



Register Factory & IsyFact

Was ist der Mehrwert einer Software Factory?

Ausgangslage

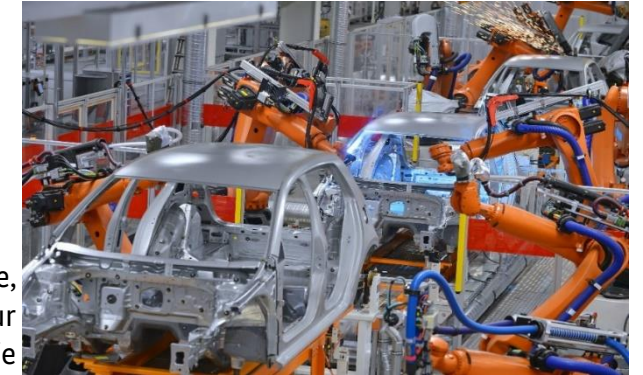
- Register sind im Kern gleich, sowohl funktional als auch nicht-funktional.
- Das BVA baut und betreibt viele Register.



Synergien ausnutzen

Standardisierung, Modularisierung, Wiederverwendung

- Konzentration auf spezifische (fachliche) Anforderungen
- Verlässliche und einheitliche Umsetzung der Anforderungen
- Reduzierung des Aufwands der
 - > Umsetzung neuer Verfahren (bis zu 30%)
 - > Weiterentwicklung der Verfahren
 - > Wartung & Betreuung der entstehenden Informationssysteme
- Einfacher Wechsel zwischen fachlichen Kontexten möglich

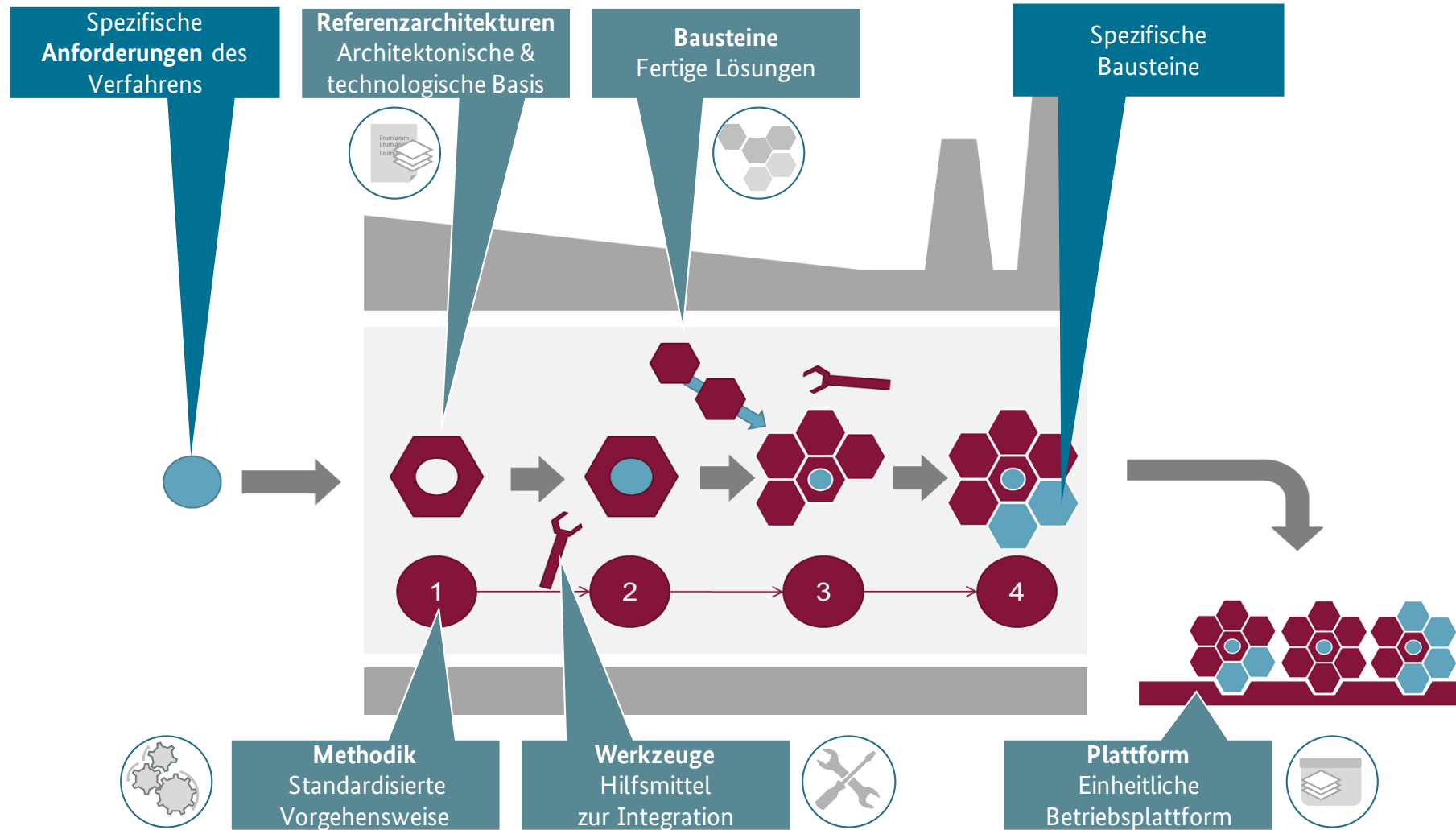


Plattformgedanke,
analog zur
Automobilindustrie

„.... A **software factory** is a structured collection of related software assets that aids in producing computer software applications or software components according to specific, externally defined end-user requirements through an assembly process.^[1] A software factory applies manufacturing techniques and principles to software development to mimic the benefits of traditional manufacturing. Software factories are generally involved with outsourced software creation....”

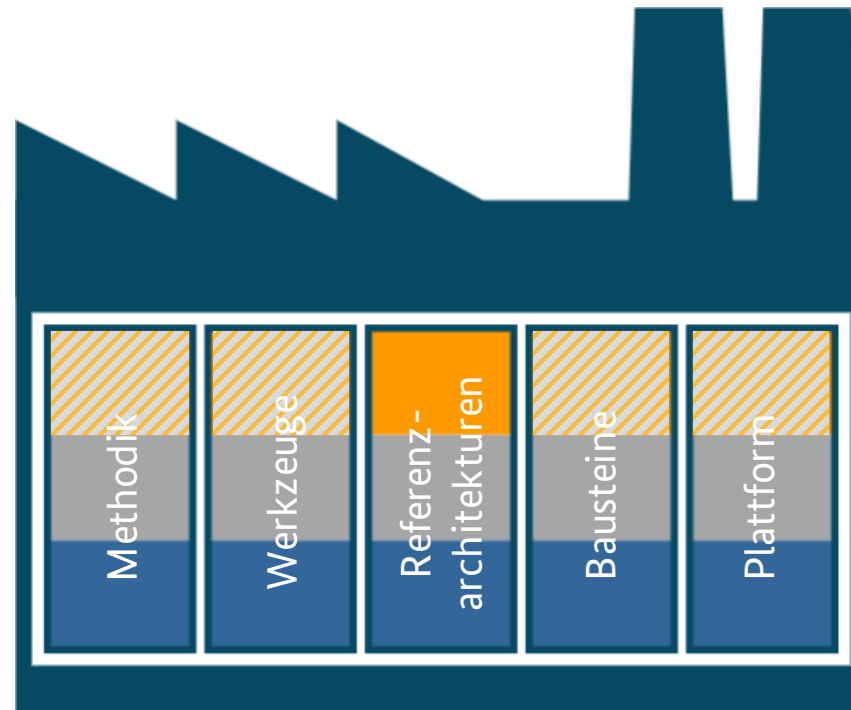
Source:
https://en.wikipedia.org/wiki/Software_factory

Das Prinzip einer Software Factory



Einführung

IsyFact Standards, IsyFact Erweiterungen, Register Factory



Aufbau der Register Factory & IsyFact (1/2)

Die Register Factory setzt auf der IsyFact auf.



IsyFact-Standards:

- Verpflichtende Bestandteile der IsyFact
- Referenzarchitektur, Grundlagenbausteine, Werkzeuge
- Open Source

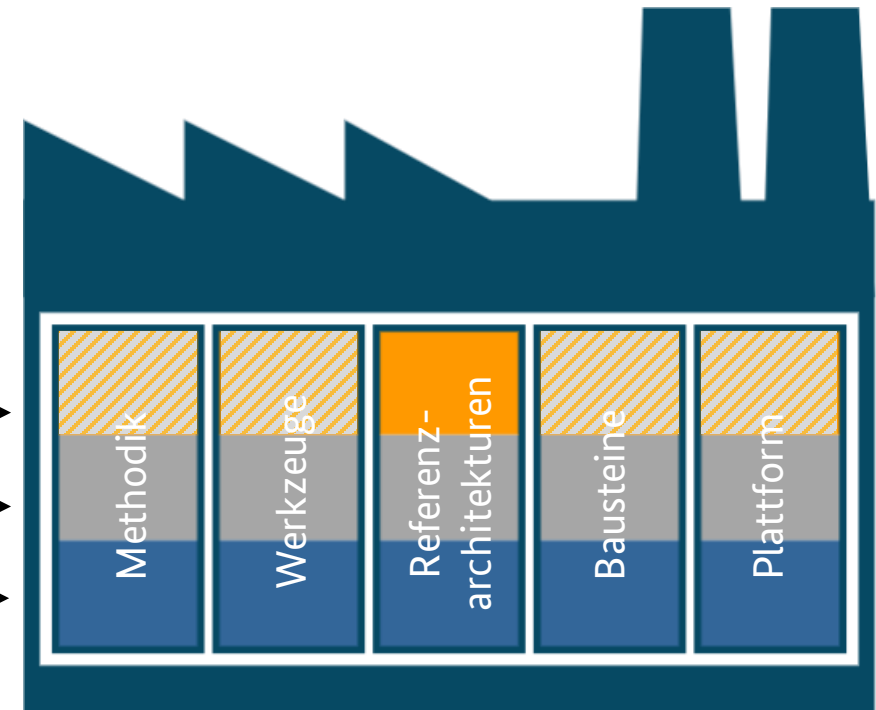
IsyFact-Erweiterungen:

- Optionale Zusätze zur IsyFact
- Meist Closed Source aber auch Open Source

Register Factory:

- Referenzarchitekturen für Registeranwendungen (u.a. Register, Spiegelregister, Beteiligungsanwendungen)
- Spezielle Bausteine und Services für den Bau von Registern
- Closed Source

Register Factory →
IsyFact-Erweiterungen →
IsyFact-Standards →



Aufbau der Register Factory & IsyFact (2/2)

Methodik

- Konventionen und Vorgaben für die Spezifikation und Konzeption und Programmierung

Werkzeuge

- Produktkatalog
- Vorgaben zu Modellierung und Konzeption, sowie die Einrichtung von Entwicklungsumgebungen

Referenzarchitekturen

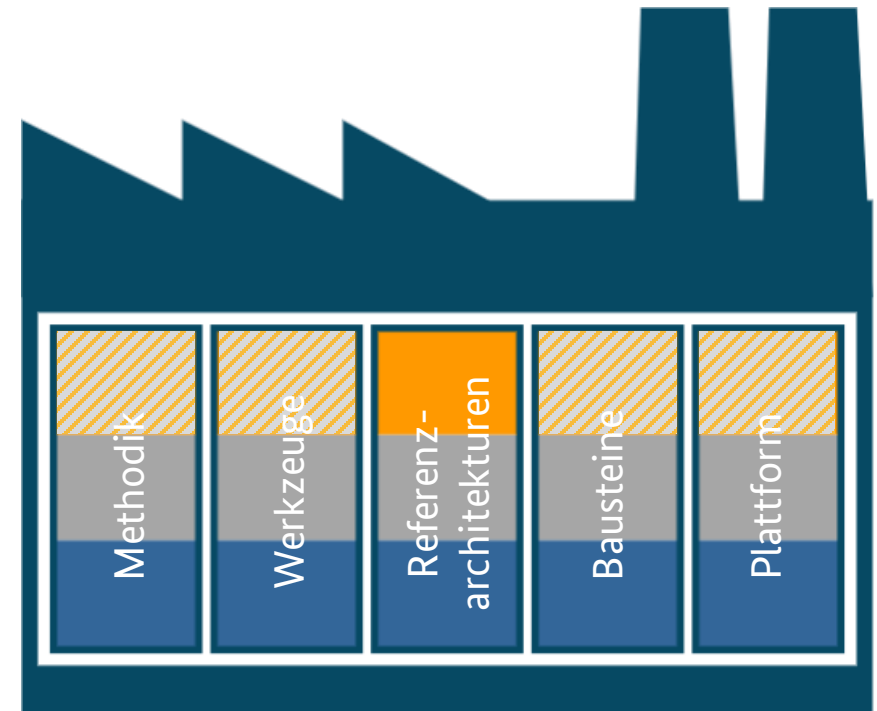
- Konzepte zur Erstellung von Anwendungslandschaften und Anwendungssystemen (fachliche, software-technische und technisch-infrastrukturelle Ebene)

Bausteine

- Programmierbibliotheken
- Konzepte

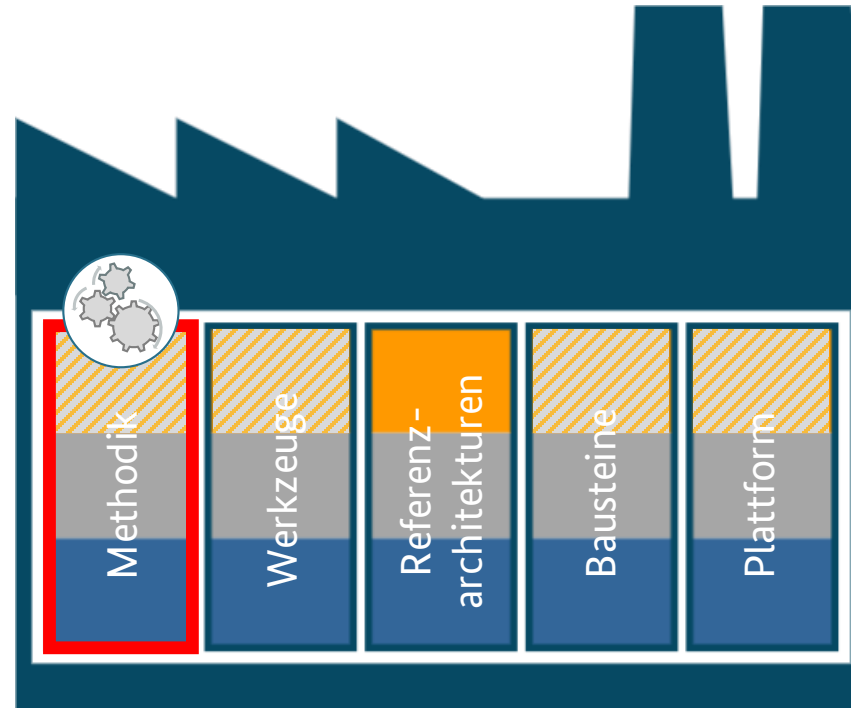
Plattform

- Vorgaben für eine einheitliche Betriebsplattform



Methodik

IsyFact & Register Factory



Was ist die Methodik?



Konventionen

- Namenskonventionen,
- Glossar,
- Programmierkonventionen für Java,
- Versionierung

Vorlagen

- für Baupläne (Systemspezifikation, Systementwurf, Systemhandbuch)
- zur Produktauswahl
- Diagrammerstellung (Modellierungs- und Diagrammdarstellungsvorgaben)

Architekturdokumentation

- Java Programmierkonventionen
- Vorlage Produktkatalog

Etablieren von Konventionen

Namenskonventionen

- Gleichartige Dinge werden gleich benannt (Präfixe, Suffixe, Begriffe)
- Anwendungsfälle, Entitäten, Schnittstellen, ...

Glossar

- Zentrale Definition wichtiger Begriffe &
- Vereinheitlichung des Sprachgebrauchs in Projekten

Programmierkonventionen

- Richtlinien für die Codierung, Vorgaben zur Benennung, Quellcodestruktur, ...
- „Fachlicher Code ist deutsch, technischer Code ist englisch.“

Richtig

```
de.bund.bva.cd.registercd.persistence.meldung
de.bund.bva.cd.registercd.service.auskunft
de.bund.bva.pliscommon.logging.common.layout
```

Falsch

```
de.bund.bva.cd.CDRegister.persistence.meldung
de.bund.bva.cd.register.cd.persistence.meldung
de.bund.bva.cd.registercd.admin.service
```

Die Methodik unterstützt in allen Phasen der Software-Entwicklung!

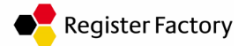


Die Methodik unterstützt in allen Phasen der Software-Entwicklung.
Durch die Methodik werden Informationssysteme mit einem einheitlichen Vorgehen erstellt und einheitlich dokumentiert.



Dokumentation einer Anwendung

- Spezifikation
 - Vorlage Systemspezifikation (Fachliche Sicht)
 - Vorlage Schnittstellendokumentation (Fachliche Sicht)
 - Anleitung Diagrammerstellung (Modellierungs- und Diagrammdarstellungsvorgaben)
- Vorlage Systementwurf (Software-technische Sicht)
- Vorlage Systemhandbuch (Technische Infrastruktur)
- Vorlage Anforderungsliste
- Vorlage Produktauswahl (Dokumentation von Produktauswahl)



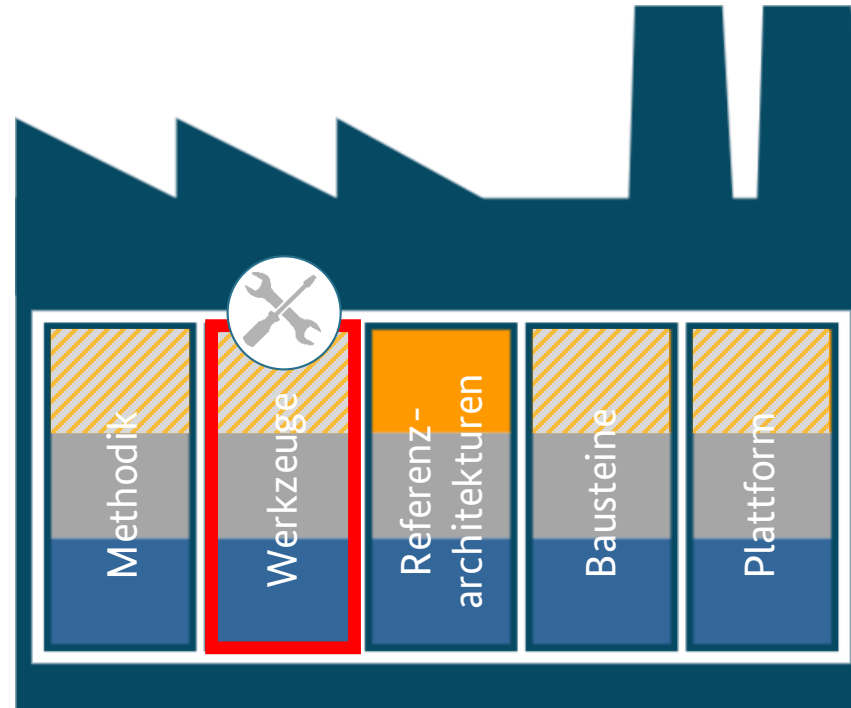
Keine registerspezifischen Methoden enthalten

Software-Entwicklung

- Java Programmierkonventionen
- Namenskonventionen
- Versionierung

Werkzeuge

IsyFact & Register Factory





Was sind Werkzeuge?

- Werkzeuge vereinfachen Abläufe in der Softwareentwicklung
- Beispielweise Anleitungen zur Einrichtung einer Entwicklungsumgebung

Sinnvolle Vor-Konfigurationen und Nutzungsvorgaben

Definierte Werkzeuge für alle relevanten Bereiche

- Modellierung und Konzeption
- Generierung
- Software-Entwicklung: Editoren,
- Build, Deployment
- Entwickler- und Integrationstests
- Dokumentation

Modellierung und Konzeption

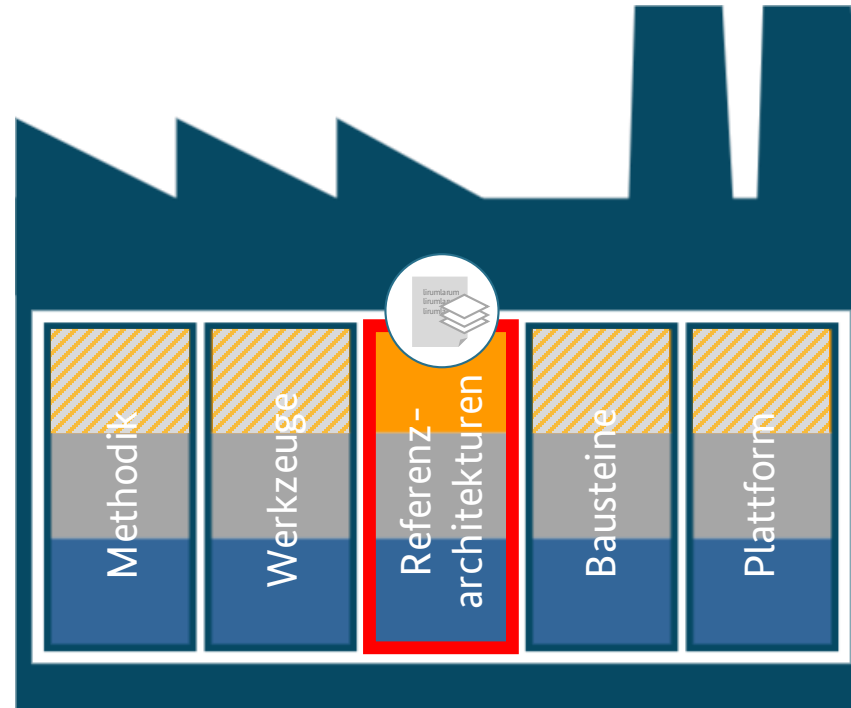
- UML-Profil für SparxSystems Enterprise Architect

Einrichtung der Entwicklungsumgebung

- Eclipse & IntelliJ IDEA
- Checkstyle, Templates, Code-Formatierung, Generierung von Boilerplate-Code

Durch die Werkzeuge werden bestimmte Abläufe der Softwareentwicklung einheitlich durchgeführt und führen schneller zu gleichartigen Ergebnissen.

Referenzarchitektur

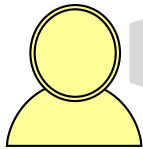


Was sind Referenzarchitekturen*?

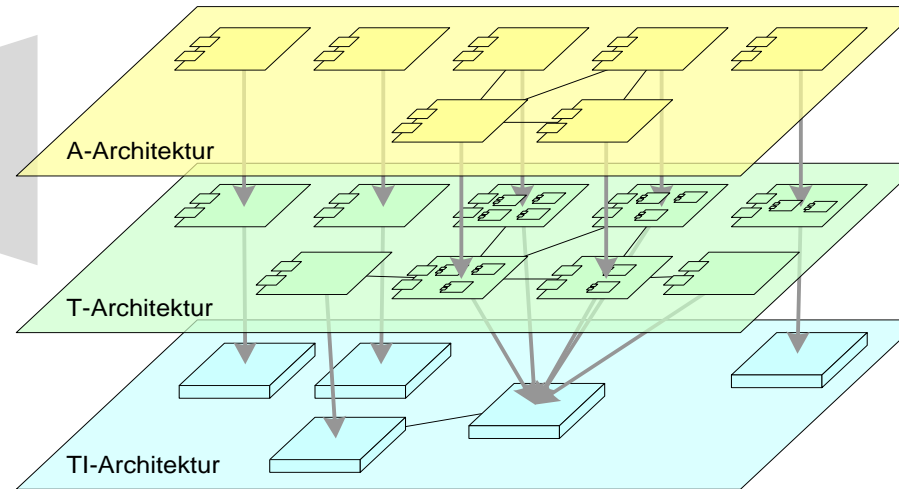


Baupläne für Informationssysteme aus unterschiedlichen Sichten!

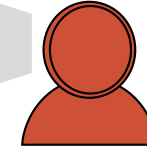
Fachliche Architektur (A-Architektur)



- Fachbereich
- Fachlicher Architekt
- Fachlicher Designer
- Anforderungsanalytiker



Softwaretechnische Architektur (T-Architektur)



- Technischer Architekt
- Technischer Designer
- Entwickler

Architektur der technischen Infrastruktur (TI-Architektur)



- Betrieb / Betriebsdienstleister

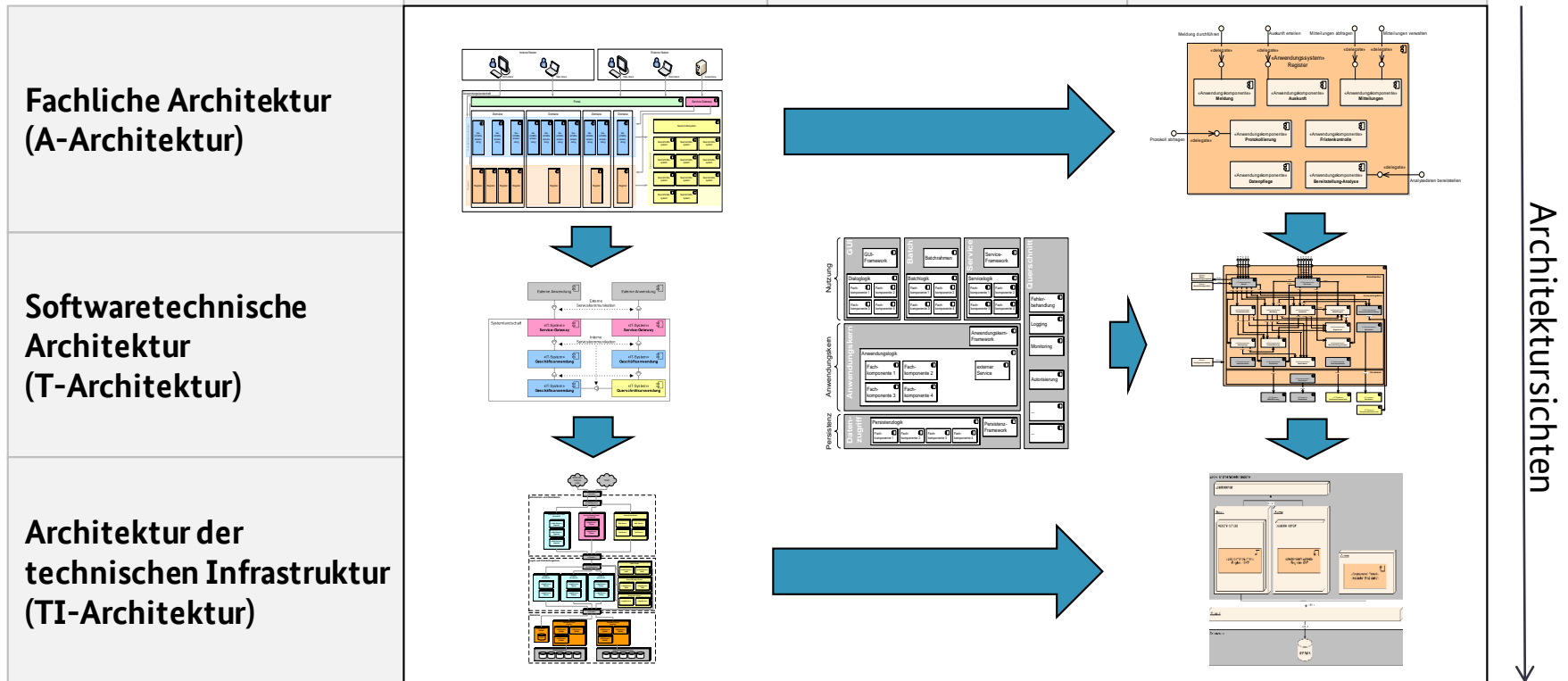
* ehem. Blaupausen; Begriffsanpassung

Einordnung der Referenzarchitekturen in Sichten und Ebenen



Referenzarchitekturen ermöglichen die Steuerung durch den Fachbereich über einen definierten Prozess von der Spezifikation bis zur Realisierung!

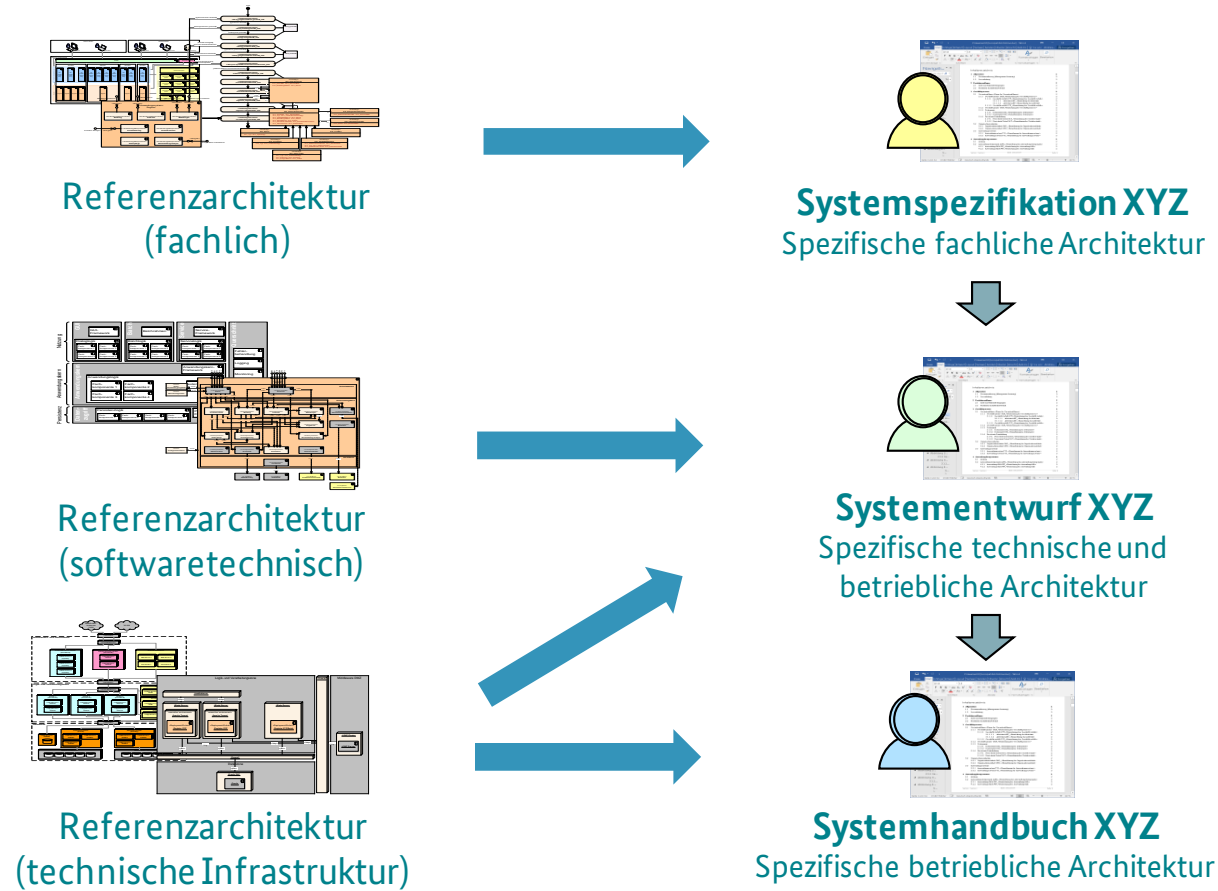
Architekturebenen (Detaillierung) →



Wie liegen Referenzarchitekturen vor?



Als Referenzarchitektur und Vorlagen für systemspezifische Baupläne!



Zweck

- Strukturierter Überblick über die Anwendungslandschaft aus fachlicher Sicht:
 - Die Menge aller Anwendungssysteme einer Organisation und deren Nutzungsbeziehungen untereinander bilden eine Anwendungslandschaft.
 - Die fachliche Architektur einer Anwendungslandschaft gibt die Strukturierung aus fachlicher Sicht vor und legt fest, wie Anwendungssysteme in die Anwendungslandschaft integriert werden.

Inhalte

- Strukturierung der Anwendungslandschaft in fachliche Hierarchieebenen:
 - Anwendungsdomäne
 - Anwendungssystem (Geschäftsanwendungen und Querschnittsanwendungen)
- Bestandteile der Anwendungslandschaft

Vorgaben der Referenzarchitektur

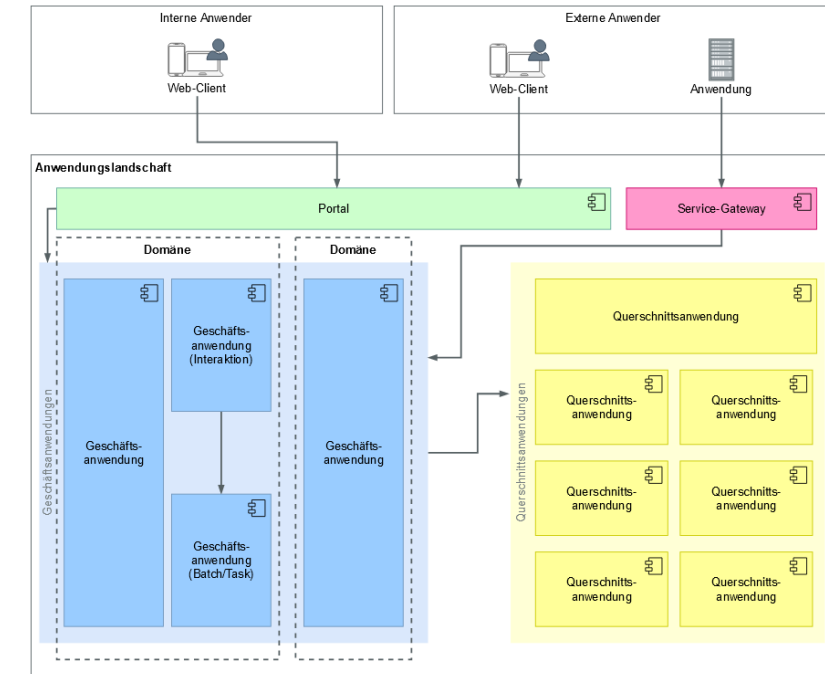
- Die Anwendungslandschaft wird in Anwendungsdomänen und Anwendungssysteme strukturiert.
- Es gibt drei Typen von Anwendungssystemen: Register, Geschäftsanwendungen und Querschnittsanwendungen.
- Die Anwendungslandschaft stellt einen in sich geschlossenen Sicherheitsbereich dar.
- Portal und Service-Gateway bilden die Schnittstellen zu den Anwendern und externen Systemen.
- Querschnittliche Services werden für die Register und Geschäftsanwendungen bereitgestellt.

Checkliste: Einsatz der Referenzarchitektur

Als Teil der fachlichen Architektur ist Folgendes zu definieren:

- ☒ verschiedene Arten von Geschäftsanwendungen innerhalb des Anwendungskontexts
- ☒ Aufgaben und Verantwortlichkeiten der einzelnen Arten von Geschäftsanwendungen
- ☒ Interaktionen und Abhängigkeiten zwischen den Geschäftsanwendungen, insbesondere zulässige Kommunikationsbeziehungen

Architektur



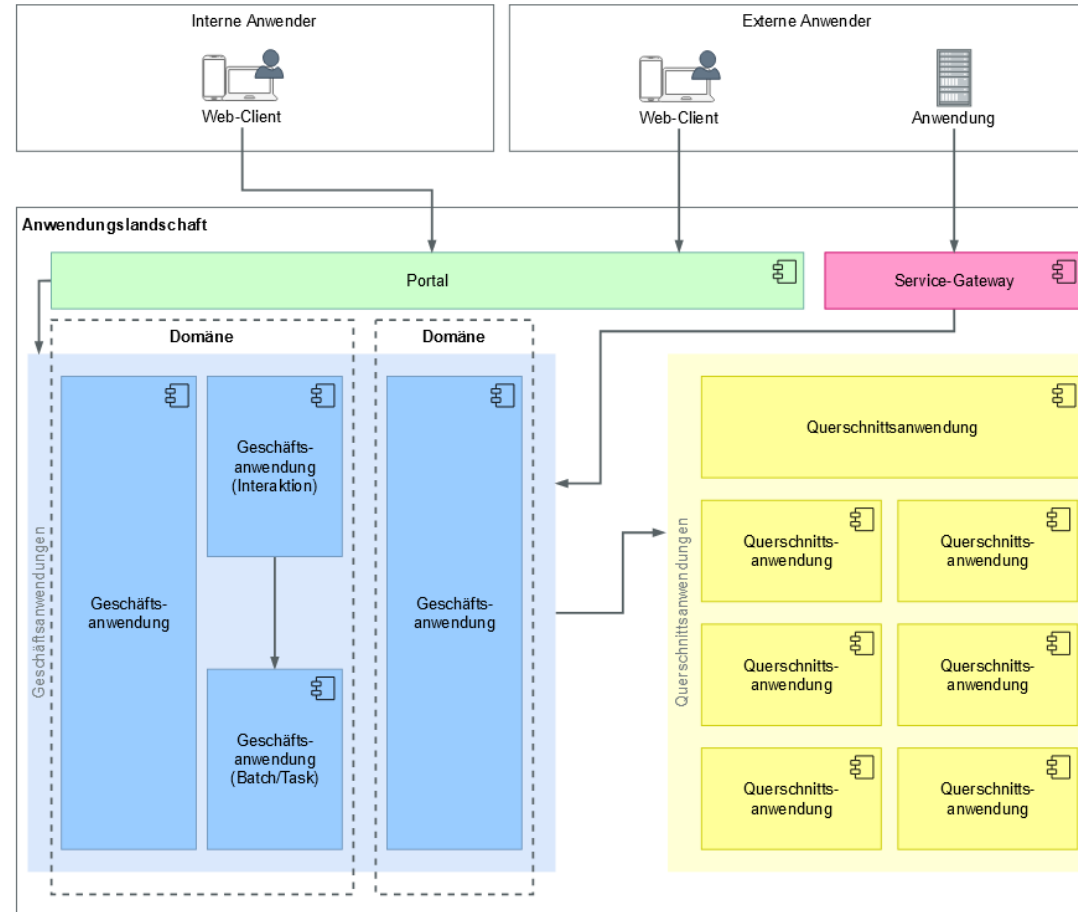
Zweck

- Strukturierter Überblick über die Anwendungslandschaft aus fachlicher Sicht:
 - Die Menge aller Anwendungssysteme einer Organisation und deren Nutzungsbeziehungen untereinander bilden eine Anwendungslandschaft.
 - Die fachliche Architektur einer Anwendungslandschaft gibt die Strukturierung aus fachlicher Sicht vor und legt fest, wie Anwendungssysteme in die Anwendungslandschaft integriert werden.

Inhalte

- Strukturierung der Anwendungslandschaft in fachliche Hierarchieebenen:
 - Anwendungsdomäne
 - Anwendungssystem (Geschäftsanwendungen und Querschnittsanwendungen)
- Bestandteile der Anwendungslandschaft

Architektur



Portal

Zentraler Zugangspunkt der Web-Oberfläche

Service Gateway

Schnittstelle zu Anwendungen außerhalb der Plattform

Geschäftsanwendung

Behördliche Informationssysteme

Querschnittsanwendung Bsp.: BHVZ, SVZ *

Querschnittlich verwendete Funktionalität und Daten, eine Instanz pro Anwendungslandschaft

Zweck

Umsetzung der Anwendungslandschaft (fachlich) in eine Systemlandschaft (software-technisch):

- Umsetzung der in der fachlichen Architektur definierten Elemente (Anwendungssysteme, Anwendungskomponenten, fachliche Entitäten, Anwendungsfälle etc.) in IT-Systeme, Komponenten, Klassen, physische Datenmodelle etc.

Inhalte

- Strukturierung der Systemlandschaft
- Schnittstellen und Aufrufbeziehungen pro Systemtyp und zwischen Systemtypen (Geschäftsanwendung, Querschnittsanwendung, Service-Gateway, Portal)
- Servicekommunikation (intern, extern, synchron, asynchron)
- Webservices
- Queuing

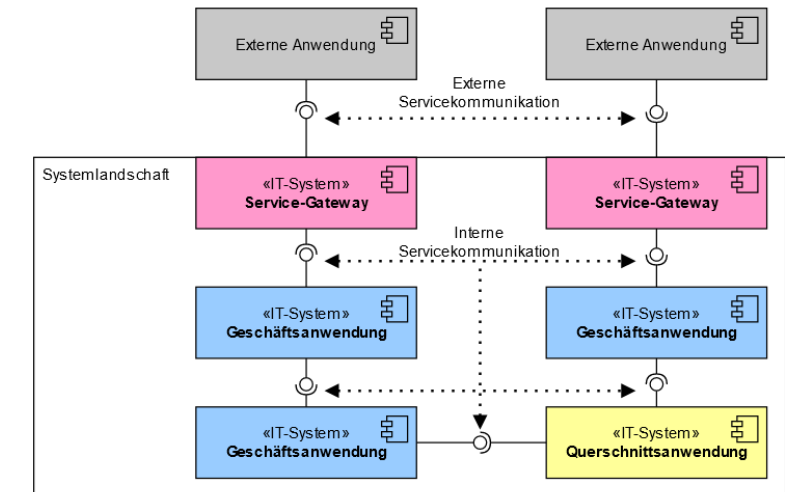
Vorgaben der Referenzarchitektur

- Kommunikation mit externen IT-Systemen über Webservices und Service-Gateways
- Synchrone Service-Aufrufe finden über REST Schnittstellen statt und werden sowohl zur internen als auch externen Servicekommunikation genutzt.
- Queuing über JMS (Jakarta Messaging, ehemals Java Message Service) ausschließlich in der internen Servicekommunikation
- Vorgaben zur Verwendung von Produkten

Checkliste: Einsatz der Referenzarchitektur

- ☒ Erstellen eines konkreten Systementwurfs nach den Vorgaben der Referenzarchitektur Software und der Referenzarchitektur IT-System

Architektur



Zweck

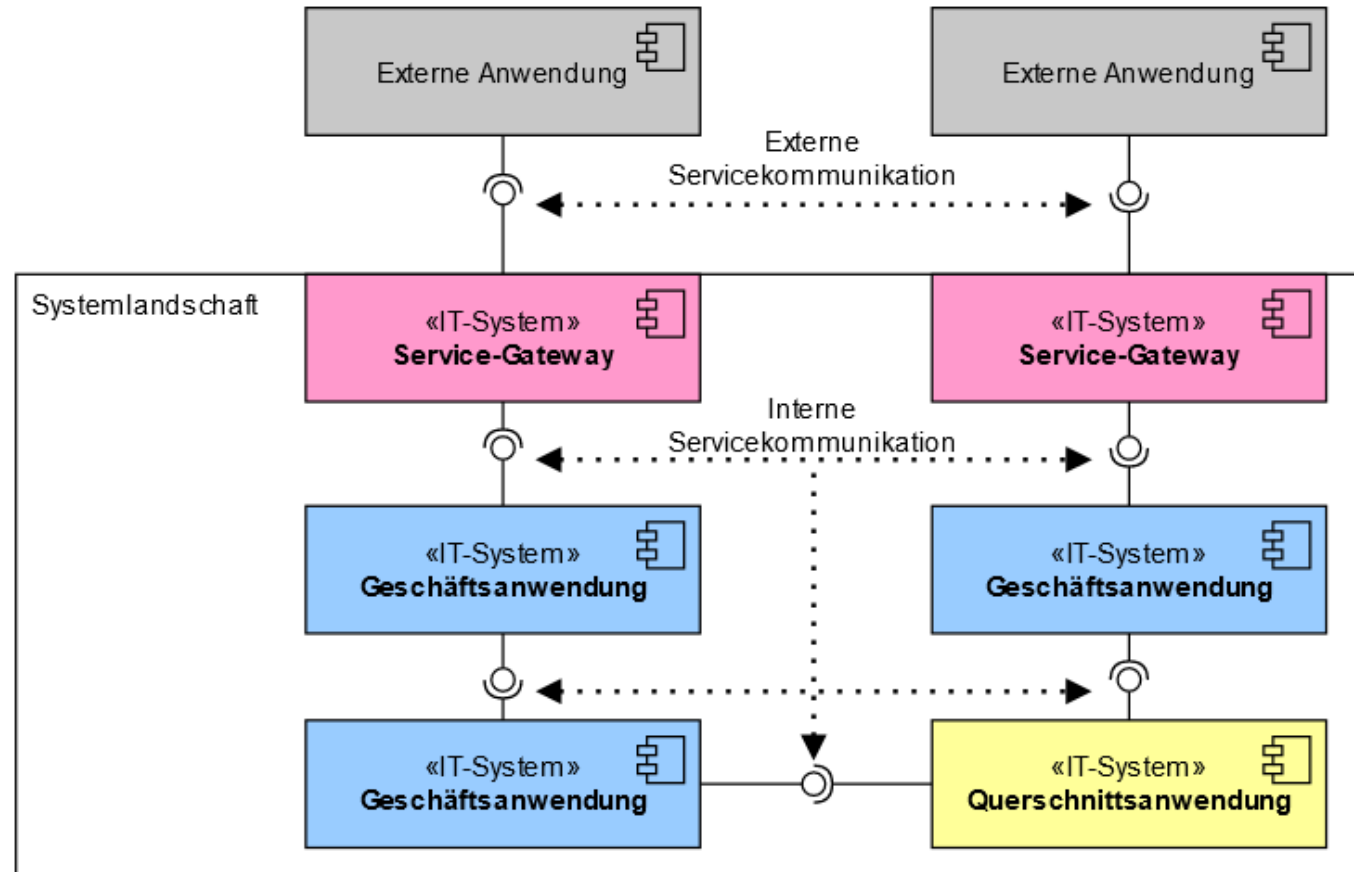
Umsetzung der Anwendungslandschaft (fachlich) in eine Systemlandschaft (software-technisch):

- Umsetzung der in der fachlichen Architektur definierten Elemente (Anwendungssysteme, Anwendungskomponenten, fachliche Entitäten, Anwendungsfälle etc.) in IT-Systeme, Komponenten, Klassen, physische Datenmodelle etc.

Inhalte

- Strukturierung der Systemlandschaft
- Schnittstellen und Aufrufbeziehungen pro Systemtyp und zwischen Systemtypen (Geschäftsanwendung, Querschnittsanwendung, Service-Gateway, Portal)
- Servicekommunikation (intern, extern, synchron, asynchron)
- Webservices
- Queuing

Architektur



Zweck

Die Referenzarchitektur der technischen Infrastruktur, TI-Architektur, beschreibt den Aufbau der Betriebsumgebung für die IT-Systeme einer Anwendungslandschaft.

Dazu gehören:

- Physische Geräte (Rechnersysteme, Netzwerkverbindungen und -komponenten, Drucker etc.)
- Installierte Systemsoftware (Betriebssystem, Applikationsserver, Middleware, Datenbanksystem)
- Zusammenspiel von Hardware und Systemsoftware

Inhalte

TI-Architektur einer Anwendungslandschaft:

- U.a. Staging-Umgebung, Produktionsumgebung

TI-Architektur einer Anwendung:

- Zonenzuweisung der IT-Systeme einer Anwendung
- Kommunikationsverbindungen und -protokolle zu Nachbarsystemen
- Skalierungsfähigkeit
- Anforderungen an Systemumgebung (z.B. Applikationsserver)

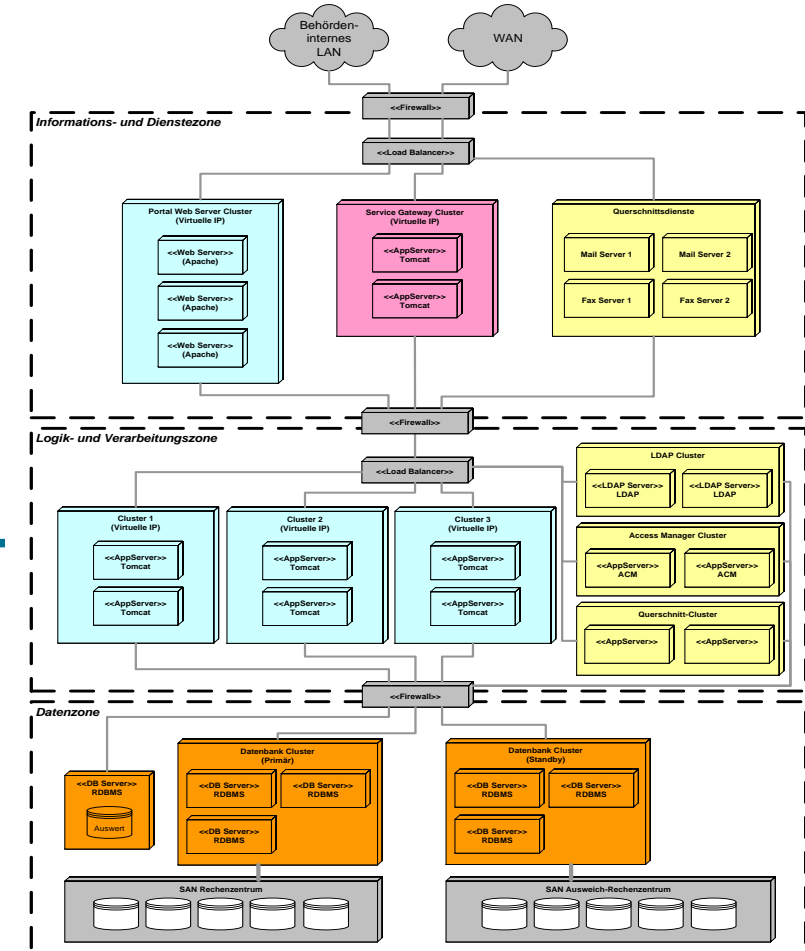
Vorgaben der Referenzarchitektur

- TI-Architektur der Anwendungslandschaft
 - SAGA Konformität
 - Netzwerktrennung
 - UML-Modellierung
- TI-Architektur einer Anwendung
 - Aufteilung der Anwendung in mehrere IT-Systeme
 - UML-Modellierung

Checkliste: Einsatz der Referenzarchitektur

- ☒ Konkrete Ausprägungen der TI-Architektur-Vorgaben

Architektur



Zweck

Die Referenzarchitektur der technischen Infrastruktur, TI-Architektur, beschreibt den Aufbau der Betriebsumgebung für die IT-Systeme einer Anwendungslandschaft.

Dazu gehören:

- Physische Geräte (Rechnersysteme, Netzwerkverbindungen und -komponenten, Drucker etc.)
- Installierte Systemsoftware (Betriebssystem, Applikationsserver, Middleware, Datenbanksystem)
- Zusammenspiel von Hardware und Systemsoftware

Inhalte

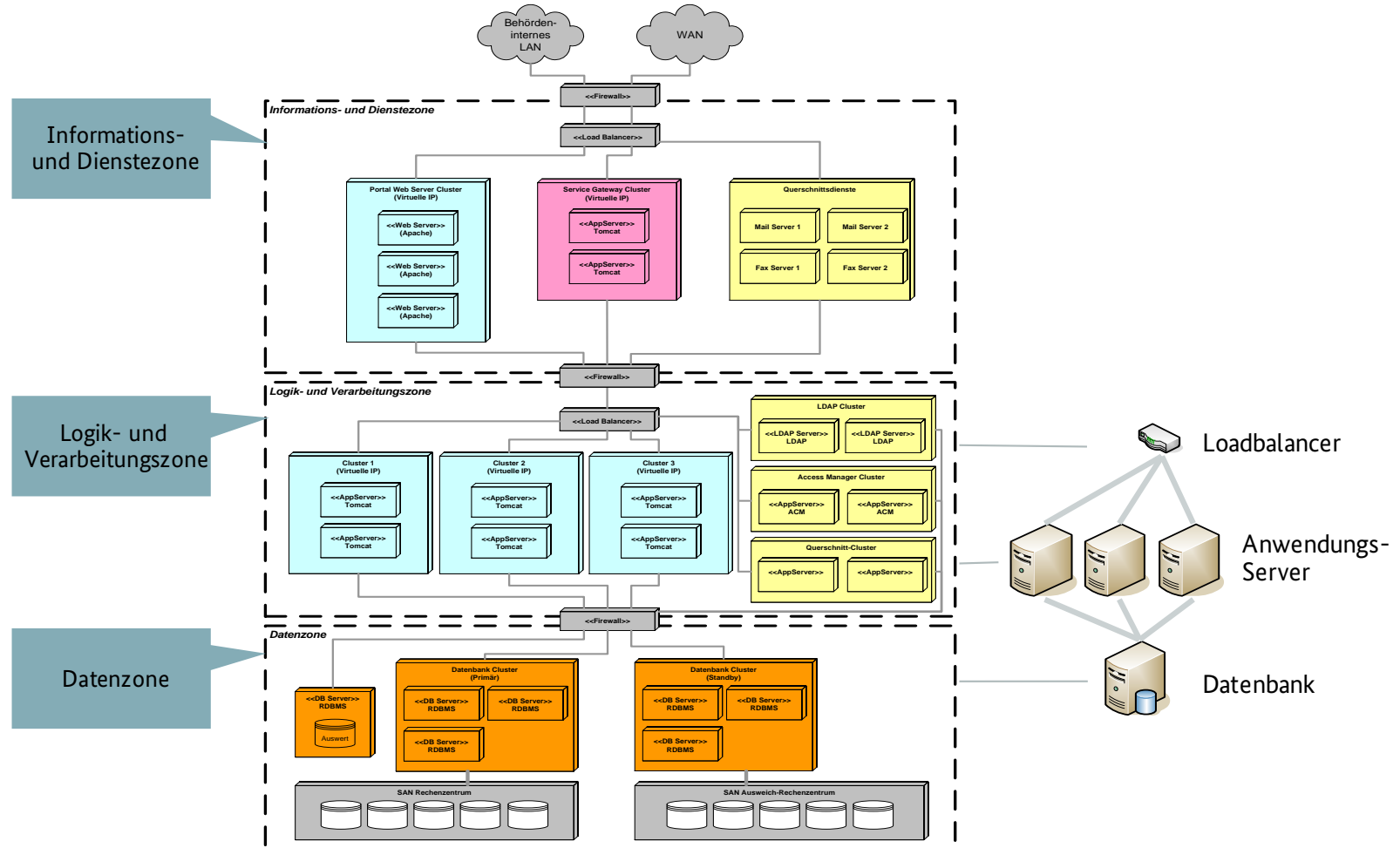
TI-Architektur einer Anwendungslandschaft:

- U.a. Staging-Umgebung, Produktionsumgebung

TI-Architektur einer Anwendung:

- Zonenzuweisung der IT-Systeme einer Anwendung
- Kommunikationsverbindungen und -protokolle zu Nachbarsystemen
- Skalierungsfähigkeit
- Anforderungen an Systemumgebung (z.B. Applikationsserver)

Architektur



Zweck

- Vorgaben zur Beschreibung und Umsetzung eines IT-Systems der Systemlandschaft
- Technisch gleichartiger Aufbau der individuell erstellten Anwendungssysteme der Systemlandschaft
- Erhöhung der Wartbarkeit

Inhalte

- Strukturierung eines IT-Systems
- Vorgaben zur Drei-Schichten-Architektur
 - Nutzung
 - Anwendungskern
 - Persistenz (Datenzugriff)
- Unterteilung der Nutzungsschicht in GUI, Batch und Service
- Nutzung von Querschnittsfunktionalitäten

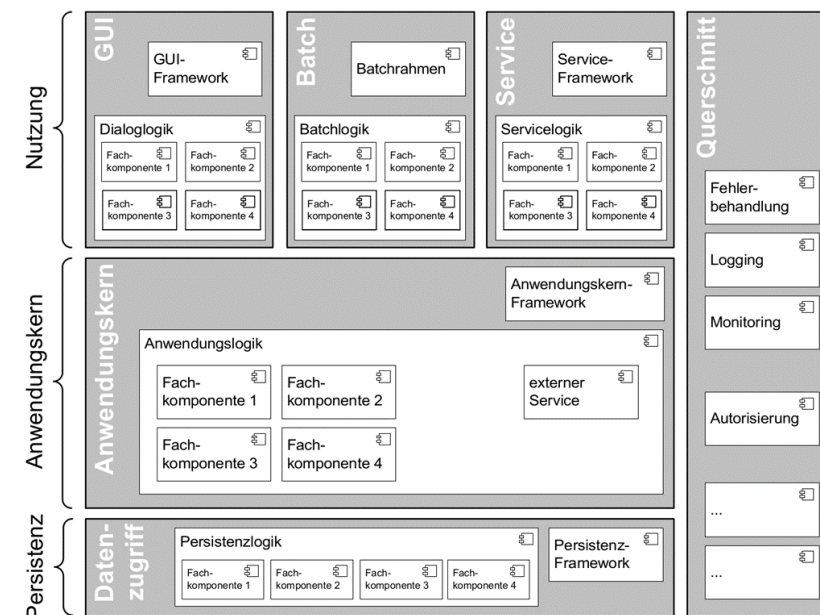
Vorgaben der Referenzarchitektur

- GUI
 - Single Page Application: Angular
- Anwendungskern
 - Fachkomponenten sind technische Umsetzung der fachlichen Anwendungskomponenten
 - Framework: Spring
- Persistenz (Datenzugriff)
 - Transaktionen werden von Nutzungsschicht gesteuert
 - Strukturierung der Fachkomponenten der Persistenz-Schicht gemäß den Komponenten der fachlichen Architektur
 - O/R-Mapping
 - Optimistische Sperrstrategie
 - Verwendung von JPA/Hibernate
- Service
 - Service-Komponenten werden entsprechend den Anwendungskern-Komponenten geschnitten
 - Service-Framework: REST

Checkliste: Einsatz der Referenzarchitektur

- ☒ Erstellen eines konkreten Systementwurfs nach den Vorgaben der Referenzarchitektur Software und der Referenzarchitektur IT-System

Architektur



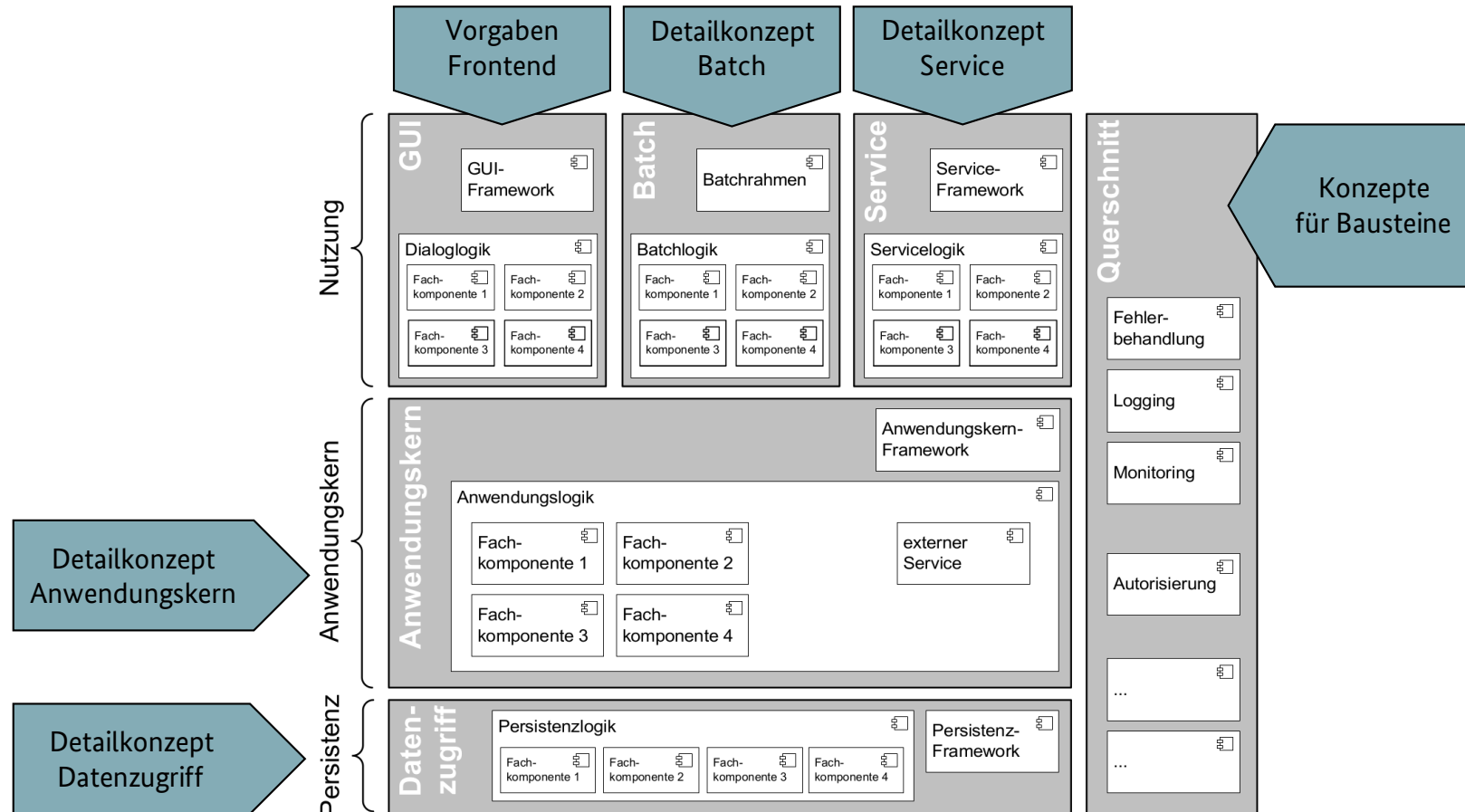
Zweck

- Vorgaben zur Beschreibung und Umsetzung eines IT-Systems der Systemlandschaft
- Technisch gleichartiger Aufbau der individuell erstellten Anwendungssysteme der Systemlandschaft
- Erhöhung der Wartbarkeit

Inhalte

- Strukturierung eines IT-Systems
- Vorgaben zur Drei-Schichten-Architektur
 - Nutzung
 - Anwendungskern
 - Persistenz (Datenzugriff)
- Unterteilung der Nutzungsschicht in GUI, Batch und Service
- Nutzung von Querschnittsfunktionalitäten

Architektur



Zweck

- Vorgaben für den Aufbau des Anwendungskerns in einem individuell erstellten IT-System

Inhalte

- Bestandteile des Anwendungskerns
- Einbindung externer Services
- Unterteilung des Anwendungskerns in fachliche Teilanwendungen
- Übergabe von Objekten zwischen Teilanwendungen

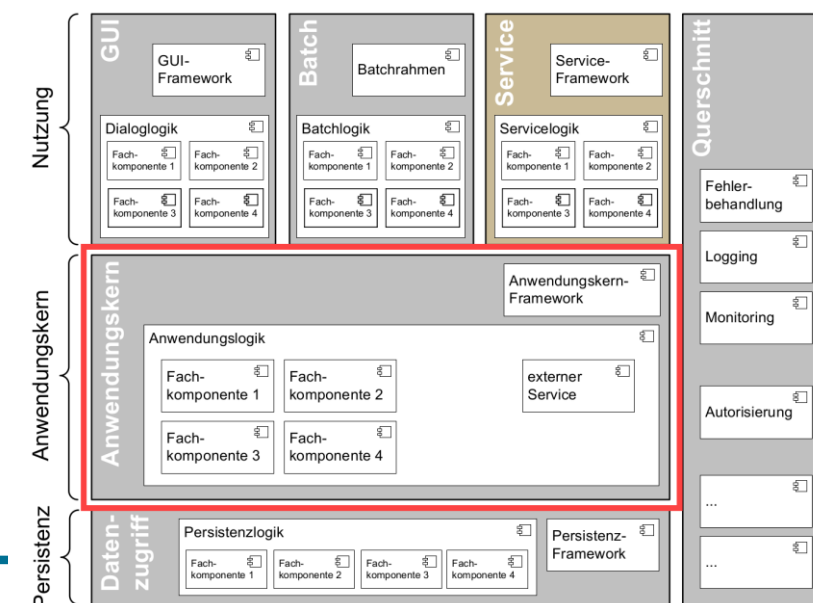
Vorgaben der Referenzarchitektur

- Zwei wesentliche Bestandteile: Fachkomponenten und Anwendungskern-Framework
- Nutzungen externer Services werden als eigene Komponenten in den Anwendungskern integriert.
- Zwischen zwei Teilanwendungen dürfen nur Objekte übergeben werden, deren Modifikation keine Auswirkungen auf die liefernde Teilanwendung hat.
- Innerhalb einer Teilanwendung dürfen fachliche Objekte auch über Schichtgrenzen hinweg übergeben werden.

Checkliste: Einsatz der Referenzarchitektur

- ☒ Erstellen eines konkreten Systementwurfs nach den Vorgaben der Referenzarchitektur
- ☒ Bilden von Komponenten mit Interfaces nach Vorgabe der Spezifikation
- ☒ Konfiguration der Komponenten mittels Spring
- ☒ Implementieren von Datenobjekten für den Datenaustausch zwischen den Komponenten

Architektur



Externe Services werden in der Regel über einen Adapter angebunden.

Keine Abhängigkeit des Fachdatenmodells auf Datenmodell des externen Services

Zentrale Stelle für notwendige Konvertierungen und Berechnungen

- Umbau von Objektstrukturen
- Konvertierung von Einheiten
- Mapping von Schlüsseln

Robustheit gegenüber Änderungen des externen Services

Der Adapter bricht, nicht direkt die Fachkomponente

Umsetzungshinweise:

Wenn eine Fachkomponente den Service benötigt...

- Umsetzung in eigenem Package „innerhalb“ der Fachkomponente
- Konfiguration über die Fachkomponente

Wenn mehrere Fachkomponenten den Service benötigen...

- Umsetzung in eigenem Package „innerhalb“ des AWK
- Konfiguration über den AWK

Architektur ähnlich zu der einer Fachkomponente

- Aufrufe des Services statt Anwendungsfälle
- Funktionen für gemeinsam genutzte Logik

Zweck

- Vorgaben für den einheitlichen technischen Aufbau der Persistenzschicht aller individuell erstellten IT-Systeme der Anwendungslandschaft

Inhalte

- Konfiguration von JPA und Hibernate
- Nutzung von JPA und Hibernate
- Eigenschaften des Persistenz-Klassenmodells
- Definition der Mappings zwischen Objekten und Datenbank
- Vorgaben für Datenbankabfragen

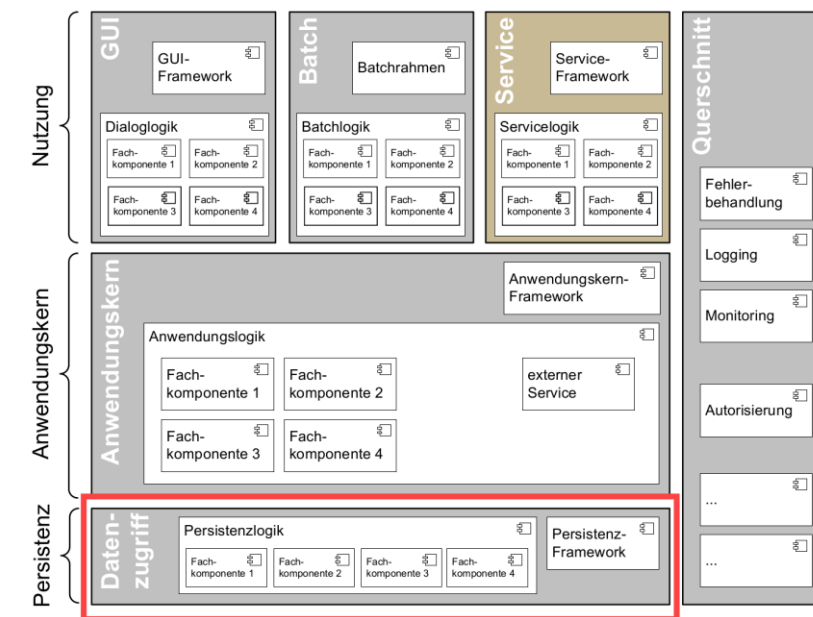
Vorgaben der Referenzarchitektur

- Keine Fachlogik in persistenten Klassen
- Definition der Mappings erfolgt über Annotationen
- Abbildung von Zeitangaben und Booleschen Variablen
- Zugriff auf JPA nur über Data-Access-Objekte (DAO)
- Für Datenbank-Abfragen JPQL nutzen
- Optimistisches Sperrverfahren ist der Standard
- Konfiguration von JPA mittels Spring Beans
- Konfiguration von Hibernate
- Eine Transaktion pro Anfrage, Transaktionssteuerung über Anwendungs- oder Nutzungsschicht

Checkliste: Einsatz der Referenzarchitektur

- ☒ Konfigurieren von JPA über Spring Beans
- ☒ Konfigurieren von Hibernate
- ☒ Implementieren des persistenten Klassenmodells
- ☒ Annotieren der persistenten Klassen für JPA
- ☒ Implementieren der DAOs

Architektur



Zweck

- Einheitlicher Aufbau von Service-Schnittstellen für alle Anwendungen der Anwendungslandschaft
 - Einsatz des Bausteins „REST“
 - Einfache Erstellung von Serviceschnittstellen

Inhalte

- Softwaredesign der Komponente "Service":
 - Aufruf der Fachlogik
 - Prüfung von Berechtigungen
 - Mapping von Parametern und Rückgabewerten
 - Transaktionssteuerung
 - Fehlerbehandlung
- Nachvollziehbarkeit von Aufrufen über Anwendungsgrenzen hinweg:
 - Erzeugen und Weitergabe von Korrelations-IDs zur Identifikation des Aufrufs
- Versionierung von Service-Schnittstellen

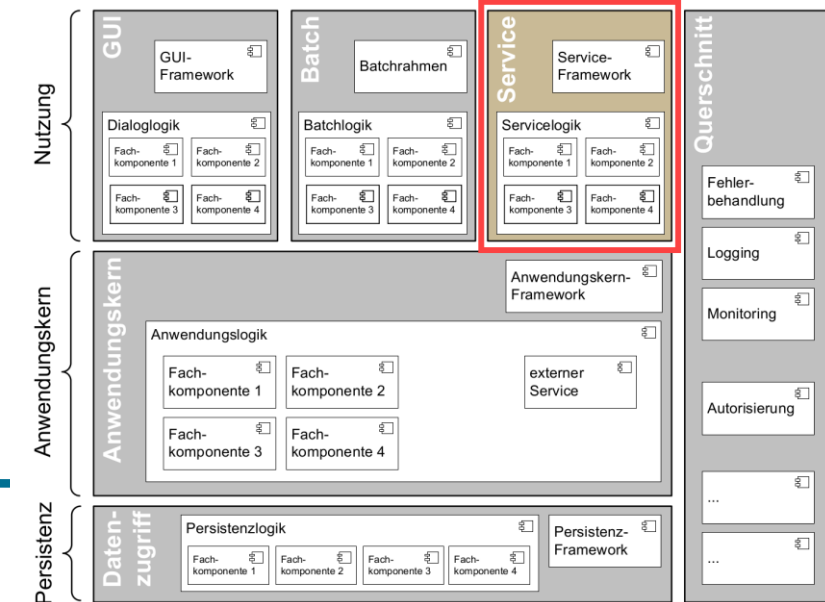
Vorgaben der Referenzarchitektur

- Verwendung des REST Bausteins (Konzept)
- Einsatz mehrerer IsyFact-Bausteine für:
 - Logging (teilweise mit AOP)
 - Fehlerbehandlung (für Schnittstellen)
 - Security
- Übersetzung von Transportobjekten (an der Schnittstellen) zu Geschäftsobjekten des Anwendungskerns

Checkliste: Einsatz der Referenzarchitektur

- ☒ Implementierung der Fehlerbehandlung
- ☒ Implementierung der Services
 - ☒ Berechtigungsprüfung
 - ☒ Transaktionssteuerung
 - ☒ Umwandlung von Transportobjekten zu Geschäftsobjekten
- ☒ ggf. Definition des Bean-Mappings

Architektur



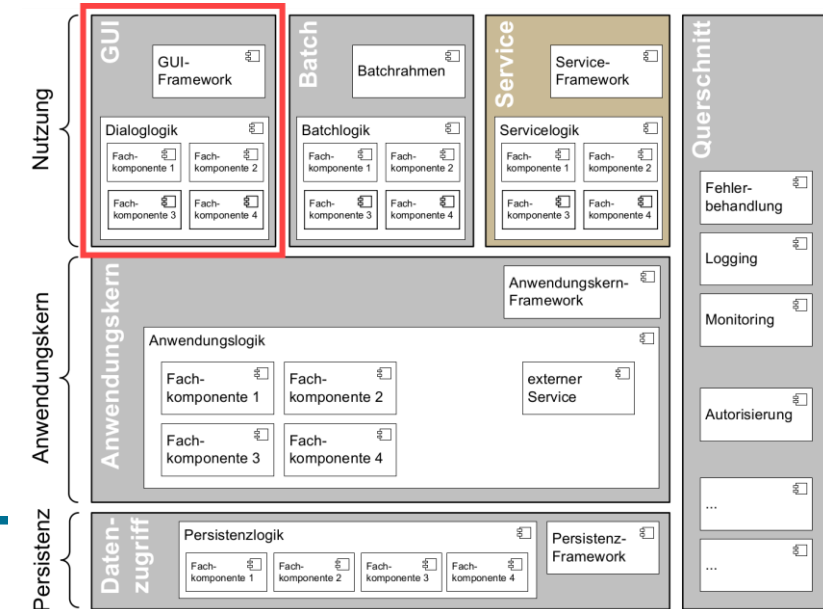
Zweck

- Schnelle und konsistente Umsetzung von Web-GUIs nach modernen Architekturmustern.

Vorgaben der Referenzarchitektur

- Vorgaben zu Seitenrahmen, Navigation und Gestaltung üblicherweise vorkommender Masken (Auskünfte, Meldungen, Suche)

Architektur



Inhalte

- Vorgaben zur Umsetzung von Dialogen und Masken mit Validierung und Datenkonvertierung
- Vorgaben zur Definition von Dialogabläufen
- Themeing zur Gestaltung bestimmter Elemente (z.B. Eingabefelder, Fehler, modale Dialoge)
- Bedienkonzept

Checkliste: Einsatz der Referenzarchitektur

- ☒ Basis-Konfiguration von Spring
- ☒ Entwurf und Konfiguration der Dialogabläufe
- ☒ Entwurf und Umsetzung von Masken, Controllern und UI Models
- ☒ Verwendung der Stylesheets

Zweck

- Einheitlicher Aufbau und Steuerung von Batches
- Fertige Lösung für typische Anforderungen an Aufbau, Ablauf und Steuerung von Batches
- Einsatz der Bibliothek "Batchrahmen"

Inhalte

- Umsetzung eines „Batchrahmens“ zur Steuerung und Realisierung konkreter Batches inklusive:
 - Paralleler Verarbeitung
 - Restartfähigkeit
 - Ergebnisprotokoll
 - Überwachung
- Persistierung des Batchstatus
- Konfigurationsmöglichkeiten
- Von der Anwendung separates Deployment
- Beschreibung der Rückgabewerte und der Struktur des Ergebnisprotokolls

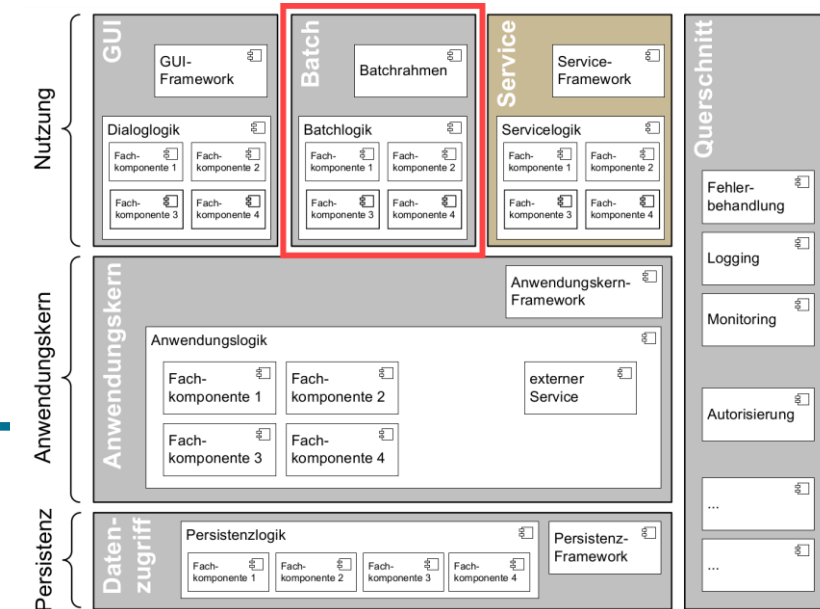
Vorgaben der Referenzarchitektur

- Die fachliche Verarbeitung wird vollständig an die Anwendung delegiert. Der Batchrahmen steuert den internen Ablauf.
- Batches bilden eine eigene Deployment-Einheit und enthalten den kompletten Anwendungskern
- Pflege des Batch-Zustands in Datenbanktabellen

Checkliste: Einsatz der Referenzarchitektur

- ☒ Erstellen eines Batch-Projekts und Aufnahme in die Maven-Konfiguration
- ☒ Erweitern des RPM-Deployments um eine Batch-Anwendung (spec-Dateien anlegen)
- ☒ Einbinden des Batchrahmens (JAR)
- ☒ Implementieren der Batch-Logik (Satzverarbeitung)
- ☒ Aufrufen der Schnittstelle des Anwendungskerns, welcher die fachliche Funktionalität bereitstellt
- ☒ Anlegen der Datenbanktabellen für die Batches

Architektur



Zweck

- Strukturierung von Registern in Komponenten und deren zugehörigen Schnittstellen.
- Identifizierung und Beschreibung der Fachlichkeit, die in nahezu allen Registern identisch umgesetzt werden muss.
- Entwurfsmuster für die Spezifikation, Konstruktion und Realisierung von Registern.

Inhalte

- Anwendungskomponenten eines Registers
- Ablaufschritte von Auskünften
- Ablaufschritte von Meldungen
- Fachliches Datenmodell
- Fachliche Architektur
- Software-Architektur

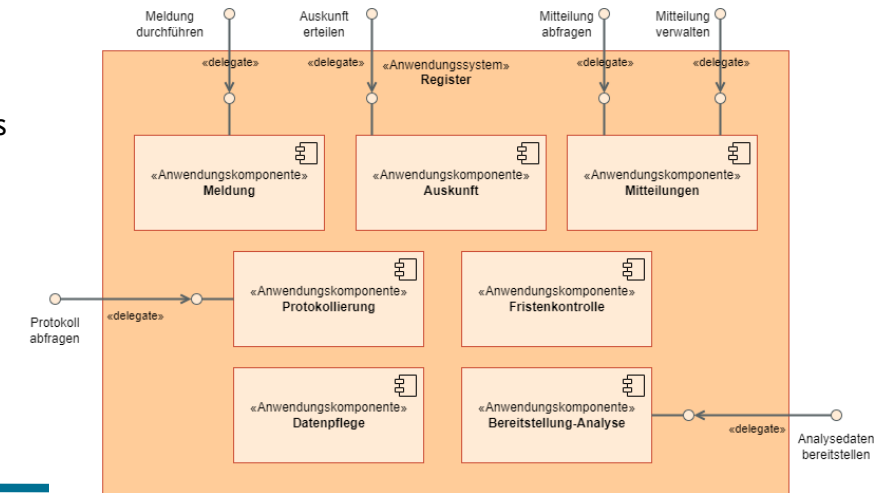
Vorgaben der Referenzarchitektur

- Register haben in der Regel den gleichen Komponentenschnitt.
- Entwurfsmuster für die Spezifikation:
 - Abläufe, Anwendungskomponenten, fachliches Datenmodell eines typischen Registers
- Entwurfsmuster für die Konstruktion:
 - Fachliche und technische Komponenten, Schnittstellen eines typischen Registers

Checkliste: Einsatz der Referenzarchitektur

- ☒ Konkretisierung der Abläufe, Anwendungskomponenten und des fachlichen Datenmodells für das zu spezifizierende Register.
- ☒ Konkretisierung der benötigten fachlichen und technischen Komponenten sowie der Schnittstellen für das zu entwerfende Register.

Architektur



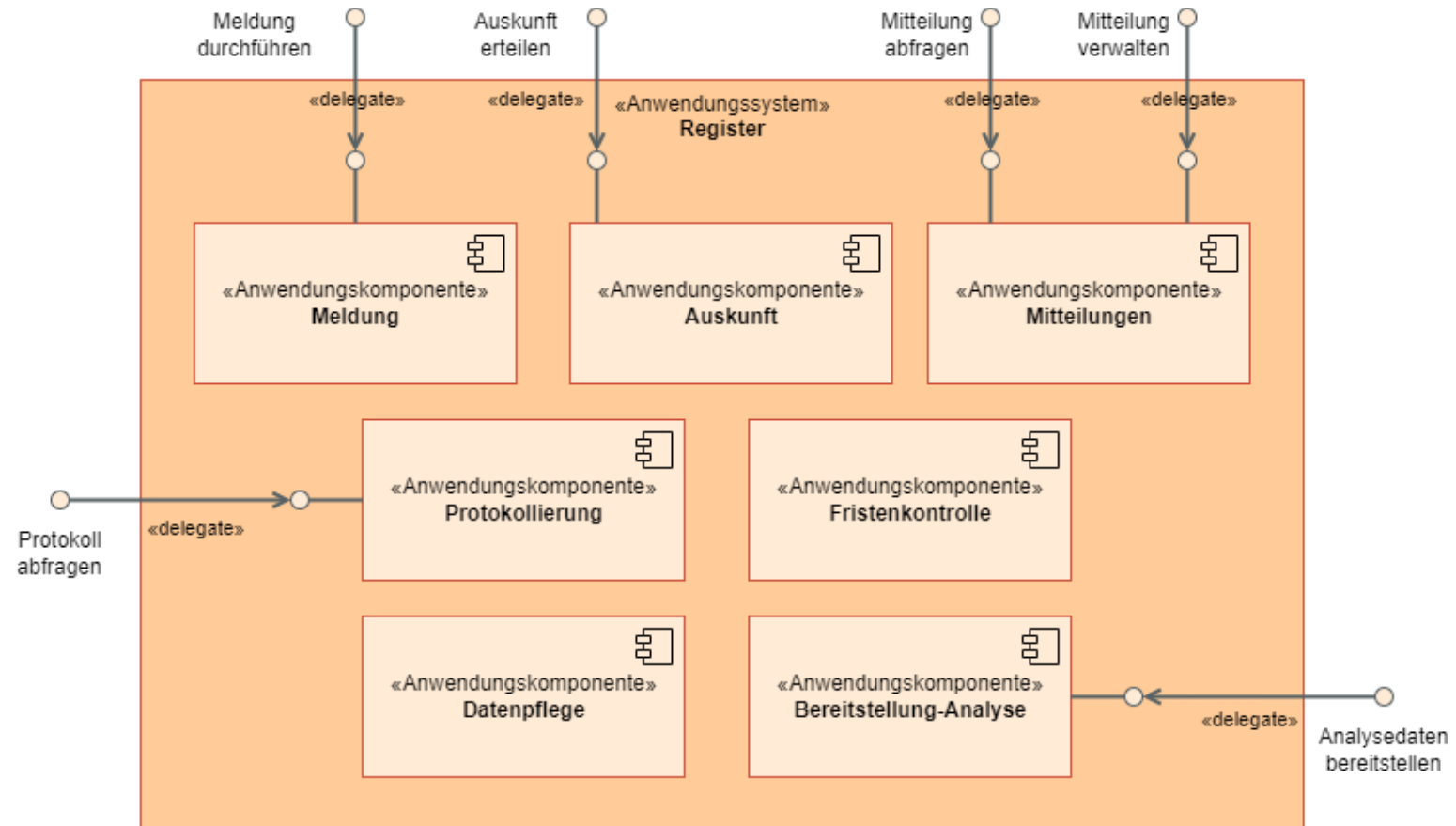
Zweck

- Strukturierung von Registern in Komponenten und deren zugehörigen Schnittstellen.
- Identifizierung und Beschreibung der Fachlichkeit, die in nahezu allen Registern identisch umgesetzt werden muss.
- Entwurfsmuster für die Spezifikation, Konstruktion und Realisierung von Registern.

Inhalte

- Anwendungskomponenten eines Registers
- Ablaufschritte von Auskünften
- Ablaufschritte von Meldungen
- Fachliches Datenmodell
- Fachliche Architektur
- Software-Architektur

Architektur



Zweck

- Ergänzung der Referenzarchitektur Register um solche Register, die Bestände anderer, registerführender Behörden in die eigene Anwendungslandschaft „spiegeln“.
- Beschreibung der gemeinsamen fachlichen Basis der Spiegelregister.

Inhalte

- Grundlage für die Spezifikation, Konstruktion und Realisierung von Spiegelregistern.
- Beschreibung der Unterschiede zwischen Registern und Spiegelregistern.
- Beschreibung unterschiedlicher Arten der Datenanlieferung.

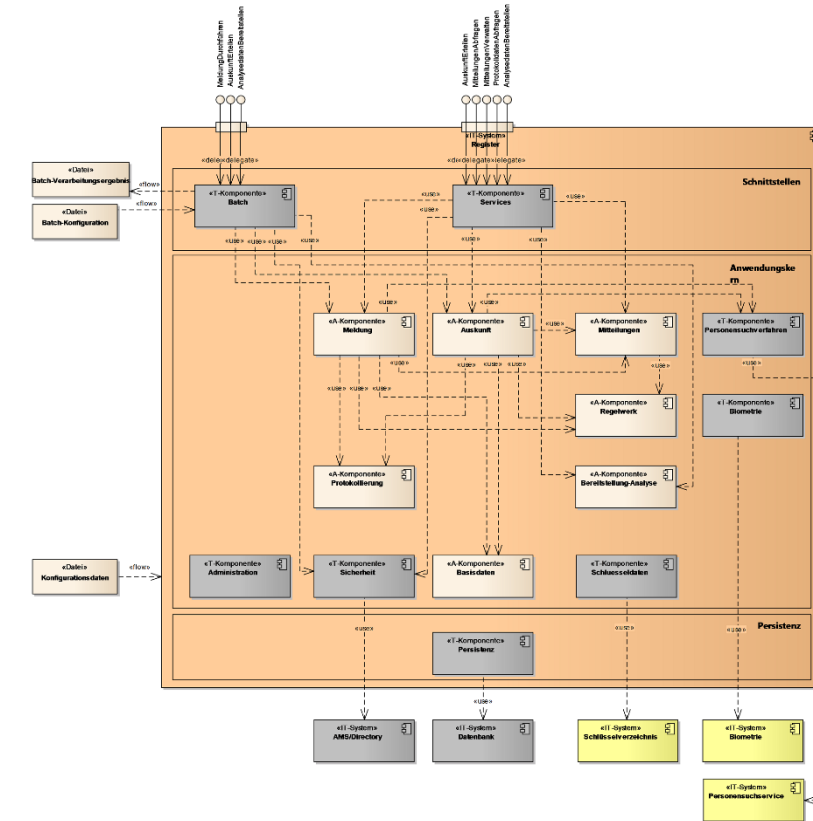
Vorgaben der Referenzarchitektur

- Implementierung gemäß der Referenzarchitektur Register.
- Spezifikation der Einbettung in das System und die Anpassung der Komponente Meldung für die:
 - Initiale Befüllung,
 - Gesamtdatenlieferung,
 - Deltalieferung
- Wegfall von Komponenten zur Datenpflege und Fristenkontrolle.
- Technischer Aufbau von laufend aktualisierten Spiegelregistern als vollwertige Registeranwendungen.

Checkliste: Einsatz der Referenzarchitektur

- ☒ Konkretisierung der Ergänzungen zur Referenzarchitektur Register für das zu spezifizierende Spiegelregister.

Architektur



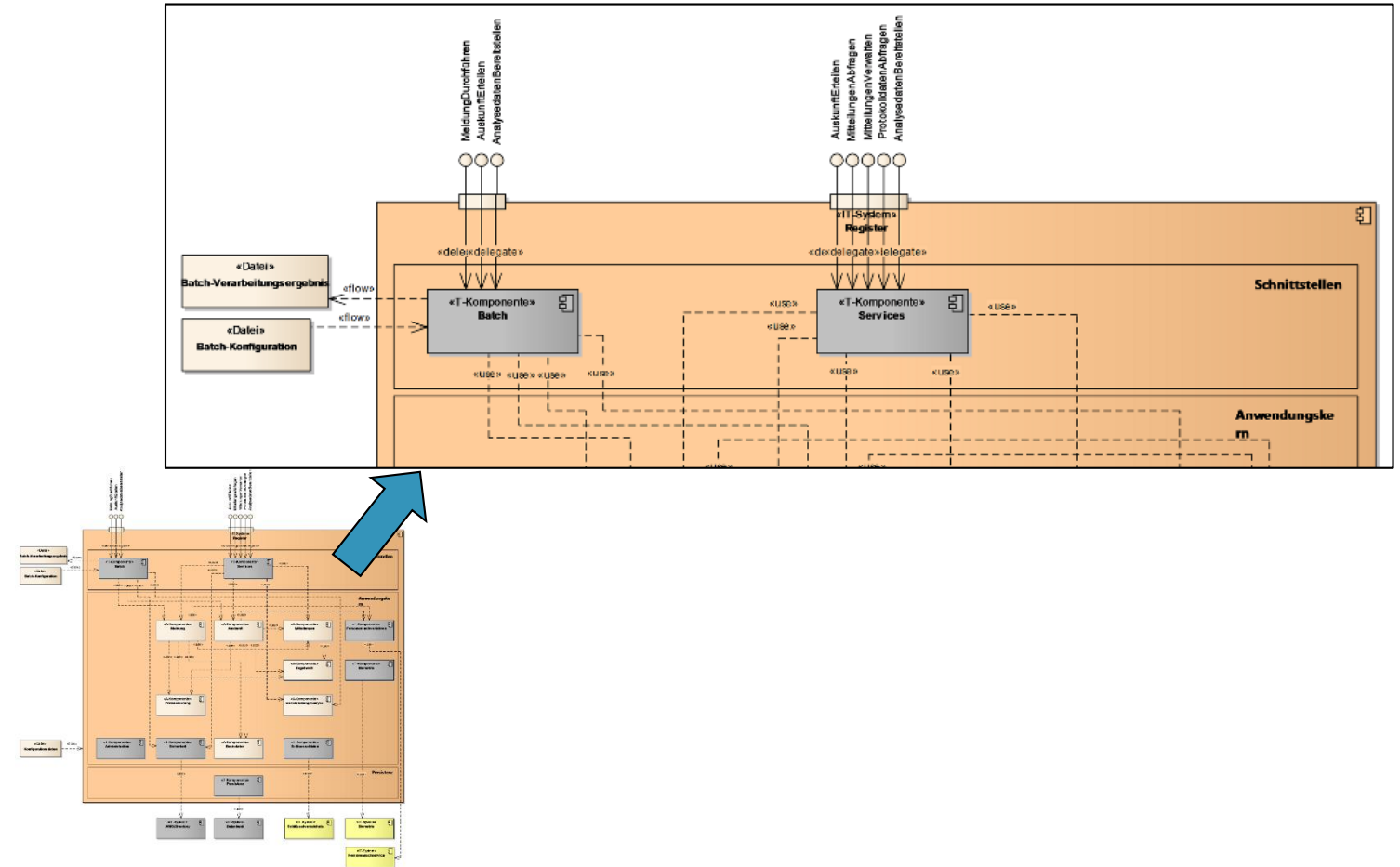
Zweck

- Ergänzung der Referenzarchitektur Register um solche Register, die Bestände anderer, registerführender Behörden in die eigene Anwendungslandschaft „spiegeln“.
- Beschreibung der gemeinsamen fachlichen Basis der Spiegelregister.

Inhalte

- Grundlage für die Spezifikation, Konstruktion und Realisierung von Spiegelregistern.
- Beschreibung der Unterschiede zwischen Registern und Spiegelregistern.
- Beschreibung unterschiedlicher Arten der Datenanlieferung.

Architektur





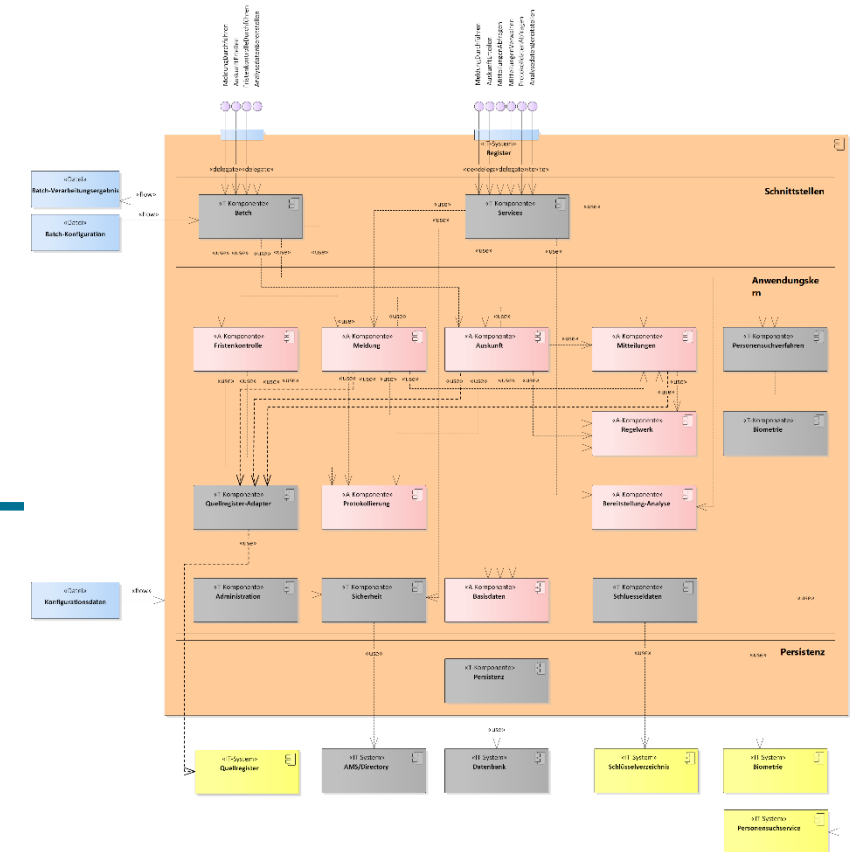
Zweck

- Der skizzierte Lösungsansatz soll als Orientierungshilfe für die Spezifikation neuer Proxyregister dienen. Er ist nicht in allen Details als verbindliche Vorgabe zu verstehen, sondern kann eingeschränkt, verändert oder erweitert werden.

Vorgaben der Referenzarchitektur

- Einsatzzweck und rechtliche Vorgaben für den Einsatz eines Proxyregisters
- Vorgaben zu Auftragsmeldung, Auftragsannahme
- Vorgaben für zweistufige unscharfe Suche

Architektur



Inhalte

- Grundlagen für die Spezifikation und Systementwurf

Checkliste: Einsatz der Referenzarchitektur

- ☒ Konkretisierung der Ergänzungen zur Referenzarchitektur Register für das zu spezifizierende Proxyregister.

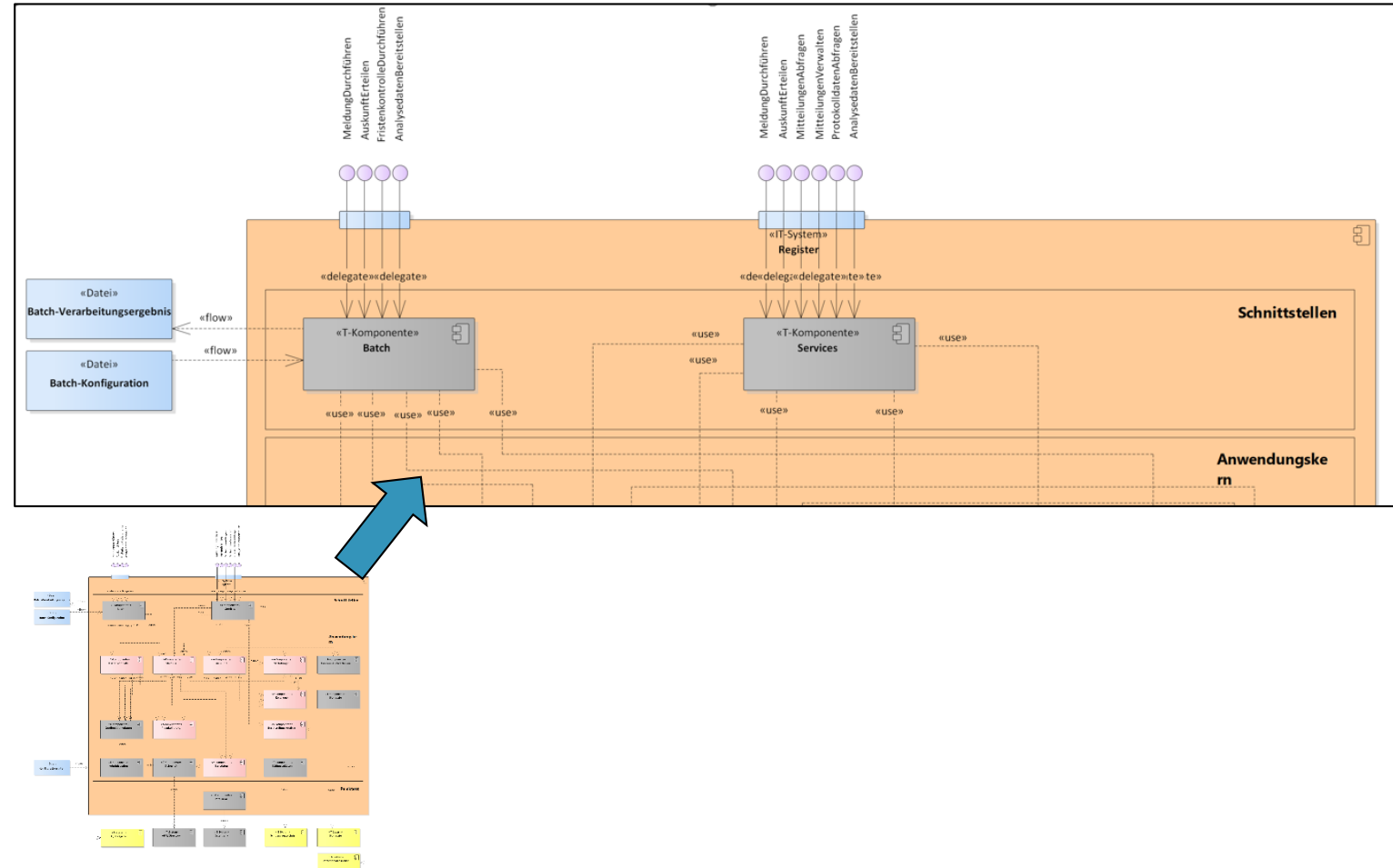
Zweck

- Der skizzierte Lösungsansatz soll als Orientierungshilfe für die Spezifikation neuer Proxyregister dienen. Er ist nicht in allen Details als verbindliche Vorgabe zu verstehen, sondern kann eingeschränkt, verändert oder erweitert werden.

Inhalte

- Grundlagen für die Spezifikation und Systementwurf

Architektur



Zweck

- Allgemeine Beschreibung von Anwendungen, deren Hauptfunktion die Präsentation von Daten für einen menschlichen Benutzer ist.
- Präsentationsanwendungen haben in der Regel keine längeren GUI-Prozesse, sondern kurz dauernde Nutzertransaktionen, die auf technische Transaktionen abgebildet werden können.

Inhalte

- Eigenschaften einer Präsentationsanwendung und Einbettung in die Anwendungslandschaft
- Systementwurf
 - Außensicht
 - Innensicht
 - TI-Architektur

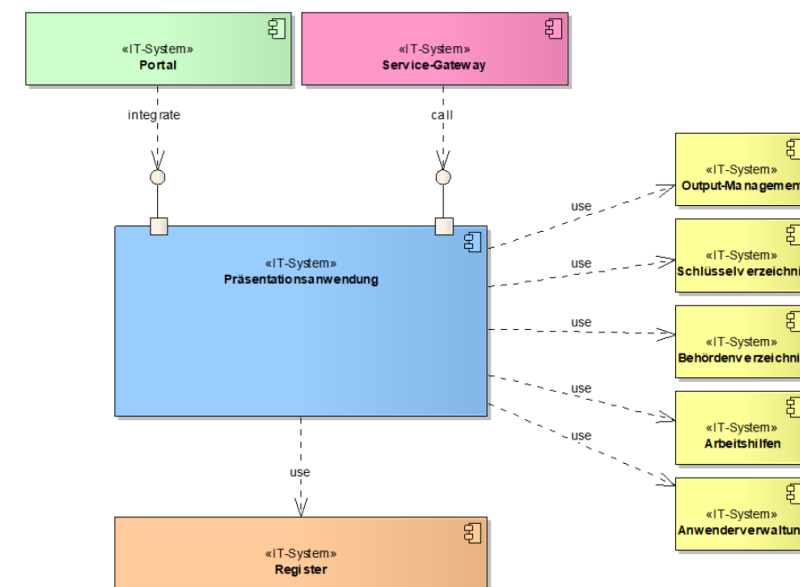
Vorgaben der Referenzarchitektur

- Spezifikation der Anwendungskomponenten einer Präsentationsanwendung und ihrer Anwendungsfälle:
 - Auskunft
 - Meldung
 - Mitteilungen
 - Datenpflege
 - Protokollierung
 - Fristenkontrolle

Checkliste: Einsatz der Referenzarchitektur

- ☒ Die bereitgestellten Dialoganteile einer Präsentationsanwendung wurden verwendet.
 - Registereintrag:
 - ☒ neu anlegen
 - ☒ anzeigen
 - ☒ editieren/korrigieren
 - ☒ löschen
- ☒ In der Präsentationsanwendung sollten nur solche Berechtigungsprüfungen implementiert sein, die direkte Auswirkungen auf die Nutzeroberfläche haben.

Architektur



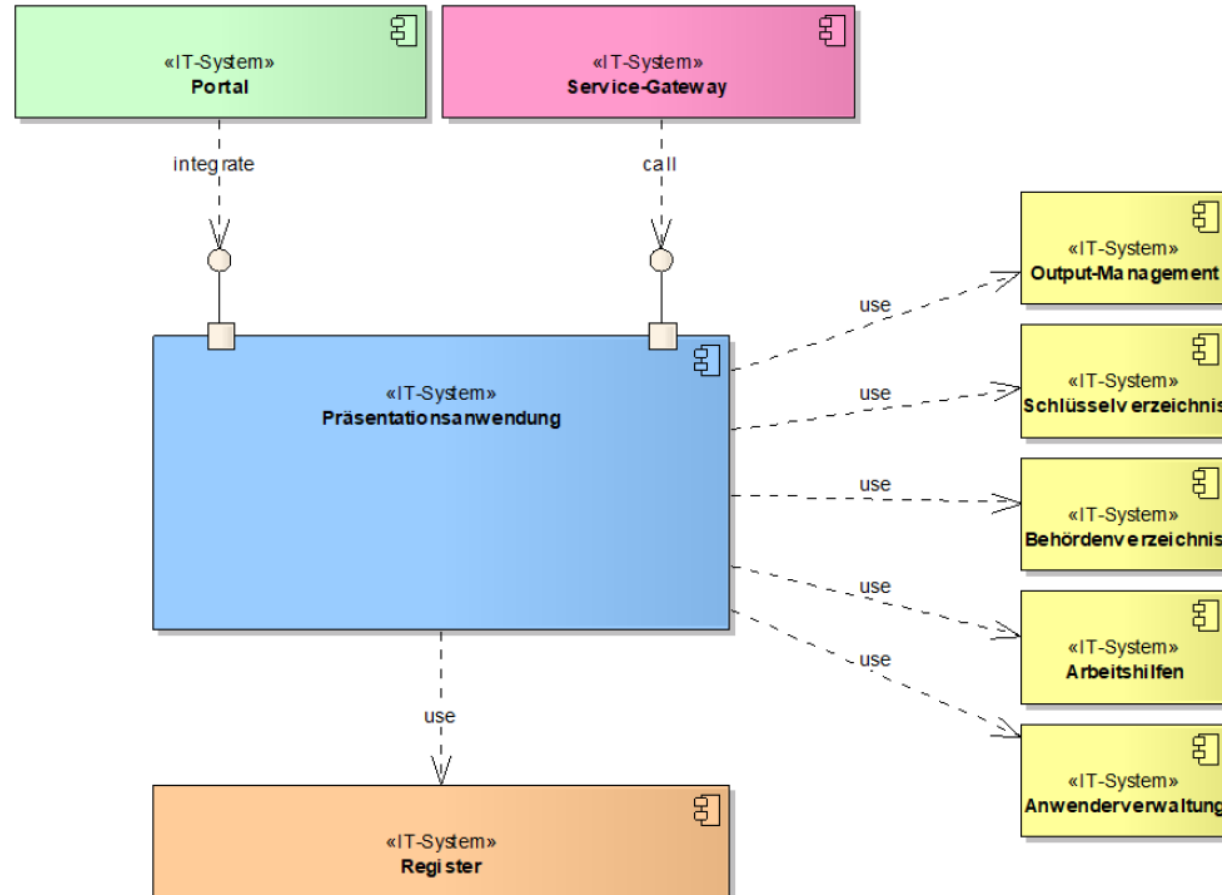
Zweck

- Allgemeine Beschreibung von Anwendungen, deren Hauptfunktion die Präsentation von Daten für einen menschlichen Benutzer ist.
- Präsentationsanwendungen haben in der Regel keine längeren GUI-Prozesse, sondern kurz dauernde Nutzertransaktionen, die auf technische Transaktionen abgebildet werden können.

Inhalte

- Eigenschaften einer Präsentationsanwendung und Einbettung in die Anwendungslandschaft
- Systementwurf
 - Außensicht
 - Innensicht
 - TI-Architektur

Architektur



Zweck

- Innerhalb von Beteiligungsanwendungen kommt es zur Beteiligung anderer Behörden, um benötigte Informationen zu erhalten oder einen Entscheidungsprozess zu ermöglichen.
- Die beteiligten Behörden treten als Dienstanbieter auf, die Kommunikationsaufgaben übernehmen und unterstützend weiterführende Dienstleistungen und Informationen für die anfragende Behörde bereitstellen.
- Die beteiligten Behörden formulieren in der Regel Rückmeldungen und übermitteln diese an die anfragende Behörde.

Inhalte

- Eigenschaften einer beteiligenden Anwendung
- Abgrenzung zu Register- und Präsentationsanwendungen
- Architekturbeschreibung einer beteiligenden Anwendung
- Beschreibung der Anwendungskomponenten

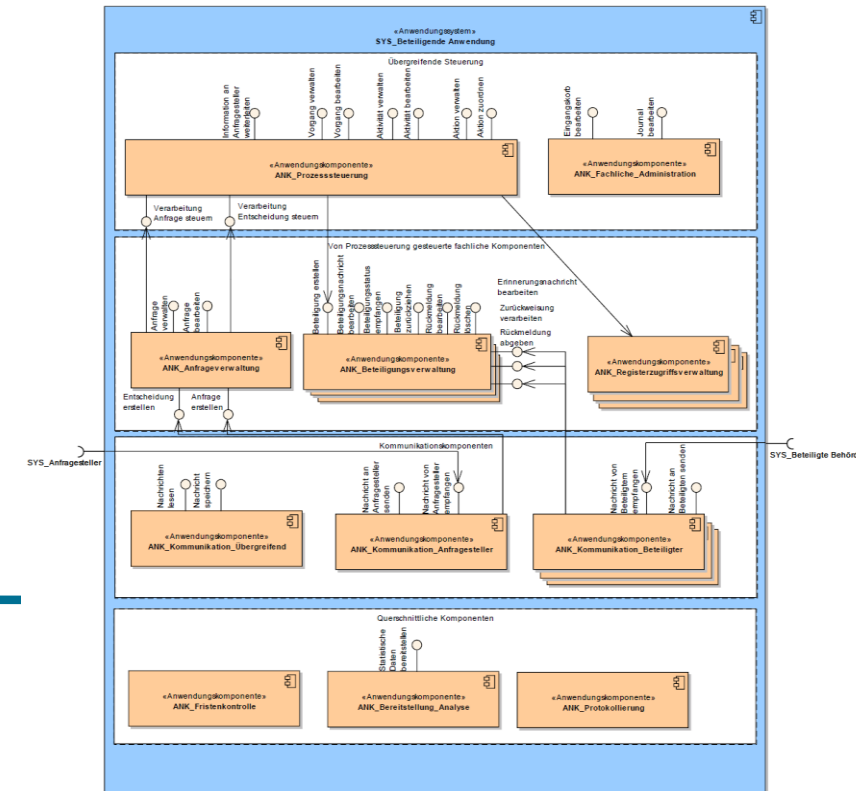
Vorgaben der Referenzarchitektur

- Entwurfsmuster für die Spezifikation:
 - Abläufe, Anwendungskomponenten, fachliches Datenmodell einer typischen Beteiligungsanwendung.
- Dialogablaufbeschreibung für:
 - Anfragersteller,
 - Beteiligte Behörde,
 - Sachbearbeiter,
 - Fachliche Administratoren.

Checkliste: Einsatz der Referenzarchitektur

- ✓ Konkretisierung der Abläufe, Anwendungskomponenten, fachliches Datenmodell müssen für die zu spezifizierende Beteiligungsanwendung.
- ✓ Konkretisierung der benötigten fachlichen und technischen Komponenten sowie der Schnittstellen für die zu entwerfende Beteiligungsanwendung.

Architektur



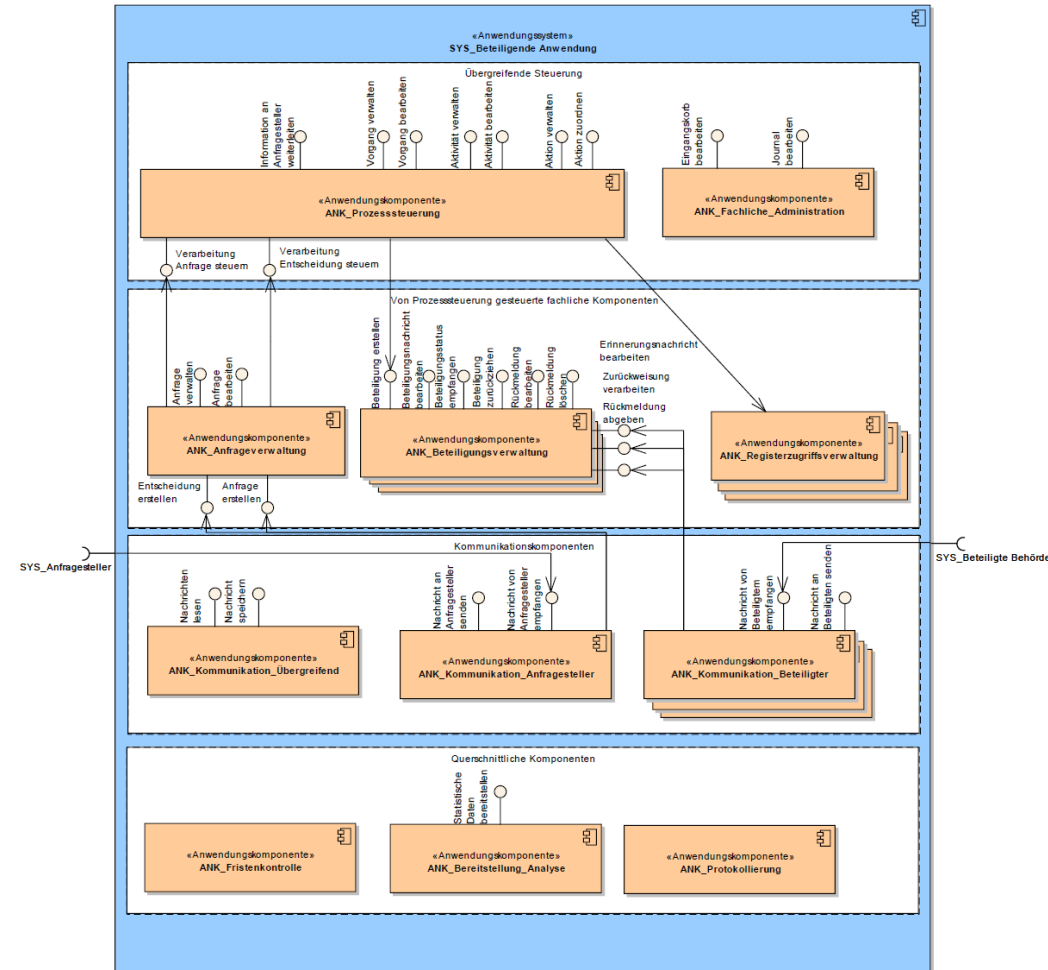
Zweck

- Innerhalb von Beteiligungsanwendungen kommt es zur Beteiligung anderer Behörden, um benötigte Informationen zu erhalten oder einen Entscheidungsprozess zu ermöglichen.
- Die beteiligten Behörden treten als Dienstanbieter auf, die Kommunikationsaufgaben übernehmen und unterstützend weiterführende Dienstleistungen und Informationen für die anfragende Behörde bereitstellen.
- Die beteiligten Behörden formulieren in der Regel Rückmeldungen und übermitteln diese an die anfragende Behörde.

Inhalte

- Eigenschaften einer beteiligenden Anwendung
- Abgrenzung zu Register- und Präsentationsanwendungen
- Architekturbeschreibung einer beteiligenden Anwendung
- Beschreibung der Anwendungskomponenten

Architektur



Zweck

- Dieses Konzeptsdokument beschreibt eine Anpassung des Zugriffswegs, die die Nutzung des Registerportals mit eigenem Zertifikat und Systembenutzer durch die Auftragsverarbeiter ermöglicht.

Typ

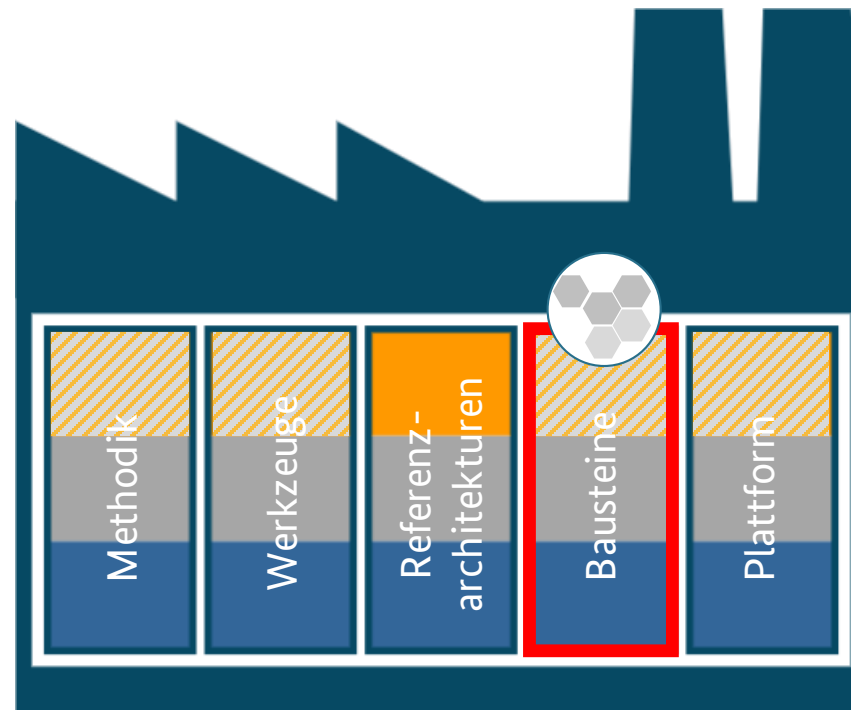


Konzept

Inhalte

- Erweiterung der Schnittstellenaufrufe um Parameter der durchführenden Behördengruppe und Behördenkennzeichen
- Protokollierung der Aufrufe
- Rolle Auftragsverarbeitung wird definiert

Bausteine





Fertige Lösungsbausteine für unterschiedliche Problemstellungen!



- Unabhängig einsetzbar
- Passend zur Anwendungsarchitektur der Referenzarchitekturen
- Für jeden Anwendungszweck kann eine Auswahl relevanter Bausteine erfolgen

Register Factory

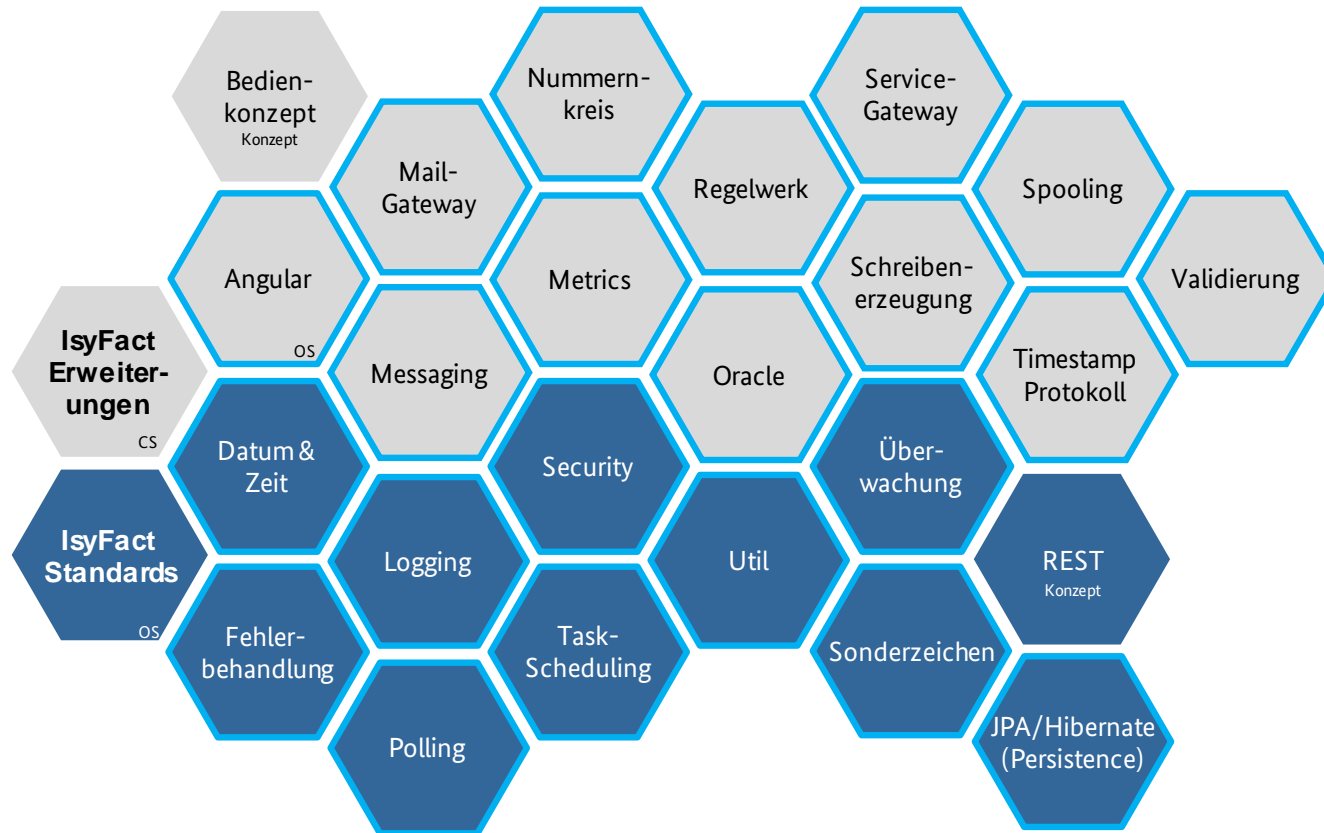
Register-spezifische Bausteine

IsyFact-Erweiterungen

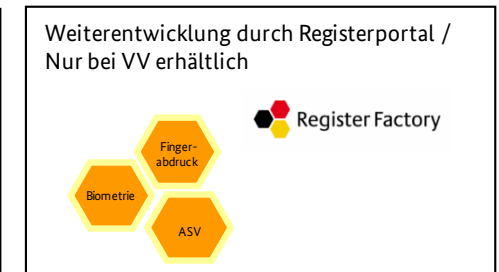
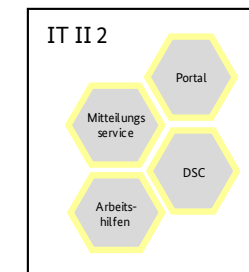
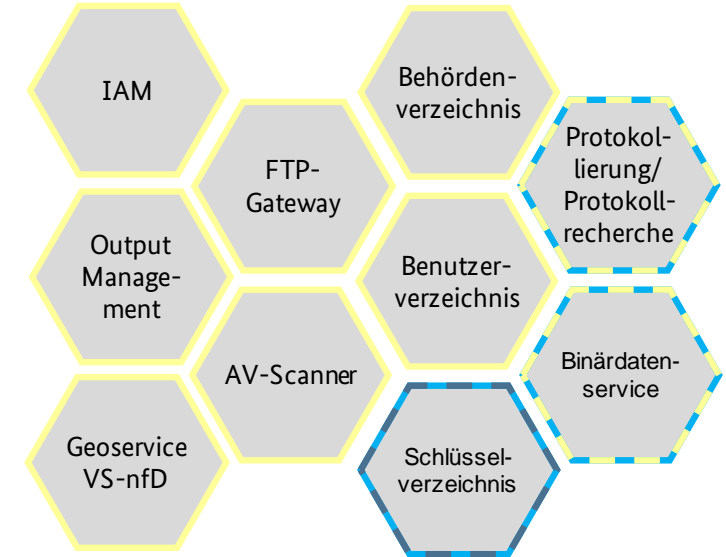
Allgemeine, fachneutrale Lösungen

IsyFact-Standards

Basiskomponenten für jedes IT-System



Querschnittsanwendungen basierend auf der IF



⬡ Eigenständiges IT-System
 ⬡ Software-Bibliothek
 ⬡ Mischform
 ⬡ Mischform wg. Client
 ⬢ Keine Weiterentwicklung



Eigenständige IT-Systeme

- Diese Bausteine werden einmal pro Anwendungslandschaft bereitgestellt (als „Querschnittsanwendungen“ (QA) innerhalb der „Querschnittsdomäne“).
- Sie sind in der Regel als Individualsoftware auf Basis der Referenzarchitektur umgesetzt.
- Anwendungen kommunizieren mit ihnen direkt über ihre Schnittstellen oder über Clients (spezielle Software-Bibliotheken).
- Kommunikation über bereitgestellte Clients
 - Einbinden der Client-JARs im Maven POM
 - Konfiguration des Schnittstellen-Endpunkts in der Konfiguration
- Direkte Kommunikation über die Schnittstellen: REST
 - Einbinden des jeweiligen Service-Bausteins im Maven POM
 - Konfiguration des Schnittstellen-Endpunkts in der Konfiguration
 - Entwicklung weiterer, benötigter Features: Adapter, Caching, Resilienz, ...

Software-Bibliothek

- Diese Bausteine sind Teil jeder Anwendung, die ihre Funktionalität benötigt.
- Sie behandeln in der Regel grundlegende technische oder fachliche Fragestellungen.
- Sie werden über Konfiguration in die jeweilige Anwendung integriert.
- Einbinden der JARs im Maven POM
- Konfiguration des Bausteins (bei Zusätzen oder Abweichungen zu den Voreinstellungen)



Eigenständige IT-Systeme

- Kompletter Satz an Dokumenten gemäß Referenzarchitektur
- Systemspezifikation (fachlich)
- Systementwurf (technisch / betrieblich)
- Systemhandbuch (betriebllich)
- ggf. begleitende Konzepte und Produktauswahlen



Systemspezifikation QA-XYZ:
Spezifische fachliche Architektur



Systementwurf QA-XYZ:
Spezifische technische und betriebliche Architektur



Systemhandbuch QA-XYZ:
Spezifische betriebliche Architektur

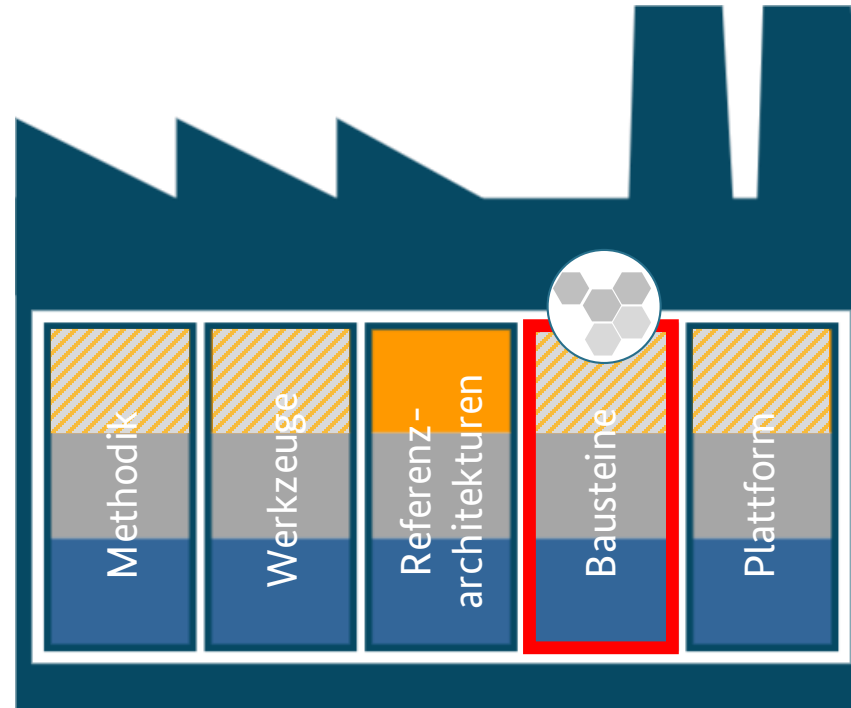
Software-Bibliothek

- Konzept: Theoretische Grundlagen
 - Grundlegende Anforderungen, Vorgaben, Rahmenbedingungen, Lösungsansätze...
- Nutzungsvorgaben: „Alles für den Einsatz“
 - Anwendung des Konzepts, Einsatz einer Bibliothek, Anbindung einer Querschnittsanwendung...



IsyFact- Standards

Bausteine



Zweck

- Vorgaben für die einheitliche Verwendung von Datums- und Zeitfunktionen
- Einheitliche Formatierung von Datums- und Zeitwerten und Zeiträumen
- Bereitstellung von Berechnungen auf Datums- und Zeitwerten und auf Zeiträumen

Inhalte

- Vordefinierte Ein- und Ausgabeformate
- Umwandlung in ISO 8601-konforme Darstellung für die Verwendung an Schnittstellen
- Umgang mit ungewissen Daten, bei denen Tag oder Tag und Monat nicht bekannt sind, oder die völlig unbekannt sind (z.B. entspricht xx.xx.2023 dem Zeitraum 01.01.-31.12.2023).
- Berechnungen u.a.: Sortierung, Vergleiche, Addition, Subtraktion, aufeinanderfolgende Tage, Werktage
- Testunterstützung: aktuelle Systemzeit mit Test-Zeitwerten überschreiben, um Abläufe in der Vergangenheit oder Zukunft testen zu können

Funktionsweise

- Ergänzungen zur Java Date and Time API
- Vorgaben zur Verwendung der Java Date and Time API

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Bibliothek.

Typ



Software-Bibliothek

Technische Basis

- Java Date and Time API

Zweck

- Effiziente und einheitliche Fehlerbehandlung
- Einfache Handhabung der Fehlerbehandlung in den Anwendungen
- Sicherstellen der Wartbarkeit der Fehlerbehandlung.
- Einsatz von Exceptions in standardisierter Form
- Bereitstellen angemessener Informationen im Fehlerfall

Inhalte

- Einheitliche Strukturierung und Implementierung durch Bereitstellung von Oberklassen
- Vorgaben zum Einsatz und der Behandlung von Exceptions
- Nutzungsvorgaben für Einsatz und Behandlung mit Code-Beispielen
- Konventionen für Fehlertexte/-nummern
- Dos und Don'ts für Entwickler

Funktionsweise

- Nutzung einer vorgegebenen abstrakten Fehlerhierarchie
- Ablage der Fehlertexte/-nummern gebündelt in Property-Dateien
- Variablen-Ersetzung in Fehlertexten mittels Spring-Mechanismen und eigenen Erweiterungen im Baustein Util

Checkliste: Einsatz des Bausteins

- ✓ Einbinden des Baustein-JARs
- ✓ Erweiterung der IsyFact-Exception-Hierarchie durch Vererbung um bis zu drei abstrakte Anwendungsfehlerklassen und entsprechende konkrete Fehlerklassen
- ✓ Erstellen einer Property-Datei für Fehlertexte
- ✓ Erstellen eines Fehlertextproviders zum Lesen der Fehlertexte -> Referenzimplementierung vorhanden

Typ

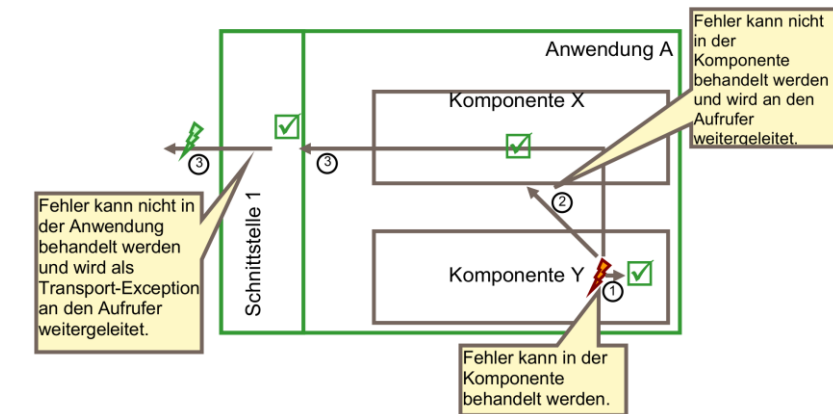


Software-Bibliothek

Technische Basis

- Java (Exception, RuntimeException)

Architektur



Zweck

- Der Baustein JPA/Hibernate ergänzt den Aufbau der Fachkomponenten aus der Referenzarchitektur um die Verwendung von Spring Data JPA.

Funktionsweise

- Der Baustein beschreibt die Umsetzung des Datenzugriffs auf Basis von JPA und dem O/R-Mapper Hibernate. Hierbei handelt es sich um die Möglichkeit, das Mapping zwischen einer Geschäftsanwendung und einer relationalen Datenbank zu automatisieren.

Typ



Software-Bibliothek

Technische Basis

- Spring Boot
- Liquibase

Inhalte

- Vorgaben zur Verwendung von JPA und Hibernate zum Datenzugriff
- Vorgaben zur Definition des O/R-Mappings
- Konventionen zur Erstellung von Datenbankschemas
- Umsetzung der Data Access Objects (DAO)
- Versionierung mit Liquibase

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Bibliothek

Zweck

- Einheitliche und standardisierte Log-Nachrichten
- Effiziente Log-Auswertung
- Nachvollziehbare Aufrufe bzw. Geschäftsvorfälle über Anwendungen hinweg
- Logging „so still wie möglich“
- Sicherstellung datenschutzkonformer Log-Nachrichten

Inhalte

- Vorgaben zu fachlichem & technischem Logging
- Fest definierte Log-Level, Log-Kategorien & Log-Schlüssel
- Verwendung von Ereignis- und Fehlerschlüsseln
- Verwendung von Korrelations-IDs zum Verfolgen einzelner Aufrufe durch eine Anwendungslandschaft
- Hinweise und Vorgaben zur Log-Auswertung, auch von externen Logs (z.B. Logs des Applikationsservers)
- Konfigurationsänderungen zur Laufzeit
- Vorkonfigurierter Appender

Funktionsweise

- Der Baustein unterstützt die Erstellung einheitlicher Log-Nachrichten über eine zentrale Schnittstelle.
- Log-Nachrichten werden automatisch durch relevante Informationen wie z.B. Klasse/Methode oder Korrelations-ID ergänzt.
- An System- und Komponentengrenzen werden Methodenaufrufe automatisch geloggt.
- Konfigurativ können auch Ausführungsdauer und Antworten von Methodenaufrufen geloggt werden.
- Log-Dateien im JSON-Format für direkte Verarbeitung durch Log-Infrastruktur.

Checkliste: Einsatz des Bausteins

- ☒ Definition eigener Ereignis- und Fehlerschlüsseln
- ☒ Konfiguration des Appenders
- ☒ Konfigurieren der Laufzeitüberwachung der Logging-Konfiguration inklusive Polling-Intervall
- ☒ Aufbau Infrastruktur zur Log-Auswertung

Typ

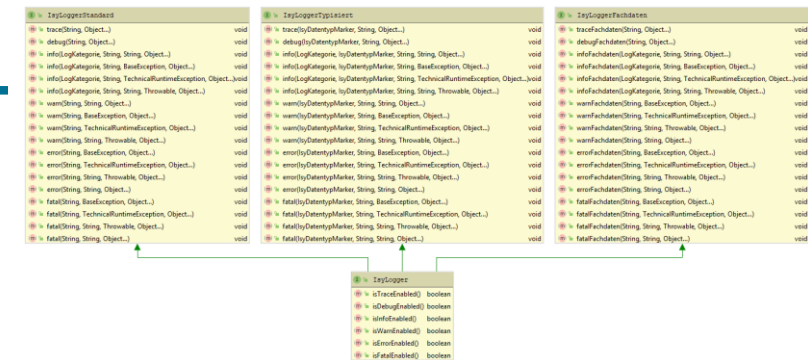


Software-Bibliothek

Technische Basis

- SLF4J (Abstraktion des Logging Frameworks)
- Logback (Logging Framework)
- Spring (I18N Support)

Architektur



Zweck

- Der Baustein ermöglicht, in regelmäßigen Intervallen über Schnittstellen neue Daten anderer Anwendungen zur Verarbeitung abzuholen.
- Die Abholung der Daten geschieht für eine hohe Ausfallsicherheit in mehreren, synchronisierten Knoten bzw. Instanzen.
- Der Baustein synchronisiert diese Knoten.

Inhalte

- Software-Bibliothek mit:
 - einer Schnittstelle zur Bereitstellung eines Zeitstempels der letzten Aktivität,
 - Mechanismen, um andere Knoten regelmäßig auf Aktivität hin zu überprüfen.

Funktionsweise

- Synchronisierungsmechanismus über JMX
- Bereitstellung des Zeitstempels der letzten Polling-Aktivität
- Automatisches Failover eines anderen Knotens, wenn ein bestimmter Zeitraum überschritten wird

Checkliste: Einsatz des Bausteins

- ☒ Konfiguration der benötigten Spring-Beans
- ☒ Anwendungskonfiguration gemäß Vorgaben
- ☒ Maßnahmen zur Erkennung von Duplikaten auf DB-Ebene getroffen
- ☒ Implementierung der Prüfung in der Polling-Aktivität oder Nutzung der Annotation bzw. des Interceptors

Typ

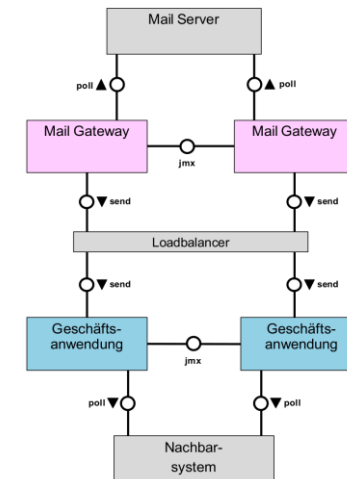


Software-Bibliothek

Technische Basis

- Java Management Extensions (JMX)
- Spring

Architektur



Zweck

- Authentifizierung und Autorisierung von Anwendern und Systemen.
- Bereitstellung von Anwenderinformationen innerhalb der Anwendung.
- Restriktive Vergabe von Berechtigungen auf Anwendungsebene.

Inhalte

- Vorgaben zur Spezifikation und zum Schnitt von Rollen und Berechtigungen
- Funktionen zur Authentifizierung von Anwendern und Systemen (Single-Sign-On und Single-Session).
- Funktionen zur Autorisierung von Anwendern (Benutzeroberfläche) und Systemen (Service-Schnittstellen, Batch)
- Nutzungsvorgaben zum Einsatz des Bausteins.
- Bibliothek „Security“

Funktionsweise

- Für die Authentifizierung und Autorisierung stellt der Baustein je ein Interface und einen Interceptor (Methoden-Annotation) bereit.
- Der verwendete IAM Service wird als OAuth2-konform vorausgesetzt. Spezielle Adapter sind nicht (mehr) erforderlich.
- Nach erfolgreicher Authentifizierung werden die Anwenderinformationen (u.a. Rollen) im Security-Kontext abgelegt.
- Die Rollen aus dem Security-Kontext werden in der Anwendung auf Anwendungs-spezifische Rechte abgebildet, die zur Autorisierung an Methoden dienen.

Checkliste: Einsatz des Bausteins

- ☒ Bibliothek „Security“ einbinden.
- ☒ Spring-Konfiguration erstellen.
- ☒ Konfigurieren der Rechte in rollenrechte.xml.
- ☒ Verwendung der @Secured Annotationen (mit entsprechenden Rechten) für die Berechtigungsprüfung in der Anwendung.

Typ

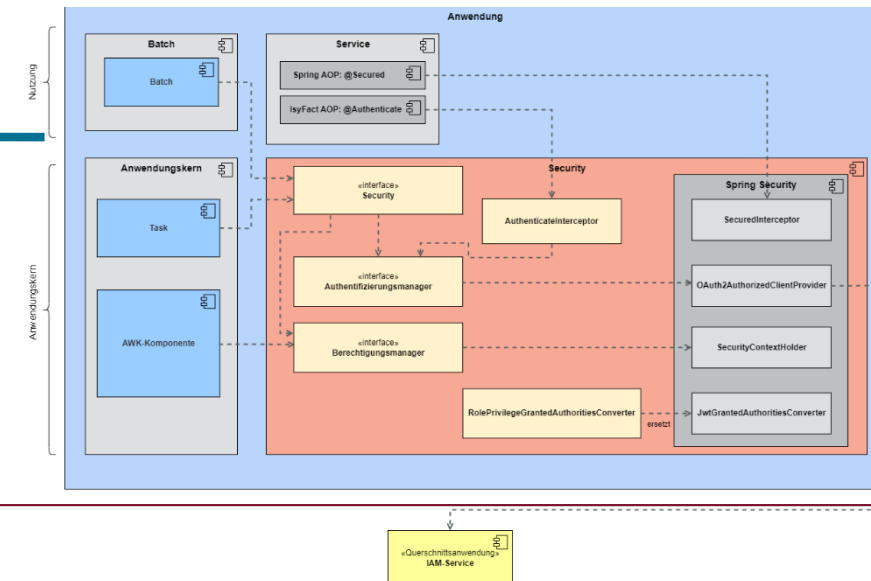


Software-Bibliothek

Technische Basis

- Spring Security
- Spring AOP

Architektur



Zweck

- Einheitlicher Umgang mit internationalen Sonderzeichen
- Unterstützung bei Konvertierung, Datenaustausch und Speicherung von Sonderzeichen
- Unterstützung des Zeichensatzes String.Latin 1.1 in der UTF-8-Codierung
- Unterstützung des Zeichensatzes DIN SPEC 91379 & DIN Norm 91379 in der UTF-8-Codierung

Inhalte

- Vorgaben für die einheitliche Konfiguration eines Zeichensatzes in der gesamten Anwendungslandschaft
 - Datenbank
 - IDE
 - JAVA JVM / Compiler
 - Maven
 - XML/HTML
- Bereitstellung verschiedener Transformatoren zur Konvertierung von Zeichen
- Transkription basierend auf ICAO Standard ICAO-MRTD

Funktionsweise

Transformatoren

- Transkription: Abbildung des DIN SPEC 91379 & DIN Norm 91379 Zeichensatzes in der UTF-8-Codierung, einer Teilmenge des Unicode v4.x Zeichensatzes, in den ASCII-Zeichensatz (phonetische Transformation).
- Suchform: Abbildung des DIN SPEC 91379 & DIN Norm 91379 Zeichensatzes auf die Grundbuchstaben A bis Z.
- Legacy: Abbildung des DIN SPEC 91379 & DIN Norm 91379 Zeichensatzes auf den Zeichensatz String.Latin 1.1.

Validatoren

- Datentyp: Überprüfung, ob eine Zeichenkette nur Zeichen eines bestimmten Datentyps der DIN SPEC 91379 & DIN Norm 91379 enthält.

Checkliste: Einsatz des Bausteins

- ☒ Durchgängige Konfiguration des UTF-8 Zeichensatzes
- ☒ Spring-Konfiguration der Transformatoren und Validatoren

Typ



Software-Bibliothek

Technische Basis

- String.Latin 1.1
- DIN SPEC 91379 & DIN Norm 91379

Architektur



Abbildung 1: Drei Zeichen, dargestellt in der Schriftart Helvetica, als Unicode Code Points und mit UTF-8 encodiert (hexadezimale Darstellung). Häufige Zeichen wie das A benötigen weniger Bytes im UTF-8 Encoding.

Zweck

- Planung und Ausführung von regelmäßigen Aufgaben (=Tasks) der Anwendungen
- Einheitlicher und vereinfachter Mechanismus für die Tasks bzgl.:
 - Umsetzung
 - Sicherheit
 - Konfiguration
 - Logging & Überwachung

Funktionsweise

- Task Scheduler zur Einplanung und Steuerung der Tasks.
- Abstrakte Klasse zur Definition von Tasks.
- Nutzung der jeweiligen IsyFact-Bausteine für die Aspekte Authentifizierung und Autorisierung, Logging, Konfiguration und Überwachung.

Typ



Software-Bibliothek

Technische Basis

- Java Concurrency API
- JMX

Inhalte

- Umsetzungshilfe für Tasks
- Konfiguration individuell für jeden Task, mit globalen Defaults (für Authentifizierung und Autorisierung sowie zur Überprüfung, ob der Task auf einem bestimmten Host ausgeführt wird)
- Beispielkonfiguration
- Direkte Kontrolle über Task Scheduler
- Vorgefertigte MBeans zur Überwachung der Tasks

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Bibliothek
- ☒ Implementieren der Tasks
- ☒ Konfigurieren der Tasks
- ☒ Erstellen der Spring-Konfiguration für die Tasks
- ☒ Erstellen der Spring-Konfiguration für die Überwachung mittels JMX

Zweck

- Die Bibliothek isy-util bietet nützliche Hilfsmittel, die von den Anwendungen der IsyFact genutzt werden können. Es handelt sich dabei um kleinere Utility-Klassen, welche die Implementierung vereinfachen.

Funktionsweise

- Bereitstellung Standard-Klassen und Werkzeuge

Typ



Software-Bibliothek

Technische Basis

- Java

Inhalte

- Common: Package enthält allgemeine Klassen
- Package exception: Klassen, die für die Fehlerbehandlung einzusetzen sind
- Package spring: enthält Werkzeuge für den Umgang mit Spring

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Bibliothek

Zweck Funktionsweise Typ

- Bereitstellung einheitlicher Administrationsschnittstellen zur einfachen Überwachung von Anwendungen.
- Unterstützung des Betriebs hinsichtlich:
 - schnellerer Problemeskalation und Reaktion,
 - vorausschauender Ressourcenplanung bzw. -erweiterung,
 - Einsatz einer einheitlichen, vorgefertigten Überwachungskomponente.

- Verwendung von Spring Boot Actuator
 - Micrometer zur Bereitstellung von Metriken
 - Wahl eines geeigneten Monitoring-Systems ohne feste Vorgaben (z.B. Prometheus)



Software-Bibliothek

Technische Basis

- Spring Boot Actuator
- Micrometer

Inhalte Checkliste: Einsatz des Bausteins

- Bereitstellen definierter:
 - Überwachungsinformationen,
 - Informationen über Services.
- Prüfung der Verfügbarkeit benötigter Nachbarsysteme
- Deaktivierbarkeit von Anwendungen

- ☒ Definition weiterer Metriken zur Überwachung von Anwendungen.
- ☒ Einbinden der Bibliothek zur Ermittlung der Metriken.
- ☒ Erstellen/Setzen der Konfiguration
 - ☒ Spring-Konfiguration für Spring-Mbean-Exporter,
 - ☒ Metric Endpoints,
 - ☒ Timer-Task.
- ☒ Konfiguration der Actuator Endpoints

Zweck Funktionsweise Typ

- Einheitliche interne und externe Service-Kommunikation auf Basis von REST.
- Ablösung der HTTP-Invoker Technologie und des gleichnamigen IsyFact-Bausteins.
- Einführung von OpenAPI3 und Code-Generatoren.

- Der Baustein stellt keine Software-Bibliothek, sondern Vorgaben und Implementierungshinweise bereit.



Konzept/Nutzungsvorgaben

Inhalte Checkliste: Einsatz des Bausteins Technische Basis

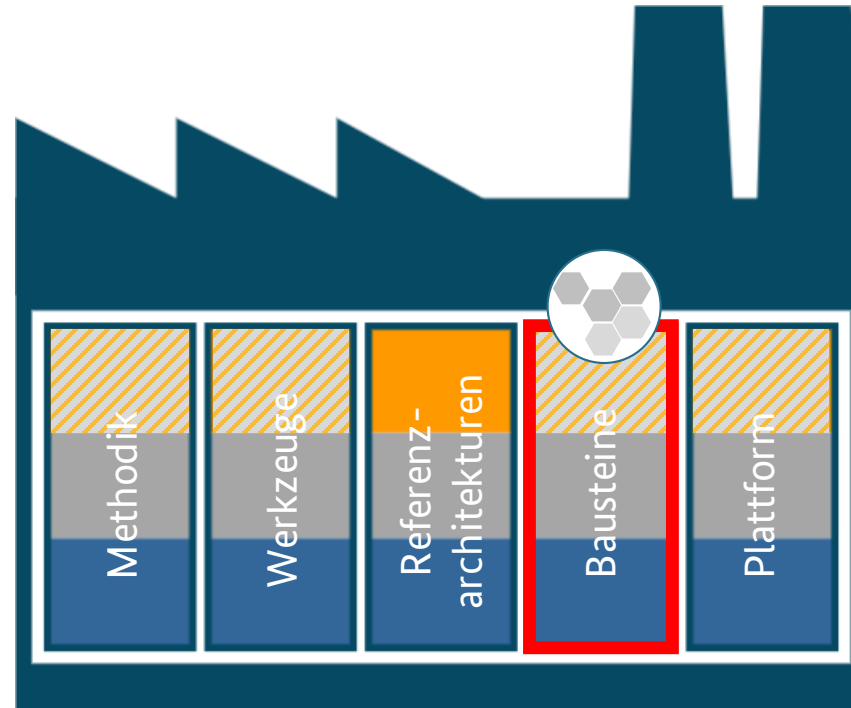
- Vorgaben zu Frameworks zur Implementierung von REST-Clients und REST-Servern
- Vorgaben zur Spezifikation und Dokumentation von REST-Schnittstellen (Contract-First)
- Vorgaben zur Generierung von REST-Clients und REST-Webservices
- Vorgaben zu querschnittlichen Aspekten (Sicherheit, Logging, Fehlerbehandlung, Versionierung)
- OAuth2.0 zur Autorisierung

- ☒ OpenAPI3 Spezifikation ist auf Grundlage der Systemspezifikation erstellt.

- Spring WebClient (Client Implementierung)
- Spring Web MVC (Server Implementierung)
- OpenAPI 3 als Dokumentationsstandard
- OpenAPI Generator zur Generierung von Client- und Server-Implementierungen

IsyFact- Erweiterungen

Bausteine



Zweck

- Erleichtert den Bau von webbasierten Benutzeroberflächen durch die Bereitstellung von spezifischen Widgets und Methoden speziell für die öffentliche Verwaltung
- Framework zur effektiven Entwicklung von Web-Front-Ends mit Angular und TypeScript
- Das konfigurierte PrimeNG Theming + Bedienkonzept ergibt die Gestaltungsbasis

Funktionsweise

- Der Baustein wird als npm Paket in einem beliebigen Angular Projekt installiert
- Die behördenspezifischen Widgets können als Module importiert und für die Erstellung von Benutzeroberflächen verwendet werden
- Zusätzliche Methoden und Interceptors bieten querschnittliche Funktionalität wie bspw. das Ergänzen von Tracing-Informationen

Typ



Software-Bibliothek

Inhalte

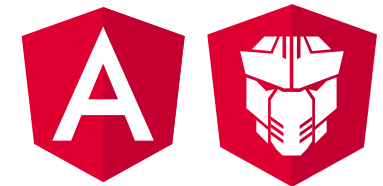
- Hauptfenster Widget (Applikationsrahmen)
- Wizard Widget
- Widgets für die Eingabe von speziellen Zeichen und unvollständigen Datumswerten
- HTTP-Interceptors für Tracing Header
- Security Funktionalität für Zugriffsbeschränkungen nach vergebenen Benutzerrechten
- Methoden zur Validierung von Benutzereingaben

Checkliste: Einsatz des Bausteins

- ☒ Vorhandensein von Node.js und npm
- ☒ Installation des Angular CLI

Technische Basis

- isy-angular-widgets
- TypeScript
- Angular
- PrimeNG



Zweck

- Der Baustein Bedienkonzept ermöglicht die einheitliche Umsetzung von Web-Oberflächen für Anwendungen jeder Art.

Funktionsweise

- Der Baustein Bedienkonzept ergänzt den Baustein Angular um die entsprechenden visuellen Darstellungsdefinitionen, basierend auf dem PrimeNG-Framework.
- Die Stylesheets können sowohl in klassischen Webanwendungen über Maven, als auch in Angular Projekten über npm verwendet werden.

Typ



Konzept

Technische Basis

- Angular

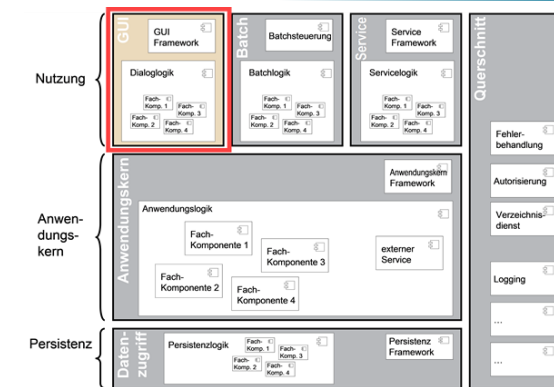
Inhalte

- Masken-Layoutvorgaben für Anwendungen
- Gestaltungsvorgaben zu visuellen Maskenelementen (z.B. Eingabefelder, Menüs, Fehlermeldungen, modalen und non-Modalen Dialogen, ...)
- Browserübergreifende Stylesheet-Definitionen
- Bereitstellung von Icons und Fonts

Checkliste: Einsatz des Bausteins

- ☒ Konfiguration des Standard-Farbschemas.

Architektur



Zweck

- Versand und Empfang von Mails
- Anbindung bestehender Service-Gateways
- Asynchrone Kommunikation mit externen Anwendungen über SMTP

Inhalte

- Mailabholungsschnittstelle zum Abruf eingehender Mails
- Adapterkonzept zur automatisierten Verarbeitung eingehender Mails
- Komponente zum Erzeugen und Versenden von Mails
- Prüfroutine zur Überwachung des Gateways
- Basisklasse zur Wandlung einer SOAP-Mail in einen Web-Service-Aufruf (HTTP)

Funktionsweise

- Abruf der Mails eines Postfach-Ordners
- Weiterleitung an Anwendungen
- Versand von Mails aus Anwendungen an Mailserver
- Bereitstellung von Schnittstellen zum Abholen und Versenden von Standard-Mails

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Client-Komponente in die nutzende Anwendung
- ☒ Implementieren des Mailgateways inkl. Schnittstellen
- ☒ Setzen der Konfigurationsparameter

Technische Basis

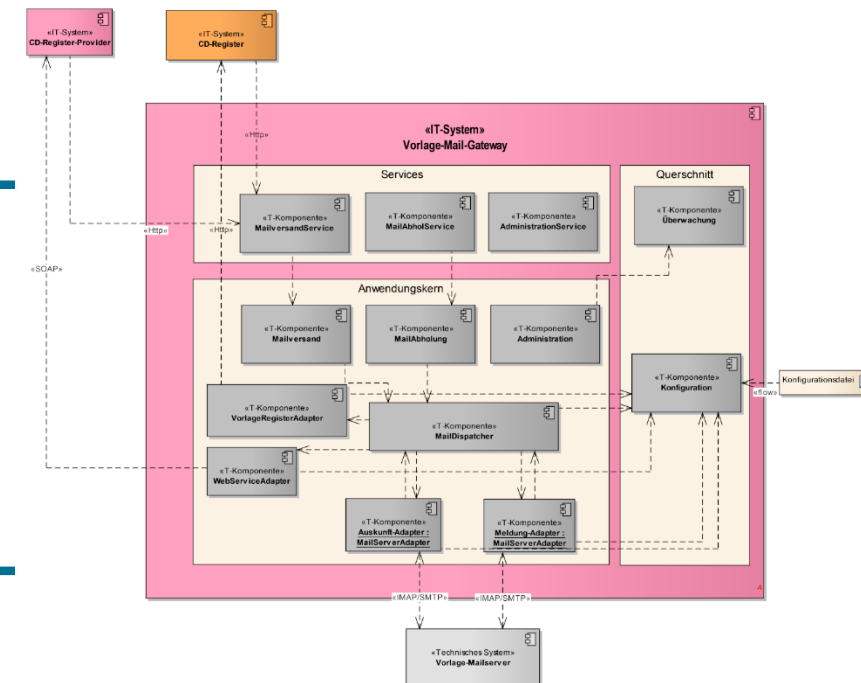
- IsyFact **V2**
- IMAP & SMTP

Typ



Software-Bibliothek

Architektur



Zweck

- Dieses Modul stellt Annotationen und Klassen zur Verfügung um basierend auf Kafka eine asynchrone Kommunikation zwischen Anwendungen einer Anwendungslandschaft zu implementieren
- Die wichtigsten Ziele hierbei sind:
 - Resilienz
 - Performance
 - Skalierbarkeit und
 - Wartbarkeit

Funktionsweise

- Bei Kafka handelt es sich um ein Produkt, das zwei Messaging-Modelle („Queuing“ und „Publish-Subscribe“) miteinander kombiniert.
 - Eine Queue ist eine lineare Datenstruktur, bei der neue Elemente von einer "Seite" eingefügt und von der anderen „Seite“ abgerufen werden.
 - Das Publish- / Subscribe-Pattern ermöglicht es, dass viele verschiedene Anwendungen auf einem Topic, unabhängig von anderen Anwendungen, konsumieren können.

Typ



Software-Bibliothek

Inhalte

- Der Messaging-Baustein bietet
 - Logging
 - IsyProducer Bean zum einfachen Erstellen von KafkaTemplates
 - IsyConsumer Bean zum einfachen Erstellen von ConcurrentKafkaListenerContainerFactories
 - Multi-Cluster / Multi-Config Support
 - erweiterte Prometheus Metriken und
 - Health-Endpunkt

Zweck

- Dieses Modul stellt die Annotation @Metrik zur Verfügung, diese nimmt automatisch Messungen vor:
 - Anzahl der Aufrufe
 - Summe der Aufrufzeiten
- Die Daten werden automatisch um CPU-Last und Arbeitsspeicherauslastung erweitert.

Funktionsweise

- Das Grafana-Dashboard wird vorkonfiguriert mitgeliefert
- Bereitstellung der Daten erfolgt über Micrometer und die Daten können mit Prometheus weiterverarbeitet werden
- Die Daten werden gesammelt und in regelmäßigen Abständen an Prometheus gesendet und über ein Grafana-Dashboard angezeigt

Typ



Software-Bibliothek

Zweck

- Bereitstellung eindeutiger Nummern für Nummernkreise der fachlichen Anwendung.
- Generierung eindeutiger, fortlaufender Nummern in einem bestimmten Zeitfenster (z.B. täglich).

Funktionsweise

- Erlaubt Reservierung von zeitbasierten oder unbegrenzt gültigen fortlaufenden Nummern.
- Nummernkreise können zu bestimmten Zeitpunkten (bspw. bei Tageswechsel) zurückgesetzt werden.
- Datenhaltung gemäß Referenzarchitektur Datenzugriff.

Typ



Software-Bibliothek

Inhalte

- Software-Bibliothek mit Spring-Bean zur Einbindung in die Anwendung und Erzeugung der Nummern.
- Datenmodell zur Verwaltung und Speicherung der Nummernkreise.

Checkliste: Einsatz des Bausteins

- ☒ Bibliothek einbinden.
- ☒ Datenbanktabellen erzeugen.
- ☒ Spring-Bean für Nummernkreise konfiguriert.
- ☒ Eigene Nummernkreise konfigurieren.

Technische Basis

- JPA/Hibernate

Zweck

- Die Nutzung der Oracle Database einheitlicher und einfacher zu gestalten.
- Erfahrungen für alle Nutzer verfügbar machen.
- Fallstricke für alle Nutzer zu vermeiden.

Funktionsweise

- Bereitstellung von Schemas und Konfigurationen.

Typ



Software-Bibliothek

Inhalte

- Definition des Datenbankschemas
- Verwendung des Universal Connection Pools (UCP)
- Query Optimierung

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Bibliothek

Technische Basis

- Oracle
- Java

Zweck

- Erstellung fachlich komplexer, änderbarer Regelwerke zur:
 - Validierung von Meldungen und Abgleiche gegenüber den Bestandsdaten,
 - Berechtigungs- und Sichtbarkeitsprüfung,
 - Steuerung fachlicher Prozesse,
- Der Baustein erleichtert die Einbindung von JBoss Drools.

Inhalte

- Vorlagen (Excel-Sheets) zur Erstellung von Regelbanken durch den Fachbereich
- Baustein zur Verwaltung und Ausführung von Regelbanken, zur Integration in die IT-Systeme

Funktionsweise

- Erstellung fachlich komplexer, änderbarer Regelwerke (z.B. für Register).
- Regelbanken besitzen eine fachliche und eine technische Sicht:
 - fachlich: Erstellung und Bearbeitung durch den Fachbereich
 - technisch: Ausführung durch das IT-System
- Änderungen an Regelbanken führen bis auf Ausnahmen nicht zur Anpassung des IT-Systems.

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Bibliothek zum Regelwerk
- ☒ Erstellen der Spring-Konfiguration
- ☒ Erstellen eigener Regelbanken auf Basis der Vorlagen

Typ



Software-Bibliothek

Technische Basis

- JBoss Drools

Architektur

	RuleSet	rulepkg2
	Variables	Validierungsergebnis ergebnis
RuleTable Validierung		
	CONDITION	ACTION
	...	
	...	ergebnis.erzeugeFehler(\$param)
	Prüfung	Fehlermeldung
Regelname	...	HT-020

Zweck

- Leichtgewichtige Erzeugung von Word- und PDF-Dokumenten aus Word-Vorlagen
- Strukturierte, einheitliche Befüllung der Word-Vorlagen mittels Platzhaltern

Funktionsweise

- Platzhalter werden durch die Velocity Template Language definiert und aufgelöst
- Schreiben werden mit XDocReport erzeugt

Typ



Software-Bibliothek

Technische Basis

- Velocity
- XDocReport

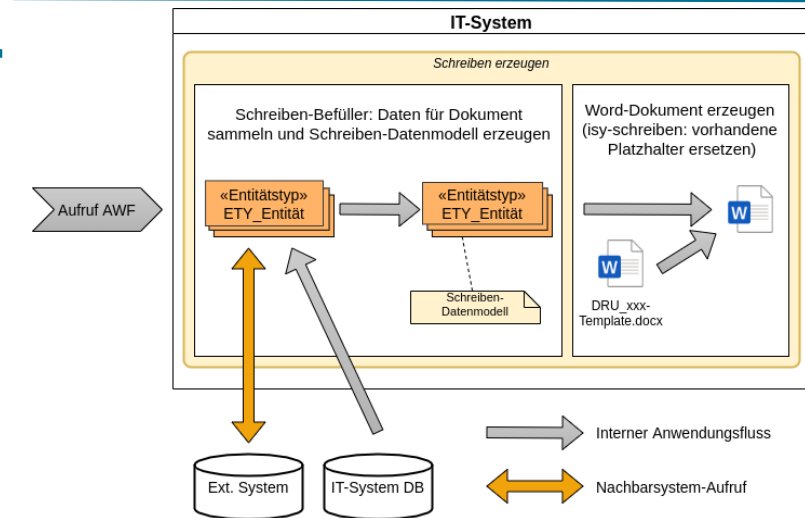
Architektur

Inhalte

- Datenmodelle zur Befüllung der Vorlagen
- Komponente zur Verwaltung der Vorlagen sowie zur Erzeugung der Dokumente

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Bibliothek
- ☒ Erstellung einer Spring-Konfiguration
- ☒ Erstellung eigener Wordvorlagen anhand der Datenmodelle und Platzhalter



Zweck

- Bereitstellung von Services der Anwendungslandschaft für externe Nutzer (Service-Provider).
- Bereitstellen von externen Services innerhalb der Anwendungslandschaft (Service-Consumer).
- Authentifizierung und Autorisierung von eingehenden Anfragen.

Inhalte

- Generator für REST- und SOAP-Service-Provider
- Generator für Client-Implementierung
- Authentifizierung und Autorisierung
- Mapping von Service-Daten in Transportobjekte.

Funktionsweise

- Kommunikation mit den Geschäftsanwendungen über REST
- Anbindung an den IAM-Service über den Baustein Security, um Anfragen zu authentifizieren.
- Keine Implementierung von Geschäftslogik

Checkliste: Einsatz des Bausteins

- ✓ Generieren eines Service-Gateways nach vorgegebener Standard-Architektur.
- ✓ Prüfen der Abhängigkeiten in den POM-Dateien und ggf. durchführen von Anpassungen.
- ✓ Prüfen der Spring-Konfiguration.
- ✓ Definieren der Fehlerwandlung.
- ✓ Vervollständigen der Service-Adapter-Klassen.
- ✓ Anbindung an den IAM-Service.

Typ

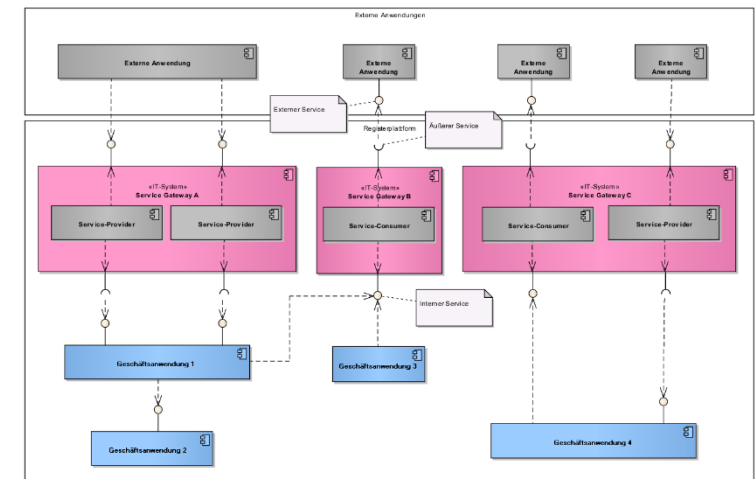


Software-Bibliothek

Technische Basis

- Spring MVC (REST)

Architektur



Zweck

- Zwischenpuffern von Aufrufen und Abarbeiten zu einem späteren Zeitpunkt (Spooling/Queueing) für asynchrone Aufrufe.
 - Asynchron bedeutet dabei, dass der Aufruf entweder keine Rückmeldung erfordert oder dass diese Rückmeldung zu einem späteren Zeitpunkt in einem separaten Aufruf erfolgen kann.

Funktionsweise

- JMS-konforme Verwendung von Oracle AQ
 - Keine Nutzung von Oracle AQ Erweiterungen
- Verwendung von Spring-JMS
- Verwendung des OCI-Treibers
 - Oracle AQ ist nicht nutzbar mit dem THIN-Treiber.

Typ



Software-Bibliothek

Technische Basis

- JMS
- Oracle AQ
- JMX

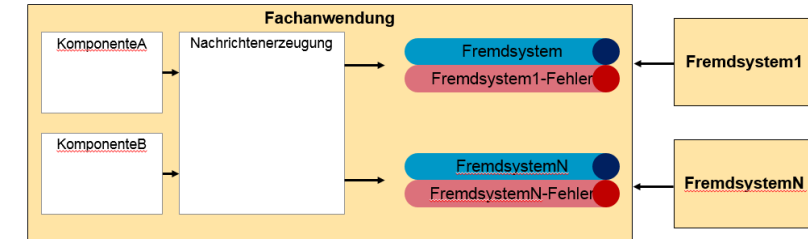
Inhalte

- Das Spooling wird über eine Queue umgesetzt, die über das Produkt „Oracle AQ“ bereitgestellt wird. Auf die Queue wird nach JMS-Standard zugegriffen.
- Administration von Queues mit Oracle AQ
- Einbinden der Queues mit Spring
- Überwachen der Queues mit JMX

Checkliste: Einsatz des Bausteins

- ☒ Konfiguration der JMS-Anbindung
- ☒ Konfiguration des OCI-Treibers
- ☒ Konfiguration der Queues mit Oracle AQ

Architektur



Zweck

- Client für den Zugriff auf die Time Stamp Authority (TSA), die RFC 3161 umsetzt

Funktionsweise

- Erzeugung Hashwert aus beliebigem String (Dokument)
- Signatur des Hashwerts mit Zeitstempel (über TSA)
- Prüfen der Signatur für eingegebenen Hashwert mit X509-Zertifikat

Typ



Software-Bibliothek

Inhalte

- Hash Generator
- Time Stamp Authority Client

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Bibliothek

Technische Basis

- Bouncy Castle Generators/Processors for Time Stamp Protocol (TSP)
- Java Security

Zweck

- Bean Validierung gemäß Bean Validation JSR 303 (1.0) und JSR 349 (1.1)
- Einfache und einheitliche Nutzung der Bean Validation

Funktionsweise

- Attribute der Datenobjekte (Beans) werden mit Annotationen versehen, die die Prüfungen angeben
- Validatoren werten die Prüfungen aus und erstellen Fehlermeldungen, wenn die Validierungen fehlschlagen

Typ



Software-Bibliothek

Inhalte

- Validierung von Eingaben von Benutzern oder Systemen
- Bereitstellung einer abstrakten Validator-Implementierung
- Bereitstellung von Default-Validatoren
- Bereitstellung des Hibernate Validators
- Erweiterung durch eigene Constraints und Constraint-Validatoren möglich

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der entsprechenden Bibliothek
- ☒ Konvertieren der Constraint-Violations zu eigenen Fehlertypen implementieren
- ☒ Konfigurieren der Fehlertexte
- ☒ Erstellen der Property-Dateien für Fehlertexte für Constraint-Violations (sowie nach Bedarf sprechende Namen für Klassen oder Attribute vergeben)
- ☒ Annotationen für zu validierende Attribute einfügen und Aufruf Validierungsmethode des gewählten Validators

Technische Basis

- Bean Validation
- Hibernate Validator

IsyFact- Erweiterungen

Querschnittsanwendungen- Details

Zweck

- Prüfung von Binärdaten, insbesondere von außerhalb der Anwendungslandschaft stammende, auf Virenbefall.
- Regelmäßige Prüfung von in der Anwendungslandschaft gespeicherten Binärdaten auf Virenbefall.

Funktionsweise

- REST Schnittstelle zur Annahme von Daten
- Übergabe der Daten via ICAP an den Virensan

Typ



Querschnittsanwendung (QA)

Technische Basis

- Virenschutzprodukt
- IsyFact 3

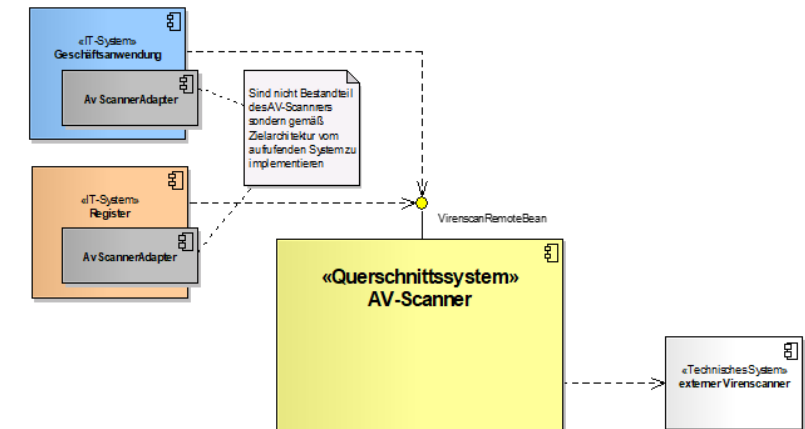
Inhalte

- Herstellerneutraler und versionsunabhängiger Zugriff auf das Virenschutz-Produkt

Checkliste: Einsatz des Bausteins

- ☒ Eigenen Adapters gemäß Vorlage implementieren.
- ☒ Spring-Konfiguration des Adapters erstellen.
- ☒ Spring-Konfiguration des Services des AV-Scanners erstellen.

Architektur



Zweck

- Zentrale Datenhaltung von Behördendaten für alle Anwendungen der Anwendungslandschaft.
- Beschränkung der Datenhoheit auf ein System.
- Zentraler Querschnittsservice.

Funktionsweise

- REST-basierte Schnittstelle für Anwendungen

Typ



Querschnittsanwendung (QA)

Technische Basis

- IsyFact 3

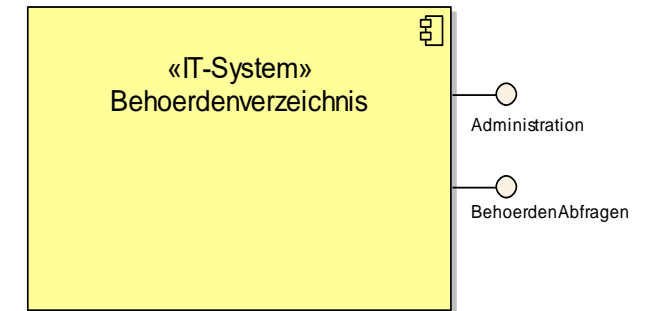
Inhalte

- Datenschema für Behörden und Behördengruppen.
- Verwaltung allgemein gültiger Behördendaten.
- Verwaltung verfahrensspezifischer Behördendaten in einem generisch erweiterbaren Datenschema.

Checkliste: Einsatz des Bausteins

- ☒ Konfiguration eines erweiterten Datenschemas für eigene Behördendaten (optional)
- ☒ Konfigurieren der Komponente und des Caches.
- ☒ Implementieren des Zugriffs auf Behördendaten aus dem nutzenden Anwendungskern.
- ☒ Zugriff auf verfahrensspezifische Behördendaten

Architektur



Zweck

- Zentrale Datenhaltung von Benutzerinformationen für eine Anwendungslandschaft.
- Verwaltung von menschlichen und technischen Nutzern
- Einheitliche Rollenzuweisung.
- Datenhaltung für den IAM-Service.
- Zentraler Querschnittsservice.

Funktionsweise

- REST-basierte Schnittstelle für Anwendungen
Integration in IAM-Service über Service-Schnittstelle (REST) oder Datenbankschnittstelle.

Typ



Querschnittsanwendung (QA)

Technische Basis

- IsyFact 3

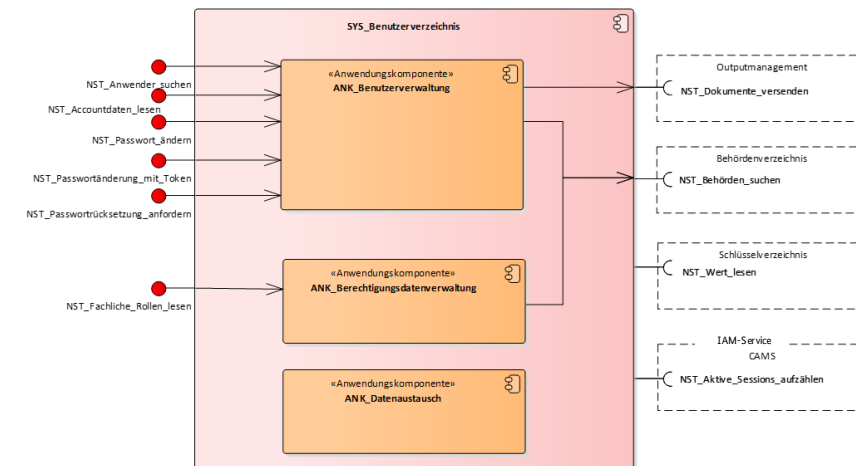
Inhalte

- Implementiert das Nutzergruppen- und rollenbasierte Datenmodell entsprechend Berechtigungskonzept.
- GUI-Komponente zur Verwaltung der Benutzer
- Schnittstellen zum Zugriff auf Benutzerstammdaten und Anwendersuche.
- Schnittstelle zur Selbstadministration des eigenen Benutzers (Passwörterneuerung).
- Datenimport für Rollen und Benutzer.
- Fristendienst für nicht mehr aktive Benutzer.

Checkliste: Einsatz des Bausteins

- ☒ Einrichten des Datenbankschemas und Import von Benutzerdaten
- ☒ Konfigurieren der Komponente und des Caches.
- ☒ Implementieren des Zugriffs auf Benutzerdaten aus dem nutzenden Anwendungskern.
- ☒ Falls gewünscht, Einbindung als Datenhaltung für IAM-Service (z.B. Baustein IAM)

Architektur



Zweck

- Produktunabhängige Verwaltung von Binärdateien.

Funktionsweise

- REST-basierte Service-Schnittstelle für Anwendungen (durch Client-Komponente vollständig transparent)
- Strukturierte Ablage der Binärdaten im Dateisystem

Typ



Binärdatenaustausch & Binärdatenservice Client: Software-Bibliothek



Binärdatenservice: Querschnittsanwendung (QA)

Inhalte

- Funktionalität zum Speichern und Lesen von Binärdaten
- Konfigurierbare, voneinander getrennte Ablageorte für Binärdaten
- Skalierbare Anwendung für die Ablage großer Mengen an Binärdaten
- Client-Komponente zur:
 - Kommunikation mit dem Binärdatenservice,
 - Pufferung von Binärdaten bei Nichtverfügbarkeit des Services.

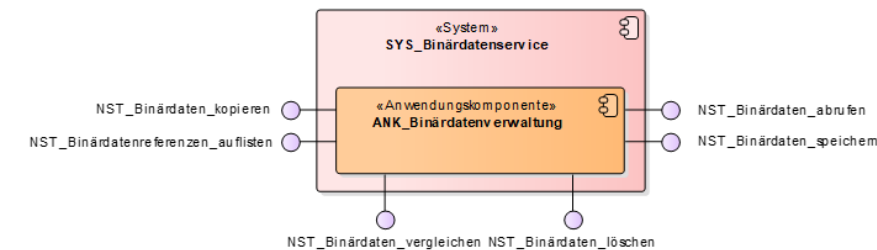
Checkliste: Einsatz des Bausteins

- ☒ Ablageort für die Binärdaten der Anwendungen konfiguriert.
- ☒ Client-Komponente eingebunden und eingerichtet.

Technische Basis

- IsyFact 3
- REST

Architektur



Zweck

- Ermöglicht Anwendungen ohne externen Netzzugriff das Herunterladen von Dateien von einem externen FTP-Server.
- Stellt Dateien von einem externen FTP-Server bereit.

Funktionsweise

- Lädt Dateien von einem externen FTP-Server herunter und stellt sie dem Aufrufer zur Verfügung.
- Service-Schnittstelle nach Referenzarchitektur „Anbieten interner Services“ (REST).

Typ



Querschnittsanwendung (QA)

Technische Basis

- (coming soon) IsyFact 3

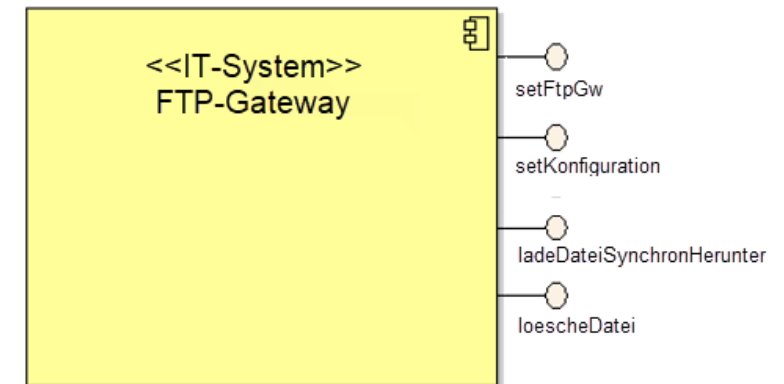
Inhalte

- Zentraler Querschnittsservice zum Ansprechen externer FTP-Server
- Konfigurierbare Allowlist für FTP-Server

Checkliste: Einsatz des Bausteins

- ☒ Implementierung eines technischen Adapters
- ☒ Spring-Konfiguration des Adapters und der Schnittstelle
- ☒ Externer FTP-Server konfigurieren.
- ☒ Kopieren der empfangenen Dateien sollte aufgrund der Größe vermieden werden

Architektur



Zweck

- Bereitstellung eines zentralen IAM-Services (IAM = **I**ntity and **A**ccess **M**anagement) zur Authentifizierung von externen und internen Benutzern
- Integration des Autorisierungsstandards OAuth2.0 in die IsyFact

Inhalte

- OAuth2.0 konforme Authentifizierung und Autorisierung
- Integration in Frontend- und Backendanwendungen
- Konfigurationen für unterschiedliche Anwendungsszenarien
- Individuelle Module zur Erweiterung der Funktionalität (z.B. Single-Session, Brute-Force-Detector, Anbindung Benutzerverzeichnis)
- Single-Sign-On

Funktionsweise

- Zugriff auf Datenhaltung des Bausteins Benutzerverzeichnis
- Basiert auf der OAuth2.0 Implementierung Keycloak
- Integration in die Anwendung bzw. in die Zugangswege der Anwendungslandschaft (Service-Gateways, Portal)
- Integration über Baustein Security (Backend) oder Baustein Portal (Frontend)

Checkliste: Einsatz des Bausteins

- ☒ Anbindung an Benutzerverzeichnis über Konfiguration aktivieren (falls gewünscht)
- ☒ Modulauswahl und Aktivierung
- ☒ Konfiguration entsprechend Vorgaben anlegen/importieren
- ☒ Integration in Zugangswege der Anwendungslandschaft vornehmen
- ☒ Integration in Anwendung vornehmen

Typ

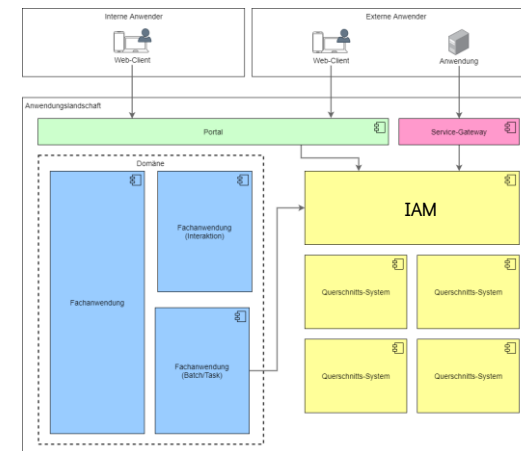


Querschnittsanwendung (QA)

Technische Basis

- Keycloak
- Spring WebClient
- IsyFact 3

Architektur



Zweck

- Querschnittsanwendung für einheitliche Generierung von Schreiben (Text, PDF)
- Versand von Schreiben über verschiedene Kanäle (z.B. Post, E-Mail) an menschliche Nutzer

Inhalte

- Template-basierte Schreibenerzeugung für Plain-Text-Dokumente, Text-Mails und PDF-Dokumente.
- E-Mail-Versand der erzeugten Dokumente/E-Mails, auch mit Anhangdokument(en).

Funktionsweise

- Einsatz eines Templating-Frameworks zur Dokumentengenerierung
- Generierung von PDF-Dateien auf Basis von Word-Dokumenten mit Aspose Word
- Erzeugung eines PDF-Generats basierend auf Pentaho
- Verschlüsseln und Signieren von E-Mails

Checkliste: Einsatz des Bausteins

- ✓ Erstellen von Text/E-Mail-Vorlagen
- ✓ Erstellen von PDF-Vorlagen mit Pentaho Report-Designer.
- ✓ Ablegen der Vorlagen in der Datenbank des Output-Managements
- ✓ Implementieren des Adapters für den Service-Zugriff im Anwendungskern der nutzenden Anwendung
- ✓ Nutzen des Adapters für Zugriffe auf Output-Management im Anwendungskern.

Typ

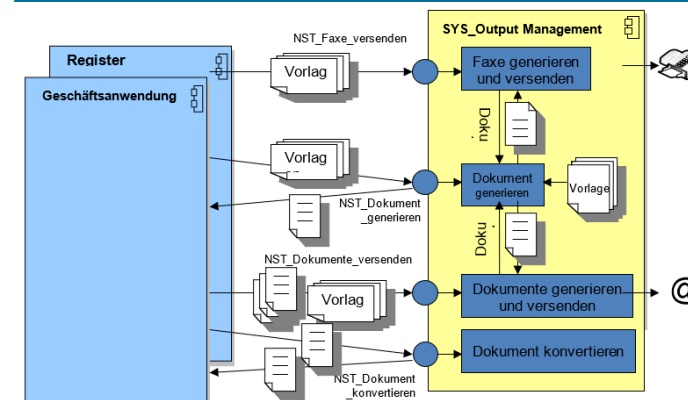


Querschnittsanwendung (QA)

Technische Basis

- Apache Velocity
- Pentaho
- JULIA MailOffice, Aspose Words/Aspose PDF
- IsyFact 3

Architektur



Zweck Funktionsweise Typ

- Erzeugung von Protokolleinträgen zur fachlichen Nachvollziehbarkeit der Arbeit mit einer Anwendung.
- Standardisierung der Protokolleinträge und fachliche Strukturierung der Protokolldaten.
- Recherche in den Protokolleinträgen über eine grafische Nutzungsschnittstelle.
- Unterstützung der Fristenkontrolle zur Löschung von Protokolleinträgen.

- Komponente „Protokollierung“ erzeugt Protokolldatensätze.
- Protokollierung wird durch Annotationen in den zu übergebenden Protokolldaten konfiguriert.
- Anwendung „Protokollrecherche“ zeigt Protokolldatensätze an und ermöglicht Recherche.
- Protokollrecherche ist generisch nutzbar für alle Protokolldatensätze, die durch die Protokollierung geschrieben werden.



Protokollierung:
Software-Bibliothek



Protokollrecherche:
Querschnittsanwendung (QA)

Technische Basis

- IsyFact 3
- JMS (optional)

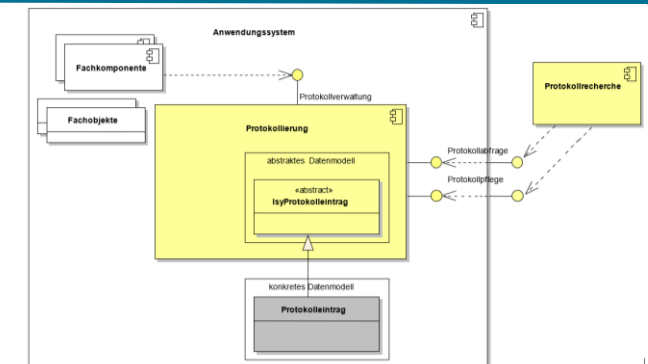
Inhalte

- Bibliothek (Spring-Komponente) zur Einbindung in die Anwendung, die Protokolldaten schreibt.
- Protokollierungsschnittstelle (Erzeugung der Einträge, Unterstützung Fristenkontrolle)
- Abfrageschnittstelle (Service) zur Anbindung der Protokollrecherche
- Eigenständige Anwendung für die Anzeige und Nutzung der Protokolleinträge.

Checkliste: Einsatz des Bausteins

- ☒ Einbinden der Protokollierung in die Anwendung
- ☒ Anlegen der Datenbanktabellen und Entitäten für die standardisierten Protokolldatensätze
- ☒ Konfiguration der zu protokollierenden Daten über Annotationen
- ☒ Konfiguration der Protokollierungskomponente
- ☒ Konfiguration der Anbindung der Protokollrecherche an das zu protokollierende System
- ☒ Verwendung der Schnittstelle Protokollverwaltung zum Schreiben von Protokolleinträgen

Architektur



Zweck

- Einheitliche Bereitstellung von Schlüsseldaten für alle Anwendungen und Services
- Beschränkung der Datenhoheit auf ein System
- Zentrale Querschnittsservices für den Zugriff auf die Schlüsseldaten

Inhalte

- Service-Schnittstelle zur Abfrage von Schlüsseln aus beliebigen Schlüsselräumen
- Flexibilität bei der Daten-Strukturierung:
 - Schlüssel und Unterschlüssel, Typen
 - Mapping von Schlüsseintrage eines Schlüssel-typs auf Schlüsseintrage eines anderen Typs
- Aktuelles Staatenverzeichnis
- Import von XÖV-Katalogen
- Client-Bibliothek zur vereinfachten und performanten Service-Nutzung und Caching
- Sicherheitsüberprüfung über Baustein Security
- Prüfen der Gültigkeit von Schlüsseln
- REST-Schnittstelle optimiert zur Anbindung an Angular-basierte Web Clients

Funktionsweise

- Schlüsselkataloge (strukturierte Excel-Sheets) als Datenquelle
- Service-Schnittstelle nach Referenzarchitektur „Anbieten interner Services“ (REST).
- Kommunikation mit Schlüsselverzeichnis über Client-Bibliothek

Checkliste: Einsatz des Bausteins

- ✓ Installation und Konfiguration des IT-Systems Schlüsselverzeichnis inkl. Bereitstellung Kataloge
- ✓ Einbinden der Client-Komponente in die nutzende Anwendung
- ✓ Konfigurieren der Client-Komponente
- ✓ Technischer System-Nutzer zur Aktualisierung der Kataloge
- ✓ Implementieren des Zugriffs auf Schlüsseldaten aus der nutzenden Anwendung

Typ



Schlüsselverzeichnis Client:
Software-Bibliothek

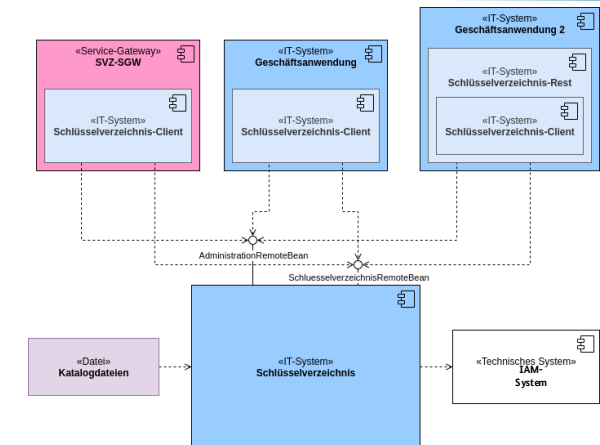


Schlüsselverzeichnis:
Querschnittsanwendung
(QA)

Technische Basis

- Apache POI, Spring Task Scheduling
- IsyFact 3

Architektur



Register Factory

Querschnittsanwendungen

Alphanumerisches Suchverfahren (ASV)

- Alphanumerische Suche nach beliebigen Datensätzen, insbesondere nach Personen.
- Suche nach gleichen und ähnlichen Personensätzen im Datenbestand.
- Bewertung der Ähnlichkeit verschiedener Personensätze.

Typ



Querschnittsanwendung (QA)

Biometrie

- Abgleich von Lichtbildern mithilfe biometrischer Verfahren.
- Prüfung der biometrischen Eignung von Lichtbildern.
- Bereitstellen einer biometrischen Suche. Anwendungen können Lichtbilder als Suchindex verwenden.

Typ



Querschnittsanwendung (QA)

Fingerabdruck

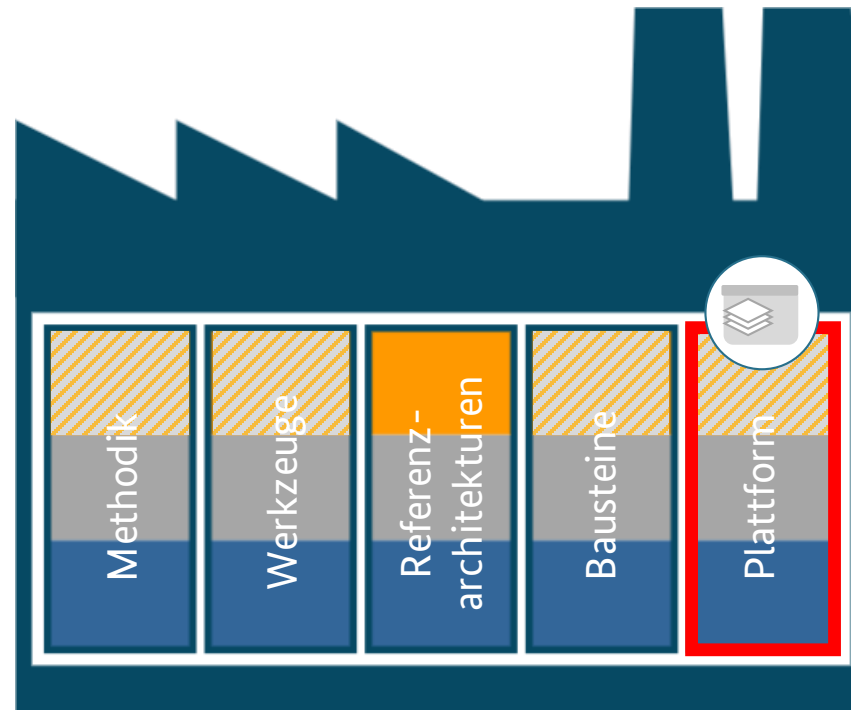
- Der Baustein Fingerabdruck stellt Funktionen zur Bearbeitung von Fingerabdrücken im ANSI/NIST-Format bereit.
- Unterstützt werden das Lesen und Beschreiben von ASCII-Records, das Lesen von binären Records sowie das Entfernen von ganzen Records.

Typ



Querschnittsanwendung (QA)

Plattform



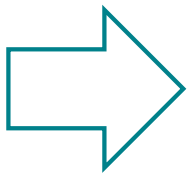


Was ist die Plattform?

- Skizze der Betriebsplattform der IsyFact/ Register Factory
- Definition der Deployment-Einheit
 - aktuell: keine Vorgabe, geplant: Container (IF)
- Darstellung der notwendigen Bestandteile der Betriebsplattform

Plattform Details:

- Deployment-Einheit: RPM
- Notwendige Bestandteile der Betriebsplattform (zwingend für den Betrieb benötigt):
 - Webserver
 - Applikationsserver
 - Middleware (Messaging, Queuing, ...)
 - Datenbanken (RDBMS, ...)
- Plattformen, die die Vorgaben erfüllen, können mit der IsyFact/ Register Factory erstellte Anwendungen aufnehmen und betreiben.



Durch die Plattform werden Systeme einheitlich installiert, konfiguriert und betrieben. Anforderungen an die Betriebsplattform sind einheitlich und zentral definiert.

Die IsyFact/ Register Factory stellt Vorgaben sowohl an die Anwendungen als auch an die betriebliche Plattform.