten, die im Zuge von Neuentwicklungen oder Erweiterungen von Anwendungen neu aufzunehmen oder zu ändern sind, werden zunächst die im Repository gespeicherten Daten auf Relevanz und Nutzbarkeit geprüft. Sofern die SSOT-Daten nicht ausreichend oder vollständig sind, müssen sie abgeglichen und ggf. aktualisiert und bereinigt oder erweitert werden. Dabei sind vorhandene Standards und Normen, die im Bereich der öffentlichen Verwaltung bereits Anwendung finden, zu berücksichtigen.
- Neu aufzunehmende oder zu ändernde Stammdaten müssen über einen definierten Prozess erfasst bzw. angepasst werden.
- Es sind Geschäftsprozesse für die Stammdatenpflege innerhalb der verschiedenen Geschäftsbereiche etabliert.
- Alle Systeme verwenden identische Stammdatenquellen.

Zeichensatz-Standard "Lateinische Zeichen in UNICODE":

Mit der Entscheidung 2014/04 hat der IT-Planungsrat in seiner 13. Sitzung den Zeichensatz festgelegt, der von IT-Verfahren in der öffentlichen Verwaltung zukünftig unterstützt werden muss. Er wird durch den Standard "Lateinische Zeichen in UNICODE" festgelegt. Diese Vorgaben und Regelungen sind folgerichtig bei der Definition und Beschreibung von Daten im Zuge des Aufbaus der zentralen Stammdatenverwaltung ebenfalls zu berücksichtigen.

Coding-Guidelines

Die Coding Guidelines (Richtlinien zur Erstellung von Programm-Code) dienen dazu, die Erstellung qualitativ hochwertiger Software zu fördern. Diese zeichnet sich unter anderem durch die Eigenschaften, die in den Normen ISO/IEC 9126 bzw. der nachfolgenden ISO/IEC 25000 definiert sind, aus. Nachstehend sind die wichtigsten dieser Eigenschaften aufgeführt, die mit diesem Dokument adressiert werden:

- Änderbarkeit / Wartbarkeit
- Benutzbarkeit
- Effizienz
- Funktionalität, hier insbesondere
    - Sicherheit
    - Interoperabilität
    - Konformität
    - Ordnungsmäßigkeit
    - Richtigkeit
- Übertragbarkeit, hier insbesondere
    - Anpassbarkeit
    - Austauschbarkeit
- Zuverlässigkeit, hier insbesondere
    - Fehlertoleranz
    - Reife
    - Wiederherstellbarkeit

Gültigkeit und Anwendung

- Die Regeln gelten in ihrer jeweils aktuellen Form für Neuimplementierung und Reimplementierungen bzw. größere Updates von Code, die im Rahmen von Projekten durchgeführt werden. Existierender Code kann, muss aber nicht, aktualisiert werden.
- Ausnahmen sind, sofern technologische Gründe dies erzwingen, möglich. Ausnahmen müssen im Rahmen eines Architekturreviews dokumentiert und in der Projektdokumentation (z.B. Abschlussbericht) festgehalten werden.
- Werden Lücken in diesen Richtlinien festgestellt oder Teile sind im Projektkontext nicht umsetzbar, so muss eine Alternative zusammen mit EAM erarbeitet werden.
- Sprachunabhängige Regeln: Die Richtlinien gelten unabhängig von der Programmiersprache und der zugrundeliegenden Architektur.
- Anforderungen, die sich aus den BAMF IT-Architektur Grundsätzen ergeben, sind in den folgenden Kapiteln zu finden.
- Die Prüfung auf Einhaltung der Richtlinien erfolgt im Rahmen von Architektur- bzw. ggf. Code-Reviews sowie durch die Nutzung des BAMF (Strive) Software-Build-Prozesses.

Nr.	Name	Beschreibung	Verbindlichkeit

CG01	Zentrales Repository	• Code wird vollständig in einem oder mehreren zentralen BAMF-git-Repositories verwaltet. Komponenten, die in der Regel einem deploybaren Artefakt entsprechen, werden in einem eigenen Repository verwaltet.
		Struktur und Namensgebung der Repositories orientieren sich am BAMF-Strive-Prozess.
		• Für jedes Repository muss eine Release Systematik vorhanden und dokumentiert sein.
		  ◦ Empfehlung: Es werden mehrstufige Release-Nummern <Major.Minor.Patch> (siehe Semantic Versioning) verwendet.
		  ◦ Die konkrete Ausgestaltung basiert stets auf Grundlage von git-Flow.
		Eine Varianz innerhalb von git-Flow wird in den Projekten selbst festgelegt, es müssen jedoch folgende Eckpunkte erfüllt sein:
		  ▪ Entwicklungen werden auf Branches und Sub-Branches durchgeführt. (dev)
		  ▪ Das konkrete Branching-Konzept muss den Betrieb von verschiedenene Maintenance-Releases (Maintenance-Branches) ermöglichen.
		  ▪ Releases werden aus dem Master nach erfolgtem Review und Merging erstellt.
		Jedes Release besitzt einen eigenen Branch, Patches werden dort mit Subbranches erstellt.
		  ▪ Releases müssen als Tag auf dem Master vorliegen.
		  ▪ Der Zustand auf dem Master muss immer qualitätsgesichert und lauffähig sein.
		
		Die Verwendung von GitFlow erleichert die Umsetzung dieser Eckpunkte (insbesondere das Branch-Benennungs-Schema und das Konzept der Release-Branches). Die Changelog Richtlinien sind zu beachten: Jedes Projekt/Git Repo sollte demnach eine CHANGELOG.md pflegen. Weitere Regeln zur Nutzung von git finden sich in der STRIVE-Anwender-Dokumentation.
			MUSS

CG02	Integrität	• Umgebungsspezifische Parameter und Konfigurationen sind ausgelagert. • Konfiguration und Code sind grundsätzlich getrennt, Konfigurationen sind ebenfalls versioniert in git abzulegen. • Software muss in der Entwicklungsinfrastruktur des BAMF gebaut werden können. Notwendige Bibliotheken und/ oder benötigte Binärartefakte müssen versioniert im git oder artifactroy des BAMF eingecheckt sein. Die Namensgebung der Artefakte/ Bibliotheken erfolgt nach Abstimmung mit EAM und die Versionierung erfolgt per Semantic Versioning-Konzept.	MUSS

CG03	Codestyle	• Sprachspezifische Konventionen werden wie folgt eingesetzt:
		  ○ Java: Google Java Style Guide
		  ○ XML: Google XML Document Format Style Guide
		  ○ übrige Sprachen: Google Style Guide Übersicht
		• Englische Sprache ist stets zu verwenden, feststehende Fachbegriff (wie z.B. Bescheid) sind weiterhin in Deutsch zu halten.
			SOLL

CG04a	Check-fit	• Software muss automatisiert gebaut (Jenkins) und mit git-Info paketiert in artifactory abgelegt werden können. • Software muss unabhängig von Jenkins automatisiert installiert und integriert werden können Erläuterung: Installationen auf Staging- und Produktionsumgebungen dürfen ausschließlich automatisch mittels Skripten durchgeführt werden. Abhängig von der Technologie sind die Standardtools zu verwenden (diese können bei EAM erfragt werden).	MUSS

CG04b	Check-Fit	• Automatisierte, strukturbasierte Tests (Unit- sowie Komponenten-Tests) müssen vorhanden sein.
		• Die Benennung der Methoden in der Testklasse muss den Zweck des Unit-Tests erkennen lassen. Beispiel: givenNormalState__whenGetServiceInfo__thenReturnHttp200
		• Strukturbasierte Tests müssen nach dem Schema **'given/ when/ then'** (‚gegeben/ wenn/ dann') aufgebaut sein.
		• Sämtliche Assertions (Prüfungen) müssen unter **'then'** erfolgen.
		• Externe Abhängigkeiten außerhalb des 'Objekts unter Test' (class, controller, application) müssen durch Mocks aufgelöst werden.
		• Strukturbasierte Tests dürfen keine Fachlichkeit enthalten, sondern folgen nur den Pfaden im Quellcode (Source Code).
		Beispiel:
		`int x;`
		`if (x = 1) {`
		`return true;`
		`} else {`
		`return false;`
		`}`
		Das Beispiel enthält zwei Pfade. Dafür sind zwei (ausschließlich zwei) Unit-Tests zu erstellen.
		X = 1 -> true
		X = 2 -> false
			MUSS

CG04c	Check-fit	• Fehlerfreier Durchlauf durch statische BAMF Codeanalyse (keine Bugs im SonarQube-Durchlauf). • Mittels statischer Code-Analyse werden der Grad der Testabdeckung sowie weitere Metriken (Bugs, Schwachstellen, Code Smells) erhoben. • Ein projektspezifisches **Quality Gate** implementiert Grenzwerte zu den Metriken und entscheidet automatisiert, ob das **Quality Gate** passiert werden darf oder nicht.	SOLL
CG05	Dokumentation	• Für die technische Dokumentation wird der „Documentation as Code"-Ansatz verwendet (ASCII Doc). Dieser muss zusammen mit dem Code versioniert verwaltet werden (git). HTML-Code zur Einbindung in Confluence wird daraus generiert (Jenkins-Job).	MUSS
CG06	Logging/Tracing	• Zentrales technisches Logging/Tracing muss verwendet werden, um End-to-End-Nachvollziehbarkeit herzustellen. (Logging Service Middleware + Sleuth im Falle von Java). Siehe auch Opentracing.	MUSS
CG07	Errorhandling	• Errorhandling mit folgenden Mindest-Anforderungen muss implementiert sein: o Adäquate Reaktion auf nicht verfügbare Dienste (Retry-Mechanismen). o Ausgabe qualifizierterer Rückmeldungen falls Dienst nicht korrekt funktioniert. (Exceptions, out-of-memory). o Behandlung fachlicher Fehler, die zum Abbruch führen (inkorrekte Daten, inkorrekter Status, inkorrekter Empfänger, unmögliche Transformation)	MUSS
CG08	Monitoring	• Monitoring-Schnittstellen des ITZBund müssen bedient werden. Ein Healthcheck muss bereitgestellt werden.	MUSS

CG09	Namenskonvention/ URLs	URLs, Applikations- und Service-Namen werden in Abstimmung mit dem Enterprise-Architektur-Management festgelegt. Hierbei wird folgendes Grundschema, das auf der Architektur-Landkarte des BAMF beruht, verwendet. *DE/BUND/BAMF/Fachdomäne/Subfachdomäne/Application/Service/[Version]* • Fachdomäne = Angabe aus Ebene 1 der Fachdomänenkarte • Subfachdomäne = Angabe aus Ebene 2/3 der Fachdomänenkarte • Application = IT-Fachanwendung • Service = Bezeichnung des Microservice (ein Aktivitätsservice wird mit einem Verb bezeichnet; ein Ressource-Service, insbes. ein lesender-Zugriff-Service, wird mit einem Substantiv bezeichnet) • Version = Major-Versionsnummer. Diese Angabe ist optional, es muss jedoch eine Mehrversionsfähigkeit sichergestellt werden (bei Nutzung von Docker/ Kubernetes mit ISTIO ist eine andere Strategie zu wählen als mit Standard-REST-Services). Die Versionierungsangabe innerhalb der URL ist vom Service-Typ abhängig (Aktivitätsservice / Ressource-Service). Bei Ressource-Services darf keine Versionsnummer in der URL angegeben werden. Bsp.: DE/BUND/BAMF/Asyl/Identitätsprüfung/DiaS/Auskunft-Service/v1 Die Namen sind eine Referenz auf die jeweiligen Einträge aus dem API-Repository, MicroService-Registry sowie dem EAM-Repository. Umlaute: • bei intern genutzten APIs sind Umlaute (ä, ö, ü) in der URL/ im Servicenamen zugelassen → bessere Lesbarkeit von deutschen Fachbegriffen • bei extern und international genutzten APIs/ Services soll nur die kanonisierte Form verwendet werden (= keine Umlaute) → Anlehnung an internationale und XÖV-Standards	MUSS
CG10	Projektstruktur	Die Implementierung, Interfaces und Mocks müssen mit einem eigenen Lebenszyklus ausgestattet sein, woraus für jedes Artefakt eine separate Projektstruktur notwendig wird. In Java ist als interne Projektstruktur (pro Artefakt) den Vorgaben von Maven zu folgen. Die Projektstruktur wird vom Projektteam erarbeitet und vor der Umsetzung zusammen mit dem EAM-Team diskutiert und genehmigt.	MUSS
CG11	Kompatibilität/ Aktualität	• Abhängigkeiten zu Libraries und Interfaces müssen bei jedem Release geprüft werden und die neueste kompatible Version verwendet werden. • Ein CVE-Check (Common Vulnerabilities and Exposures) muss erfolgen. • Die Applikation/ Software muss zur aktuellen Infrastruktur kompatibel sein	MUSS