
NOVATEC
iSAQB
Foundation Level
Beispiele für Softwarearchitekturen

Agenda

1. LZ 5-1: Bezug von Anforderungen zu einer Lösung erfassen (R3)
2. LZ 5-2: Technische Umsetzung einer Lösung nachvollziehen (R3)

Beispiele für Softwarearchitekturen

- Dauer: 90 Min. Übungszeit: keine
- Dieser Abschnitt ist nicht prüfungsrelevant.

LZ 5-1: Bezug von Anforderungen zu Lösung erfassen (R3)

- Softwarearchitekt:innen haben an mindestens einem Beispiel den Bezug von Anforderungen und Architekturzielen zu Lösungsentscheidungen erkannt und nachvollzogen.

LZ 5-2: Technische Umsetzung einer Lösung nachvollziehen (R3)

- Softwarearchitekt:innen können für mindestens ein Beispiel die technische Umsetzung (Implementierung, technische Konzepte, eingesetzte Produkte, Lösungsstrategien) einer Lösung nachvollziehen.

Plattform für Vorsorgemanagement

NOVATEC

(Neu-)Entwicklung einer Plattform für die betriebliche Vorsorge

Ausgangssituation

Ein Anbieter der betrieblichen Vorsorge möchte seinen Anwendern (HR-Mitarbeiter von Kunden, Maklern und Vertriebsmitarbeitern) eine Plattform zur Verfügung stellen.

Fachliche Ziele

- Einsicht in bestehende Firmenverträge
- Schnelle und flexible Suche
- Abschluss neuer Mitarbeiterverträge zu einem Firmenvertrag
- Online Beratungen durchführen
- Geschäftsvorfallverwaltung mit Statustracking
  (Neuer Vertrag, Änderungen des Bezugsberechtigten oder Beitragsanpassungen)
- Self-Service (Änderung persönlicher Daten)

NOVATEC

(Neu-)Entwicklung einer Plattform für die betriebliche Vorsorge

Technische Ziele

- Hochverfügbar (keine Downtime) und skalierbar
- Performante Lese- und Suchvorgänge auf Vertragsdaten
- Entkoppelte, erweiterbare und flexible Bausteine (schnelle Release-Zyklen)
- Einsatz von Frameworks und „Managed Services" einer PaaS (Cache, Datenbank, Messaging Infrastruktur)
- Verwendung von technologischen Standards (z.B. Protokolle) und Unternehmensstandards (Framework, Infrastruktur, Bibliotheken und Toolchain)

Betrieb

- Sehr gute Systemüberwachung / Monitoring und einfache Diagnose / Fehlerbehebung
- Automatische Neustarts und Health Checks

Stakeholder

Rolle	Tätigkeit	Ziel
Leiterin Enterprise Architektur	Festlegung des Projektportfolios für das Unternehmen	Kostengünstige zentrale Lösung auf Basis von Standard-Komponenten
Leiter Vorsorgemanagement	Verantwortlichkeit für das Thema betriebliche Vorsorge	Reibungsloser Betrieb Hohe User Experience für das Fachpersonal der Firmenkunden
Verantwortliche für Endsysteme (Portale)	Neuentwicklung auf Basis moderner Entwicklungsansätze und Technologien	Digitalisierung der Vorsorge Mögliche Funktionalität abgeleitet auf Basis des existierenden Legacy Portals
Information Security Officer	Verantwortlich für übergreifende Sicherheitsthemen	Sichere Webanwendung Erfüllung von Compliance Anforderungen Verwendung von OAuth2

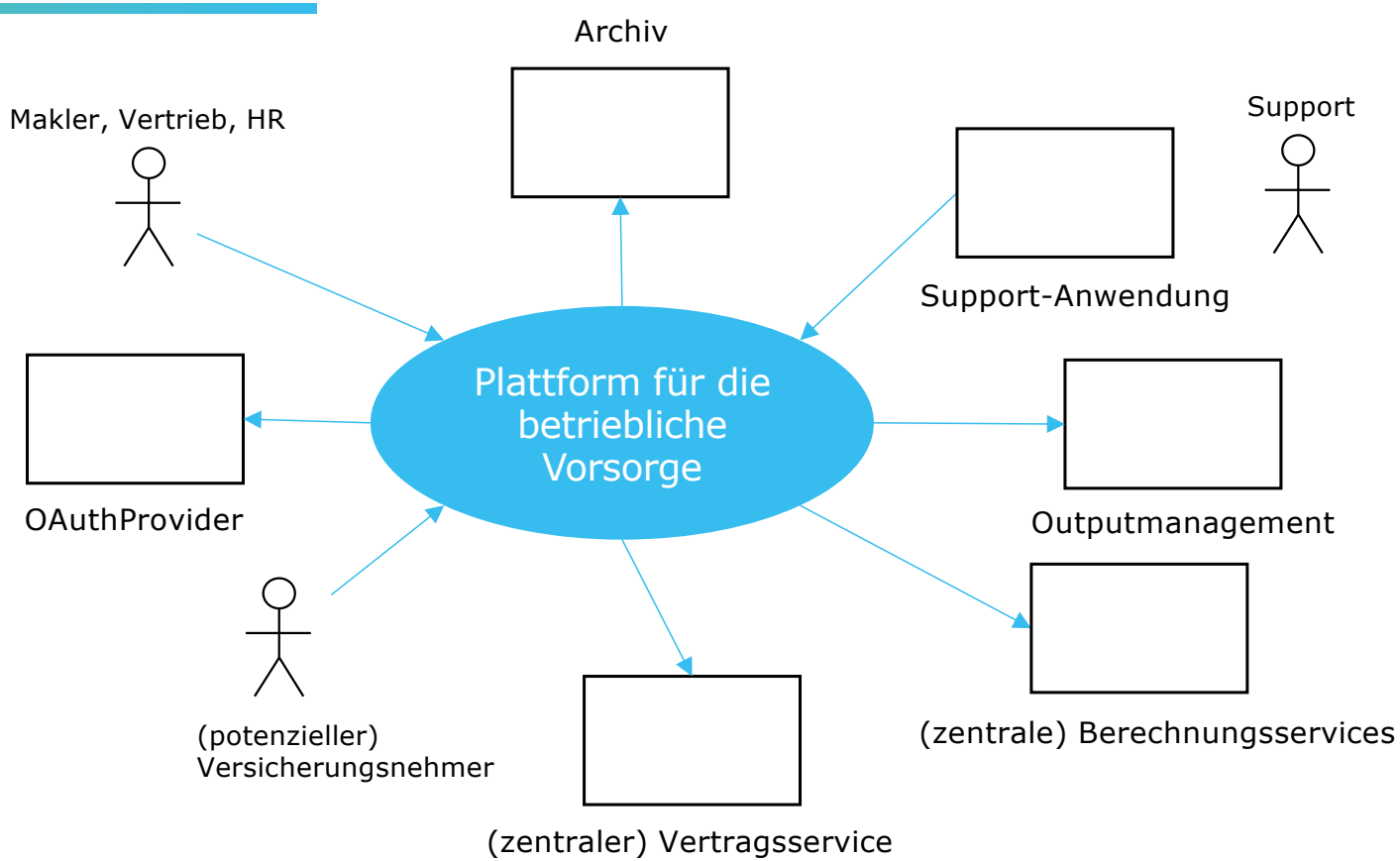
Technische Randbedingungen

Kategorie	Ausprägung
Infrastruktur	• Nutzung einer Platform as a Service für die Entwicklung und Bereitstellung der Infrastruktur. • Sensible Daten (Personendaten, personenbezogenen Daten, Betriebsgeheimnisse) dürfen nur in einer Datenbank im eigenen Rechenzentrum persistiert werden.
Entwicklungsumgebung	Einsatz von IntelliJ als IDE und Github als Source Code Verwaltungssystem sind als Unternehmensstandard definiert.
Build / Deployment	Gradle Build / Jenkins als Infrastruktur für Build und Deployment inklusive Continous Inspection Pipeline mit SonarQube als Software Quality Platform.
Programmiersprache und Frameworks	Aufgrund des Know-hows im Unternehmen soll Java und Kotlin auf Basis des Spring Frameworks im Backend und Angular im Frontend verwendet werden.
Architekturstil	• Asynchrone Kommunikation zwischen den Services (wo immer möglich) • Anwendungen müssen gemäß den Vorgaben der 12-Factors Apps zustandslos sein

Organisatorische Randbedingungen

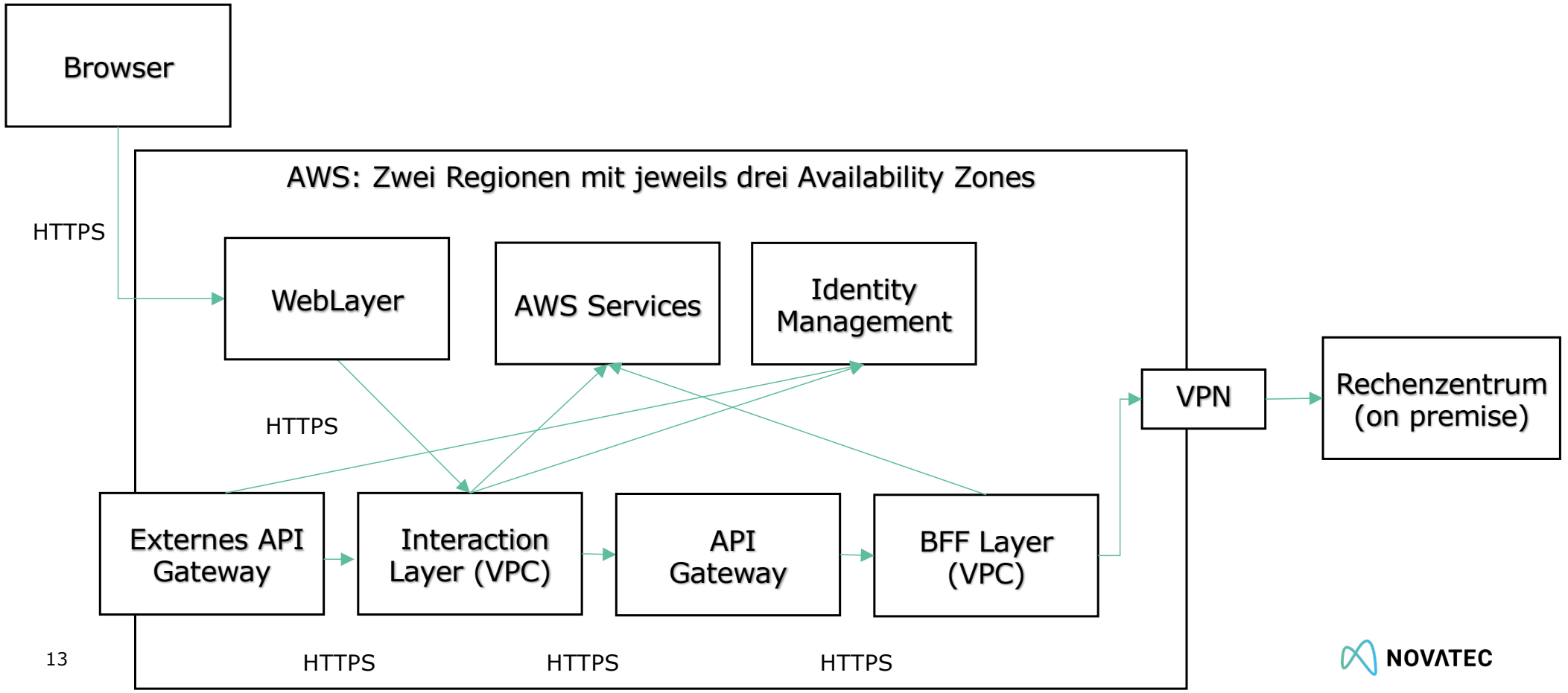
- Projekte müssen ihr Produkt in Ausschüssen vorstellen, um Budget-Freigaben für die nächste Etappe (3-Monate) zu erwirken
- Das Unternehmen besitzt die Strategie nach dem Vorbild des **Lean Startup** in unabhängigen Cross-Funktionalen Teams **agil Produkte** zu entwickeln
- Teams haben Freiheitsgrade bezüglich Technologien und Architekturstil, müssen aber zentrale Compliance Vorgaben der zentralen Architektur einhalten
- Polyglotte Microservice-Architekturen (Java- / Kotlin-basiert versus Typescript-basiert) werden von der Enterprise Architektur bis zu einem gewissen Grad unterstützt

Systemkontext (fachlich)



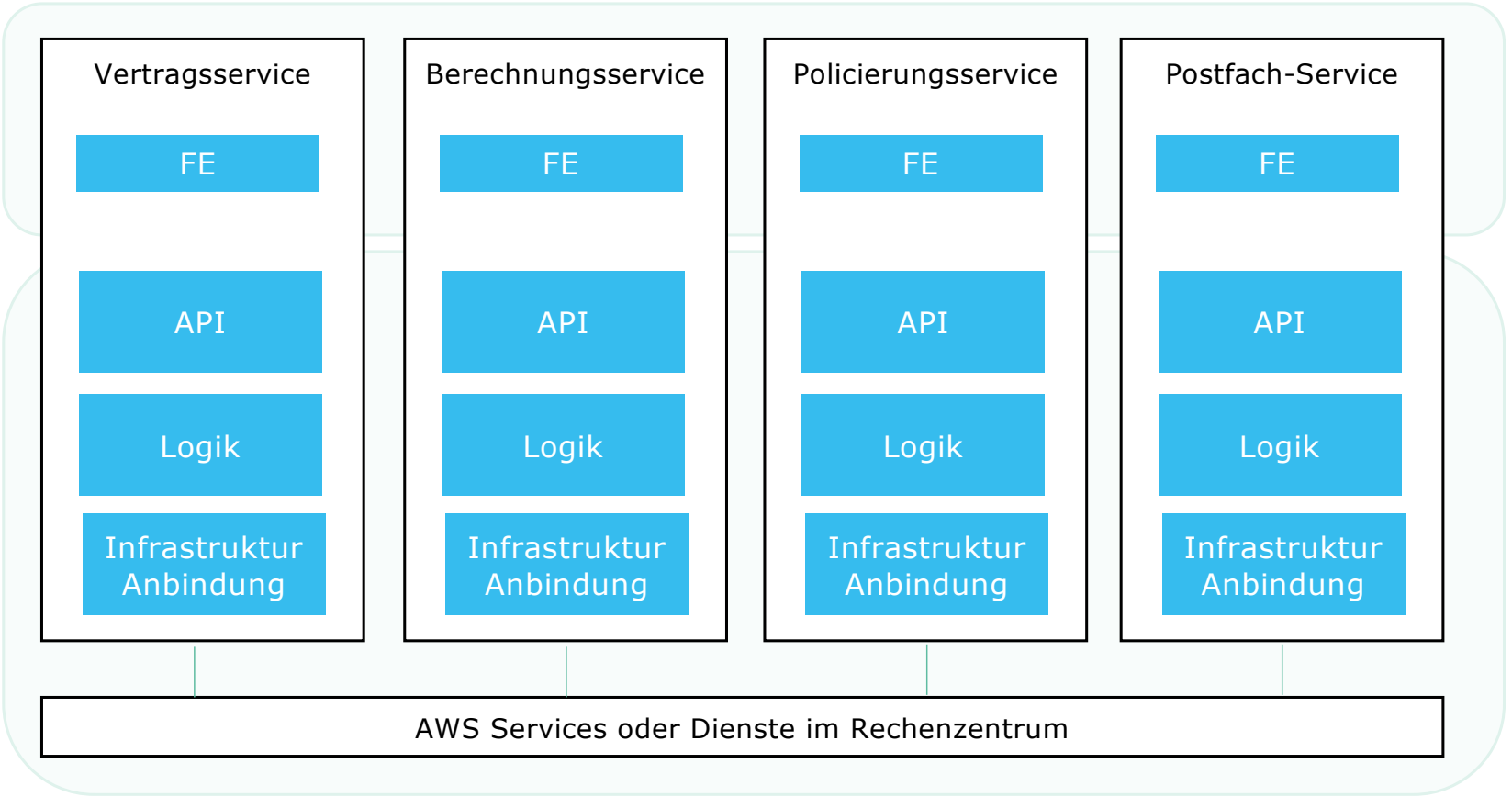
NOVATEC

Systemkontext (technisch) - Übersicht



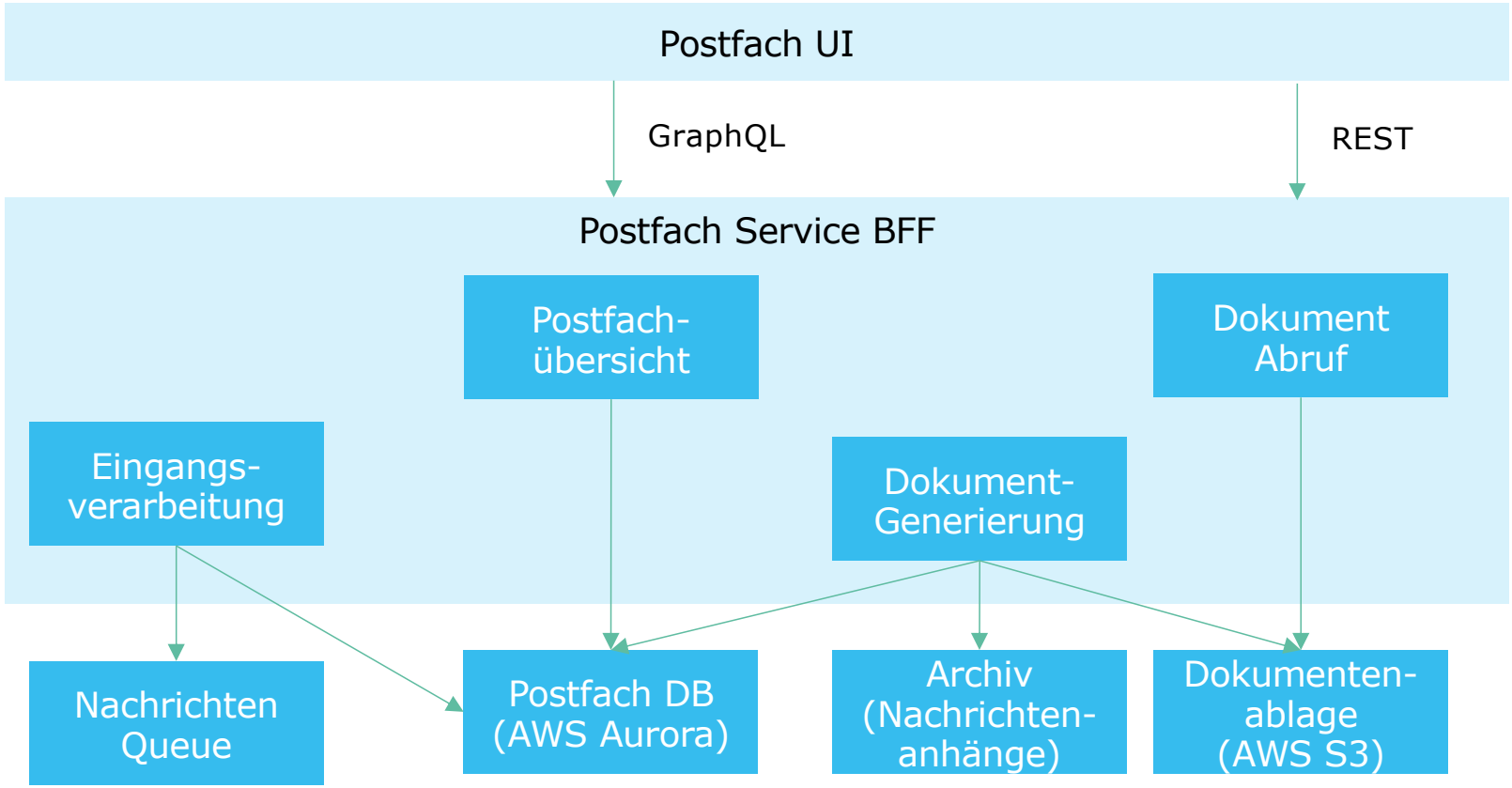
NOVATEC

Bausteinsicht Ebene 1 (Blackbox)



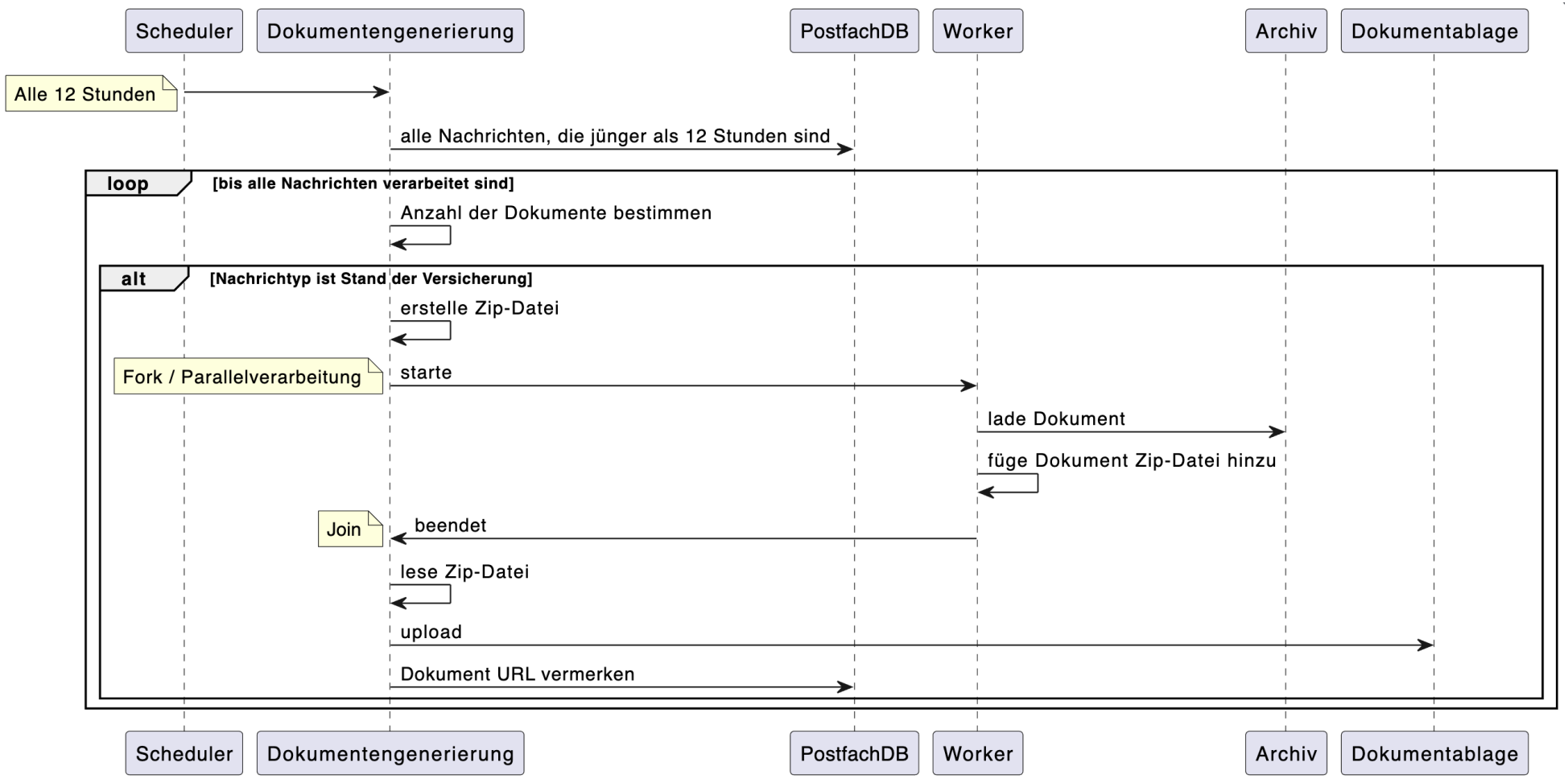
NOVATEC

Postfach-Service (Whitebox, Ebene 2)

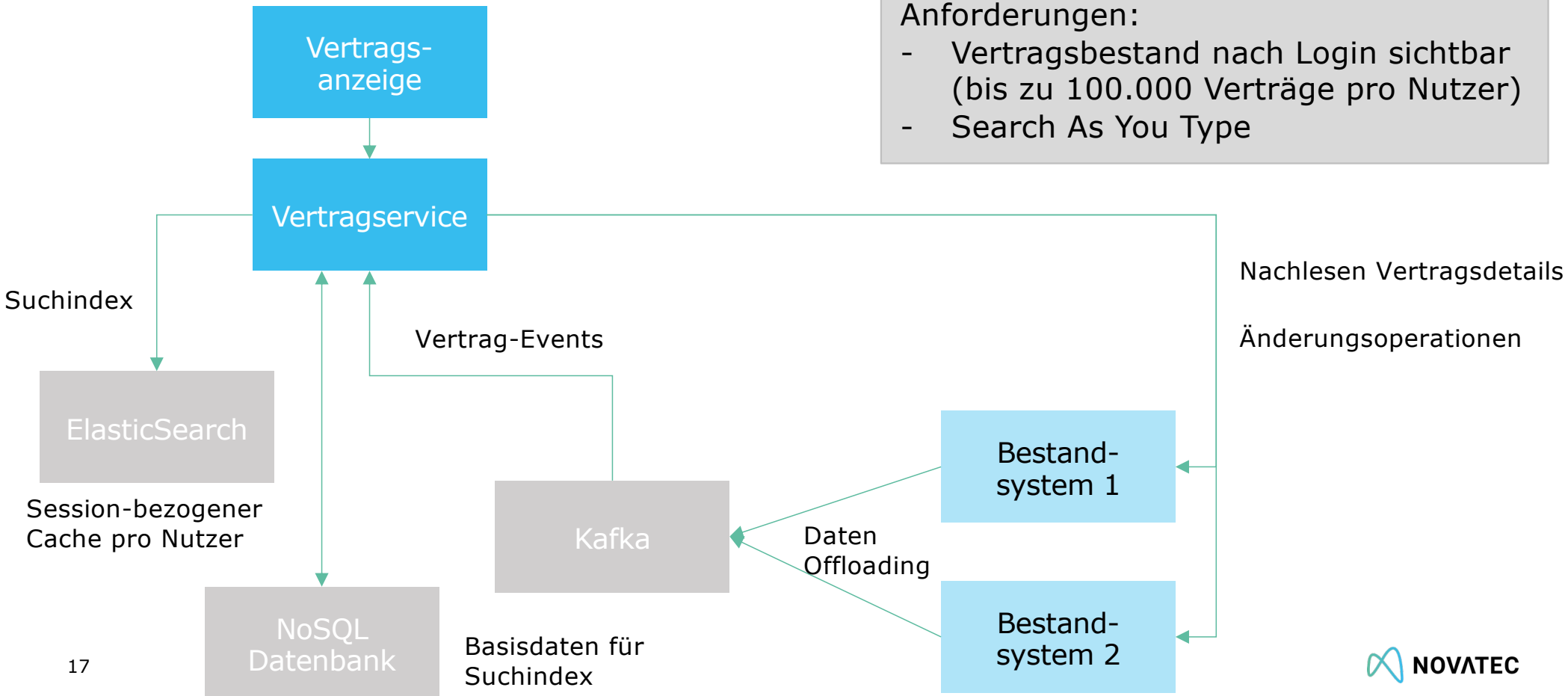


NOVATEC

Dokumentgenerierung Ebene 2 (Whitebox)

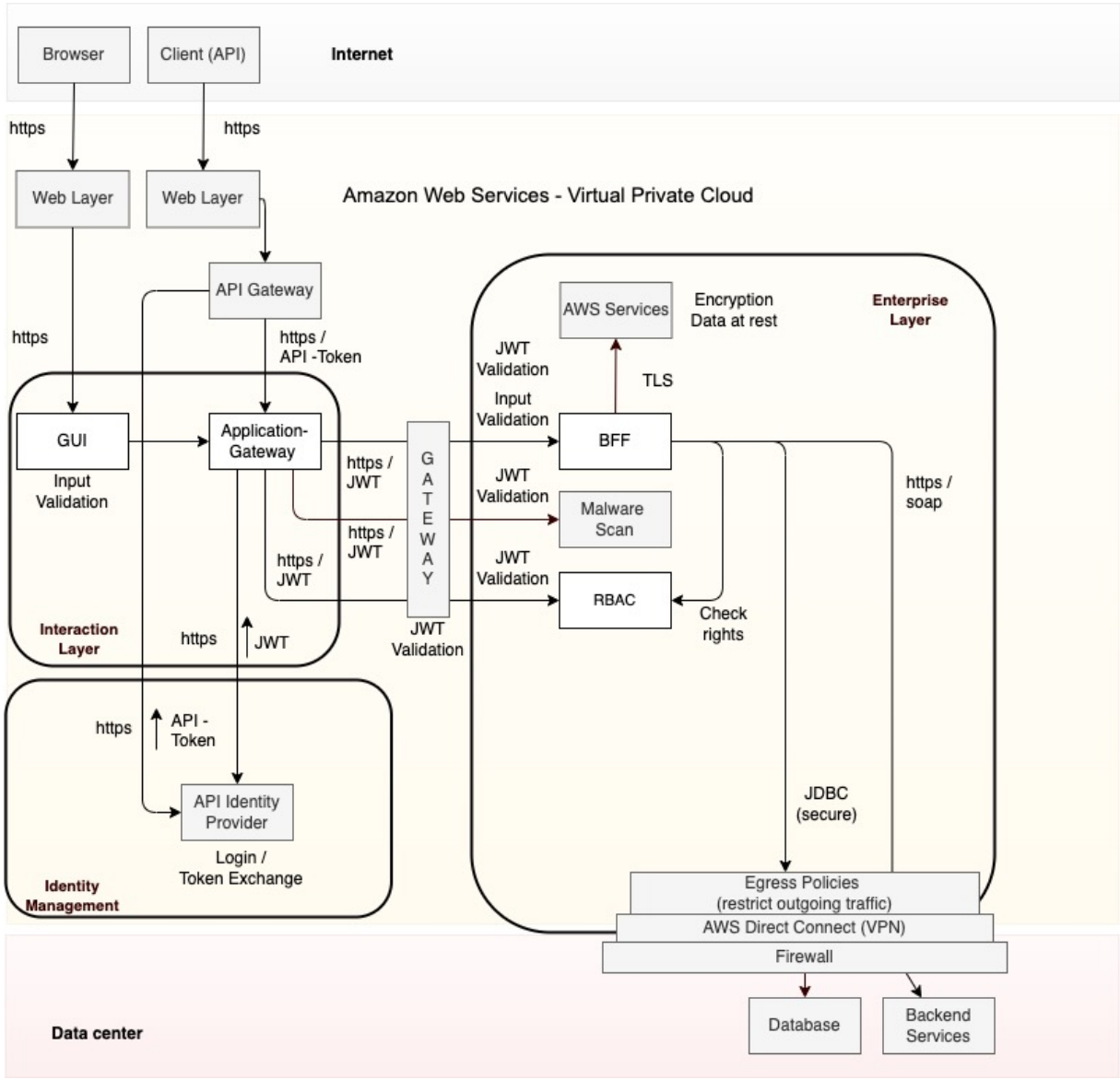


Architektur Vertragszugriffe: Lesen / Ändern



NOVATEC

Verteilungssicht



Technische Konzepte

API:

- REST und Messaging als Architekturstile
- CRUD für die Erzeugung von Geschäftsobjekten sowie die Abfrage von Daten

Ausnahme und Fehlerbehandlung:

- Runtime Exceptions mit eindeutigem Fehlercode werden auf System.out geloggt und werden in einem ELK-Stack gesammelt und bereitgestellt
- Das Frontend erhält HTTP Statuscodes

Cache:

- ElasticSearch als Suchcache
- Gemanaged von AWS (AWS ElasticSearch Service)

Integration:

- Entkoppelte Interoperabilität der Microservices durch Messaging
- Interoperabilität mit externen Services durch SOAP over HTTPS bzw. HTTPS für die Integration bestehender WebServices

Lessons Learned

Wiederverwendung

- Problem liegt nicht in der Technik sondern im menschlichen Bereich (Vertrauen / Not invented here / Respekt)
- Top-down Vorgehen (Vorgaben) funktionieren oft schlechter als Bottom-Up Vorgehen (unterschiedliche Ansätze vereinheitlichen), insbesondere in einem seniorigen Team

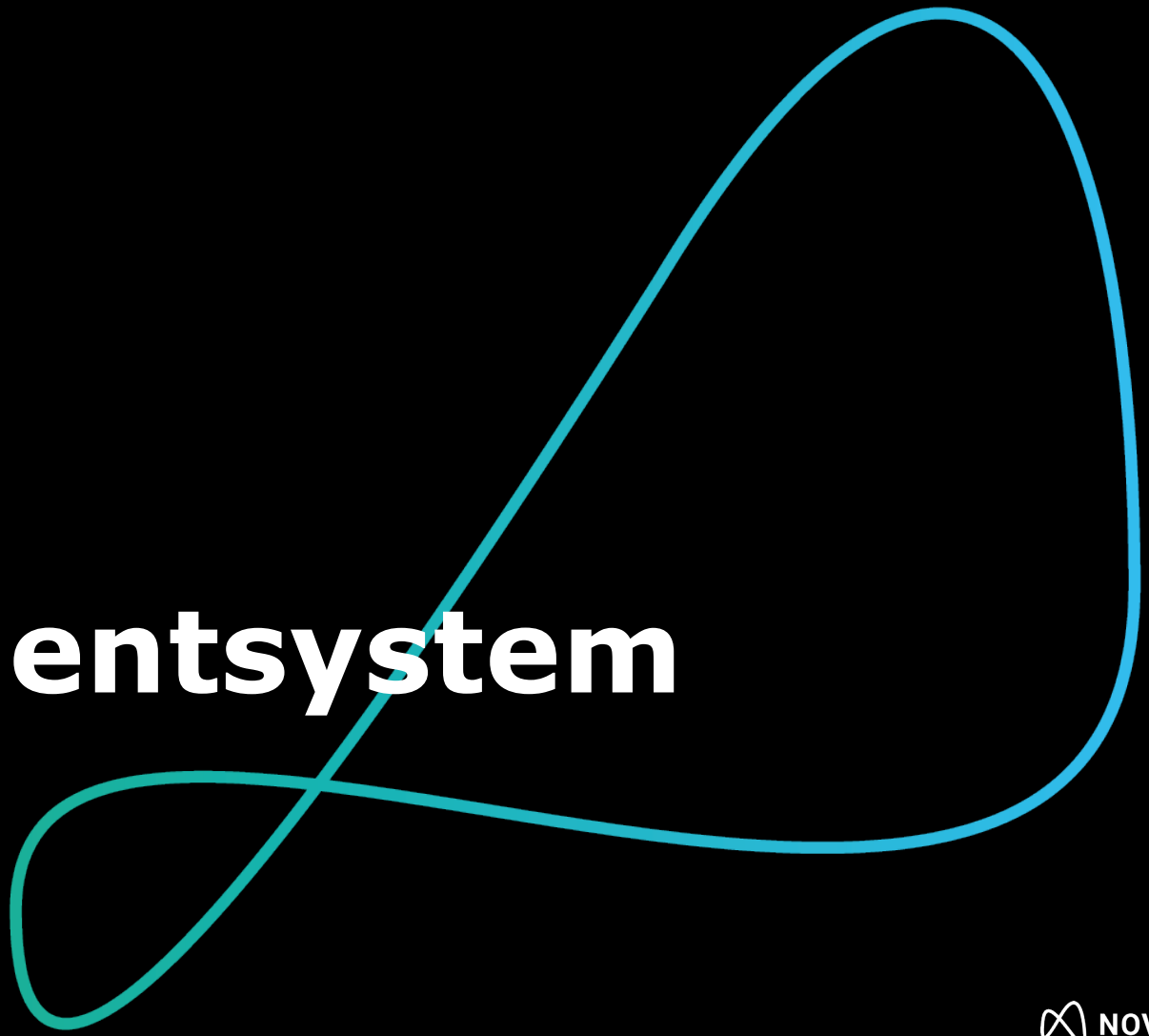
Frontend

- Mono-Repository um Wiederverwendung / Updates zu vereinfachen
- Trotzdem sind Deployments separater Service-UIs möglich

Backend

- Team-übergreifende Micro Libraries zur Erzielung von Wiederverwendung (kleine Einheiten, wenige Abhängigkeiten, alle Teams müssen zustimmen)
- Leitfäden zur Beschreibung von Konzepten (Wie setze ich Logging geeignet ein)

Output-Managementsystem



NOVATEC

Einführung eines unternehmensweiten Output-Managementsystems

Ausgangssituation

Ein Finanzdienstleister hat mehrere dezentrale, anwendungsspezifische Lösungen zur Erzeugung von Dokumenten im Einsatz.

Diese Situation führt im laufenden Betrieb zu hohen Aufwänden:

- Änderung von Dokumentformaten und Layouts
- Anpassungen von zentralen Elementen (Vorstandszeile)
- Systemüberwachung / Monitoring
- Die Fehlersuche umfasst mehrere Systeme und Teams
- Vielfach keine effiziente Unterstützung von Batchabläufen

Kunden erhalten viele einzelne Sendungen abhängig vom Quellsystem.

Zielsetzungen für das unternehmensweite Output-Managementsystem

Fachliche Ziele

- Zentrale Vorlagen-Verwaltung, einheitliche Dokumente über Systemgrenzen hinaus
- Nutzung von Geschäftsregeln bei der Dokumentgenerierung
- Einführung einer Portokostenoptimierung durch Bündelung von Dokumenten

Technische Ziele

- Modularität / gute Erweiterbarkeit hinsichtlich Compliance-Anforderungen
- Expliziter zentraler Prozess anstatt individueller Umsetzungen
- Einsatz von Standard-Komponenten, soweit möglich

Betrieb

- Sehr gute Systemüberwachung / Monitoring und einfache Diagnose / Fehlerbehebung

Stakeholder für das unternehmensweite Output-Managementsystem

Rolle	Beschreibung	Ziel
Leiter Enterprise Architektur	Festlegung des Projektportfolios	Kostengünstige zentrale Lösung auf Basis von Standard-Komponenten
Leiterin Digitalisierung	Verantwortlichkeit für das Thema Output-Management	Reibungsloser Betrieb Portokostenoptimierung durch Dokumentbündelung
Verantwortliche für Endsysteme	Austausch ihrer Individuallösung durch die unternehmensweite Lösung	Bisherige Funktionalität soll erhalten bleiben Einfache Migration
Endkunden Kundenbetreuer	Treten im Projektverlauf nicht auf	Einheitliches Erscheinungsbild der Dokumente

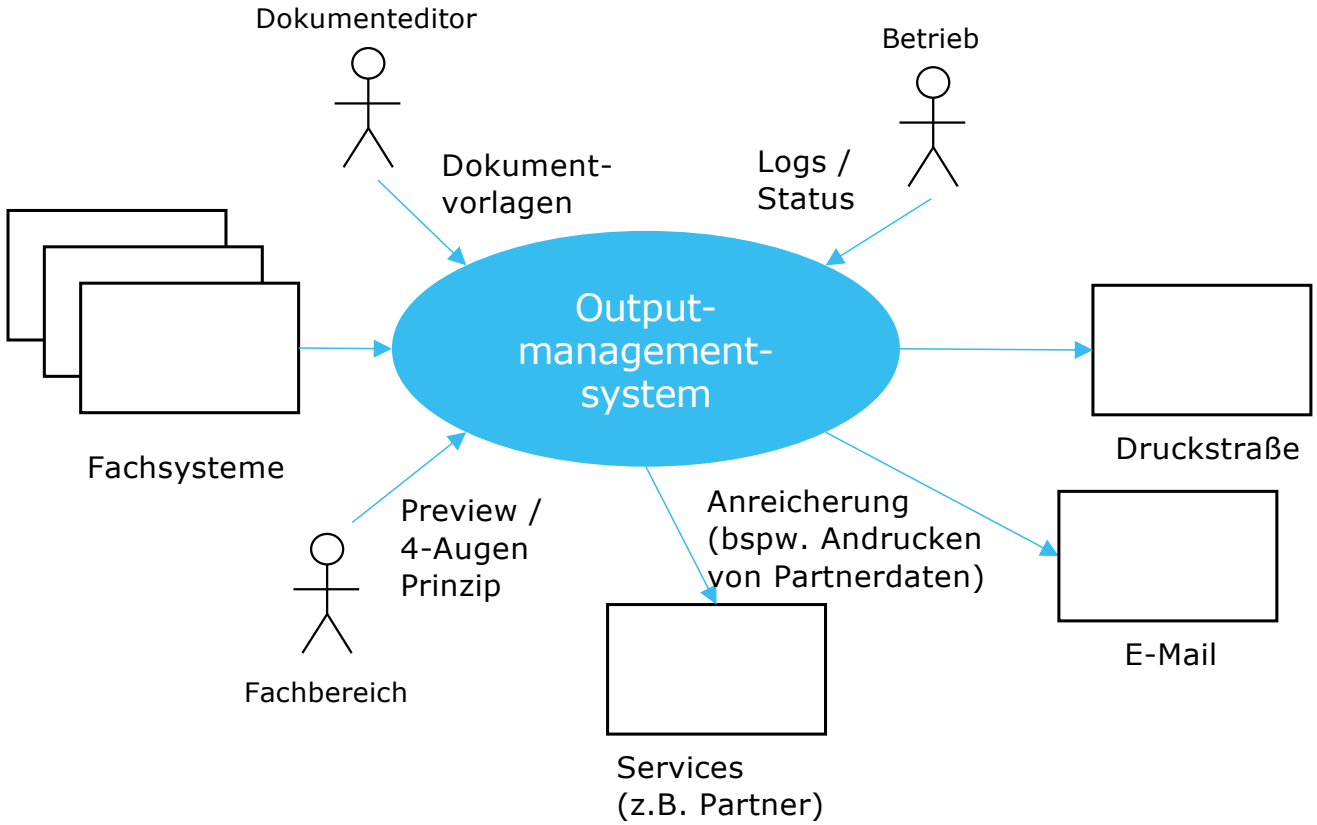
Technische Randbedingungen

Kategorie	Erläuterung
Hardware	Alle Anwendungen, die als unternehmenskritisch eingestuft sind, müssen auf Linux (im Gegensatz zu Windows) betrieben werden.
Entwicklungsumgebung	Einsatz von eclipse und git als Source Code Verwaltungssystem sind als Unternehmensstandard definiert.
Build / Deployment	Maven Build / Jenkins als Infrastruktur für Build und Deployment (inklusive Sonar Scanner Plugin) als Unternehmensstandard.
Programmiersprache und Frameworks	Aufgrund des Know-hows im Unternehmen soll Java und Spring Framework verwendet werden.
Laufzeitumgebung	• Alle Anwendungen, die als unternehmenskritisch eingestuft sind, müssen auf WebSphere Application Server betrieben werden. Für Messaging ist WebSphere MQ zu verwenden. • Als Outputmanagement Software kommt Dopix zum Einsatz. • Die Versandsteuerung wird über Posy realisiert.

Organisatorische Randbedingungen

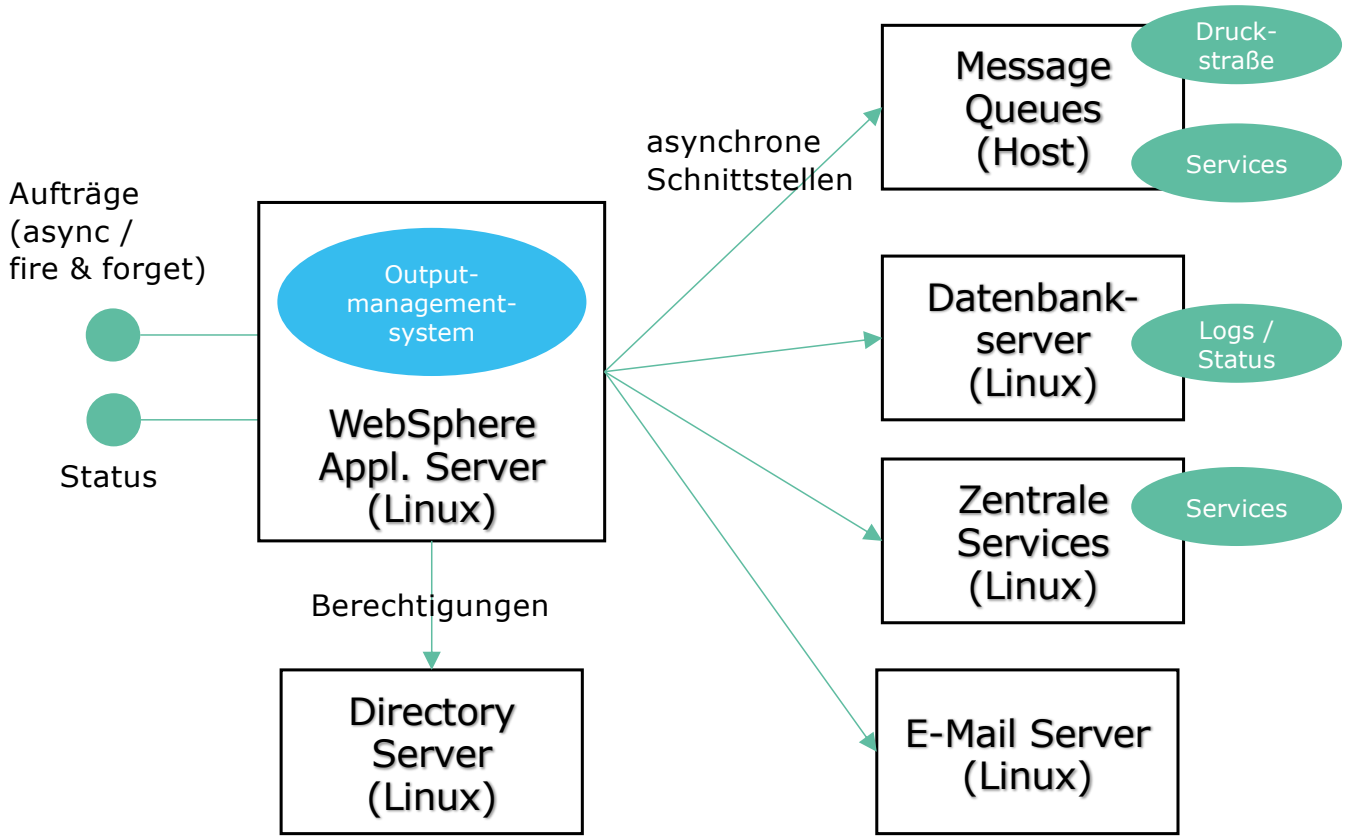
- Unternehmenskritische Projekte müssen regelmäßig ihren Fortschritt sowie ihre Architekturentscheidungen vor dem Architekturausschuss vorstellen.
- Unternehmenskritische Projekte müssen ihre technische Konzeption vor dem Architekturausschuss reviewen lassen.
- Das Unternehmen besitzt die Strategie zentrale Services einzuführen.
- Nachdem die existierenden Lösungen für Output-Management sehr wartungsintensiv sind, existieren bereits viele Fachprojekte, welche das neue System einsetzen wollen.
- Mitarbeiter des Kunden stehen nicht zu 100% ihrer Zeit dem Projekt zur Verfügung, da sie noch weitere Aufgaben innerhalb des Unternehmens wahrnehmen.

Systemkontext (fachlich)



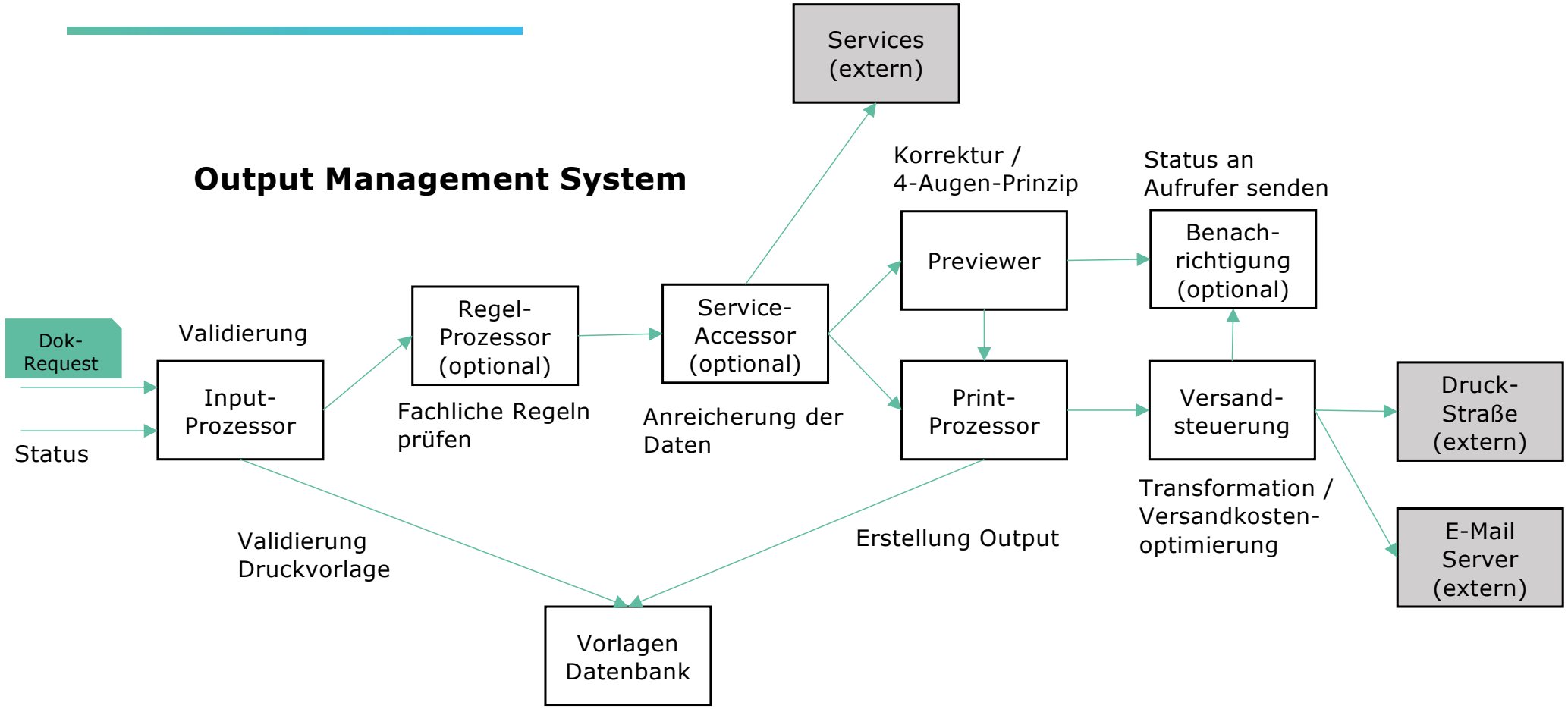
NOVATEC

Systemkontext (technisch) - Übersicht



NOVATEC

Bausteinsicht Ebene 1 vereinfacht (Whitebox)

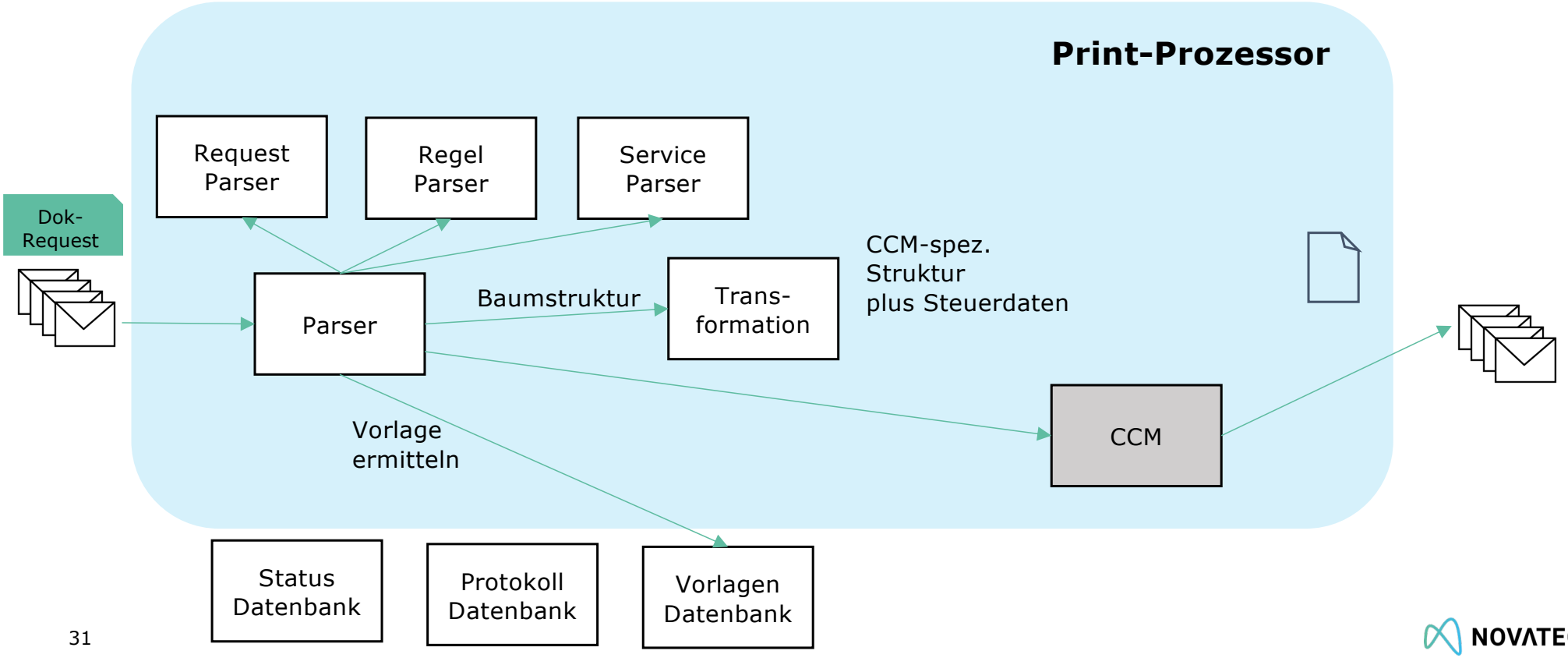


NOVATEC

Architekturentscheidung Ablaufsteuerung

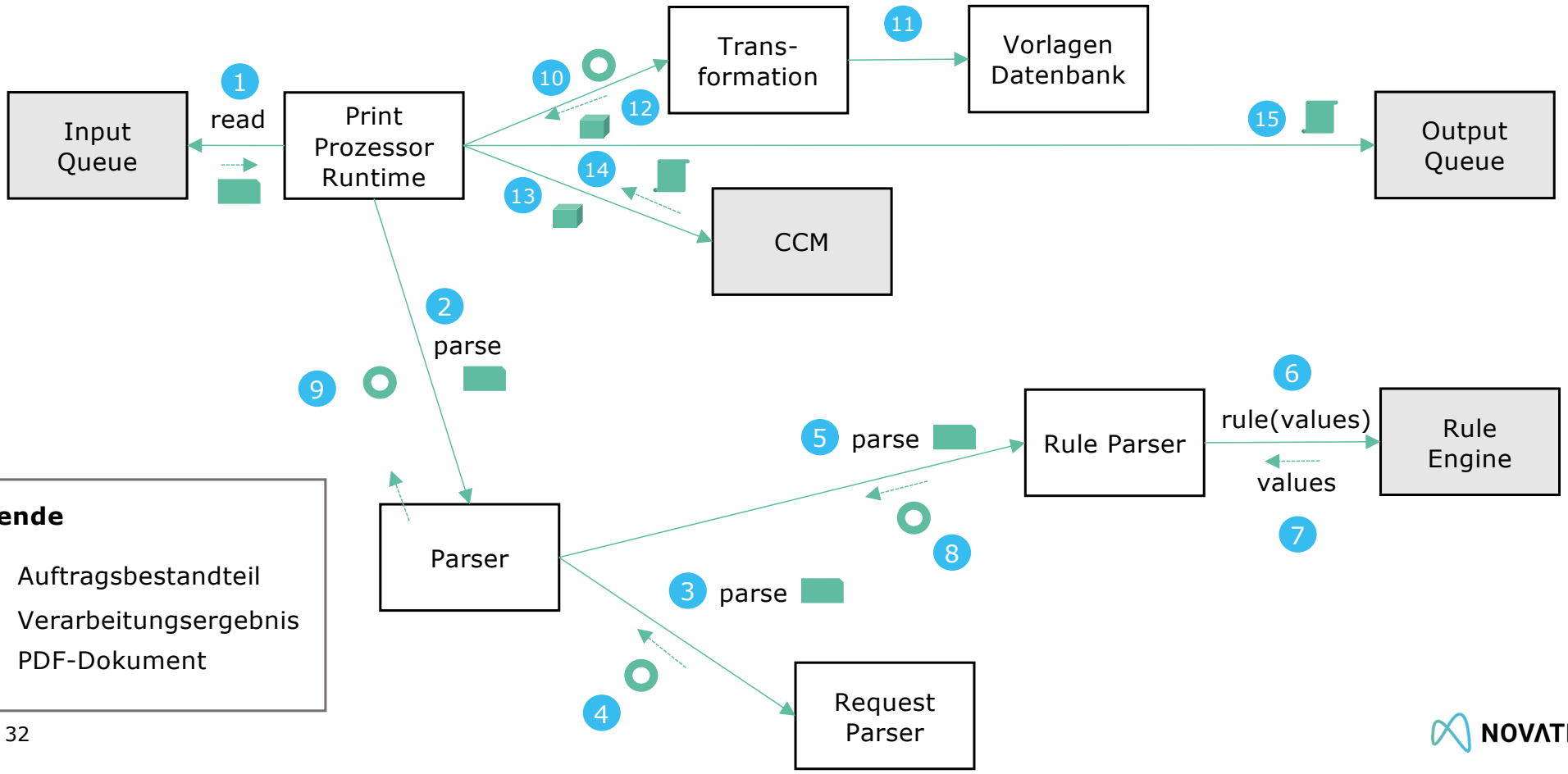
Kriterium	Kommerzielles BPM	Open Source BPM	Eigenentwicklung auf Basis von Apache Camel
Fachliche Anforderungen	+	+	+
Entwicklungsaufwand	- (fehlendes Know-how)	- (fehlendes Know-how)	o
Deployment-Aufwand	-	o	+
Pflegeaufwand	- (Versionsupdates)	- (Versionsupdates)	o (Versionsupdates Software)
Betrieb	-	+	+
Zukunftsfähigkeit	- (Abbau geplant)	o (erster Anwender)	+
Lizenzkosten	-	+	+
Performanz	o	o	o
Infrastrukturkosten	-	+	+

Bausteinsicht Print-Prozessor Ebene 2



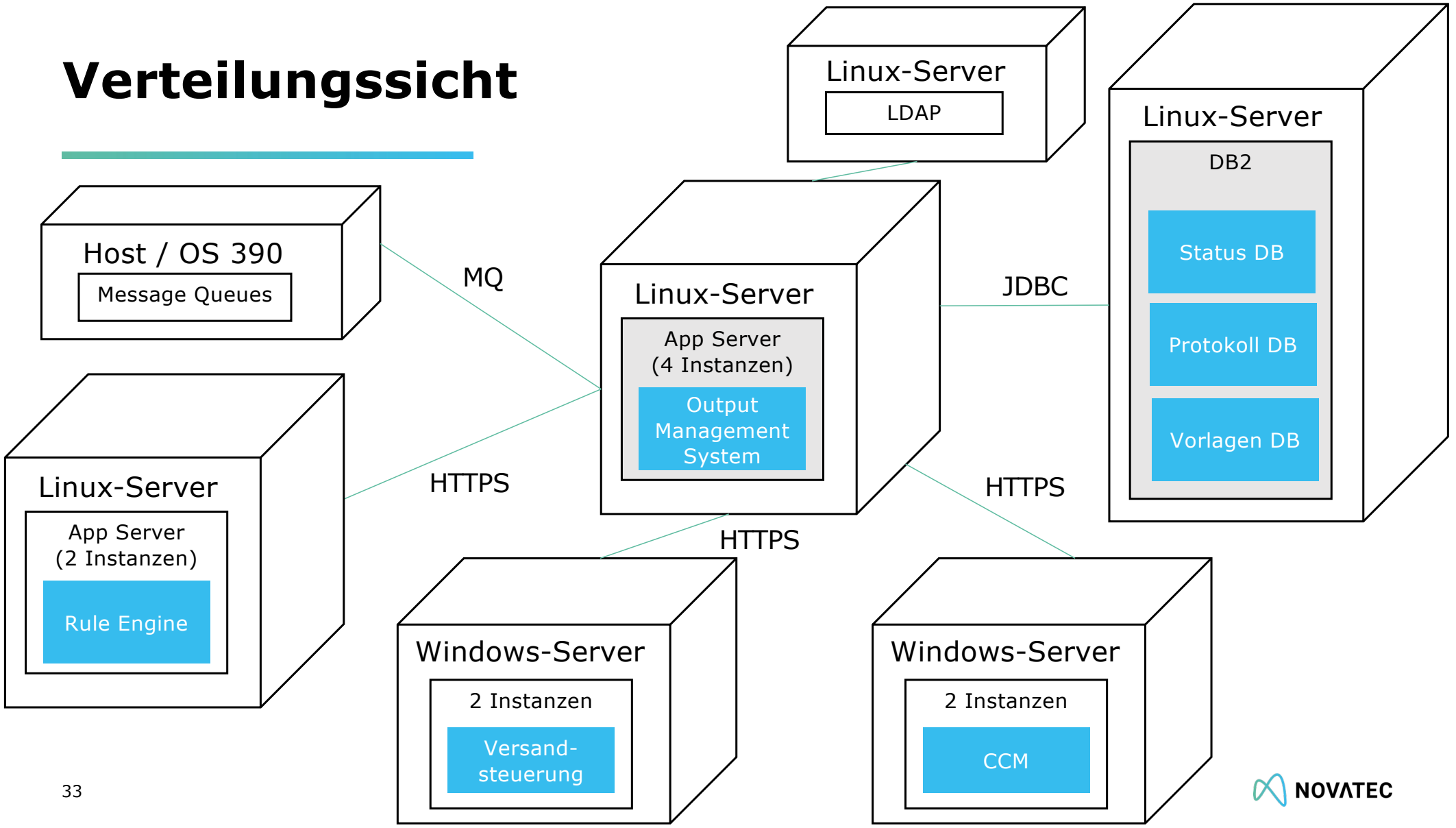
NOVATEC

Laufzeitsicht Print-Prozessor (vereinfacht)



NOVATEC

Verteilungssicht



NOVATEC

Technische Konzepte

- **Persistenz**: JPA mit Spring Data, bei komplexen Queries Spring JDBC Template
- **Ablaufsteuerung**: Eigenentwicklung wobei einzelne Komponenten über Queues entkoppelt sind (analog Apache Camel)
- **Ausnahme und Fehlerbehandlung**: Runtime Exceptions mit eindeutigem Fehlercode werden geworfen, Fehler in Status DB und Protokoll DB gespeichert.
- **Transaktionssteuerung**: 2PC mit Applikationsserver als Transaktionsmanager
- **Geschäftsregeln**: Einsatz eines kommerziellen Werkzeugs (ILOG), Unternehmensstandard
- **Integration**:
  - Integration mit Services auf dem Host und innerhalb der Anwendung: persistente Message Queues
  - SOAP/HTTP für die Integration bestehender WebServices

Weitere Dokumentation

Qualitätsszenarien und Risiken sind nicht Teil der Beschreibung.

Quellen

[Starke 2015] Starke, Gernot: Effektive Software-Architekturen: Ein praktischer Leitfaden. 7. Aufl. Hanser

Quellen

[Starke 2015] Starke, Gernot: Effektive Software-Architekturen: Ein praktischer Leitfaden. 7. Aufl. Hanser

[AWS 2021a] Overview of Amazon Web Service. [Online] https://docs.aws.amazon.com/whitepapers/latest/aws-overview/aws-overview.pdf, zuletzt aufgerufen 18.05.2021

[AWS 2021b] AWS: AWS Well-Architected Framework. [Online] https://docs.aws.amazon.com/wellarchitected/latest/framework/wellarchitected-framework.pdf, zuletzt aufgerufen 18.05.2021

Novatec Consulting GmbH
Berta-Benz-Platz 1
D-70771 Leinfelden-Echterdingen

T. +49 711 22040-700
info@novatec-gmbh.de
www.novatec-gmbh.de