HOCHSCHULE HEILBRONN

Software Engineering komplexer Systeme

Vorlesung an der Hochschule Heilbronn 2020

Prof. Dr.-Ing. Andreas Heil

HEUTIGER INHALT

- C4 Model

LERNZIELE

Probleme bei der Dokumentation von Software-Architekturen verstehen

- verstehen

Die vier Ebenen des C4-Models

- kennen

Das C4 Modell auf Problemstellungen

- anwenden können

Software-Architekturen mit Hilfe des C4-Models

- dokumentieren und kommunizieren können

Software-Architektur dokumentieren

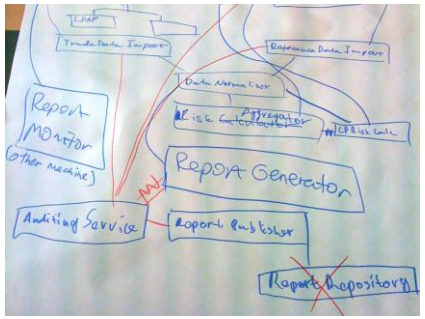
C4 MODEL

Inhalt heute

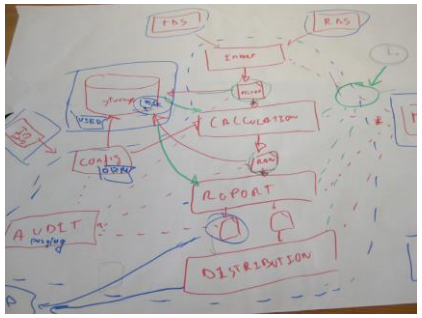
- Visuelle Kommunikation von Software-Architekturen
  - Baugewerbe vs. Software-Entwicklung
- C4 Model
  - Code Landkarten
- Abstraktions-Level
  - System Contex-Diagramme
  - Container-Diagramme
  - Komponentendiagramme
  - Code-Diagramme
  - Ergänzende Diagramme

KURZE WIEDERHOLUNG

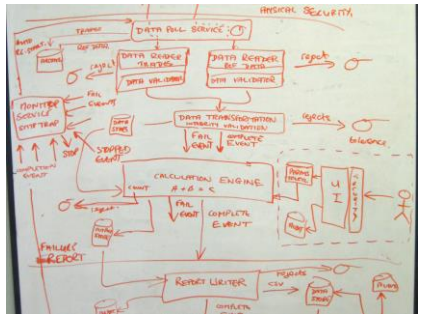
Probleme von Kästchen und Linien



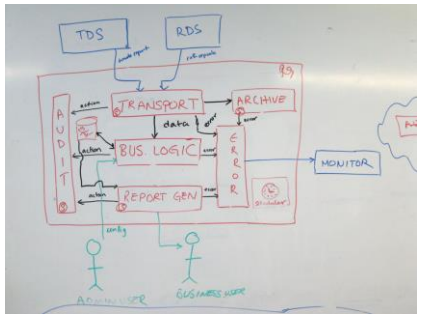
So?



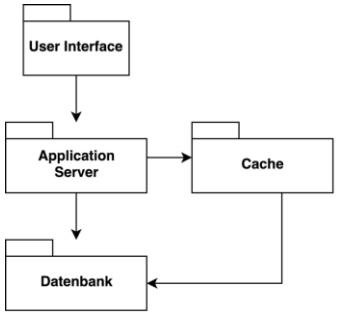
So?



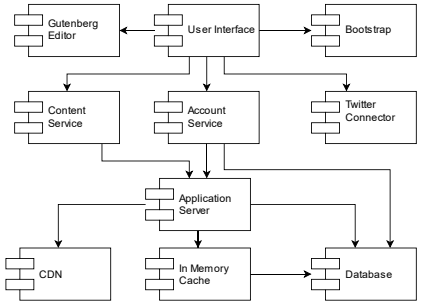
Oder so?



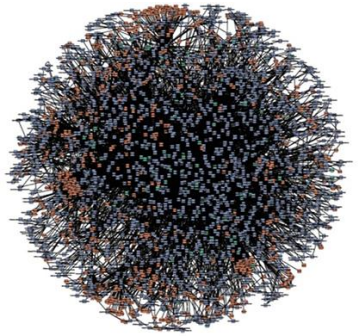
Vielleicht so?



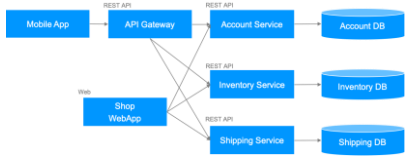
Besser so?



Noch besser so?



Automatisiert so?



Oder eben so...

Quelle: http://c4model.com/

KURZE WIEDSERHOLUNG
Übung

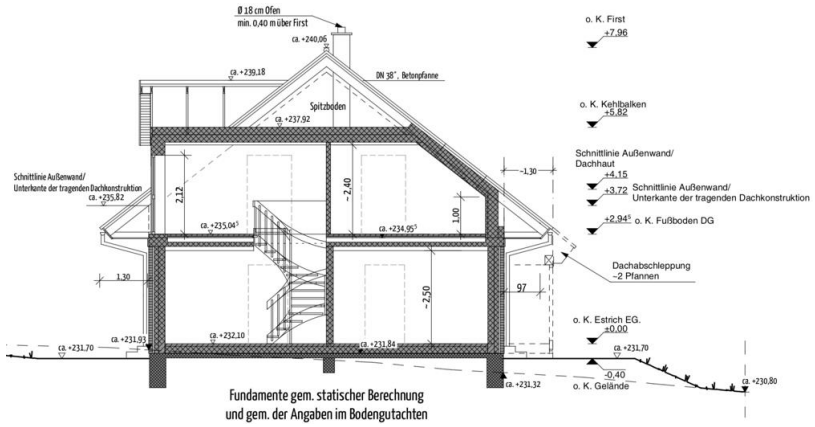
- Bildern Sie 3-er/4-er Gruppen (Banknachbarn), so dass es eine gerade Anzahl von Gruppen gibt
- Teil 1
  - Stellen Sie sich kurz gegenseitig jeweils das größte Software-Projekt vor, an dem Sie bisher gearbeitet haben und wählen Sie in Ihrer Gruppe ein Projekt aus
  - Zeichnen Sie die Architektur des Projektes mit einer Dokumentationsform Ihrer Wahl
  - Dauer 10 Minuten

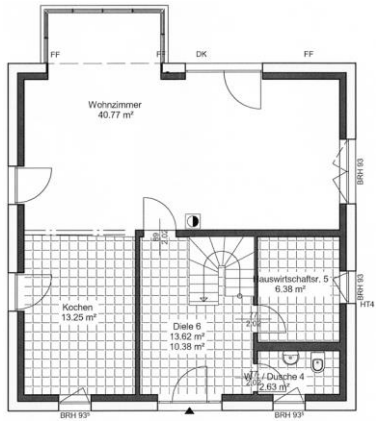
- Teil 2
  - Stellen Sie Ihre Architektur einer der anderen Gruppen vor
  - Dauer 5 Minuten

MOTIVATION

Visuelle Kommunikation im Baugewerbe

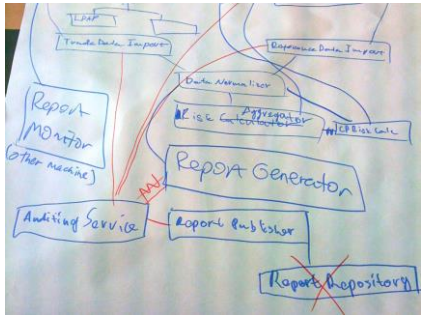
- Lagepläne
- Raumpläne
- Bebauungspläne
- Querschnittsansichten
- Detaillierte Standardzeichnungen
- Technische Zeichnungen

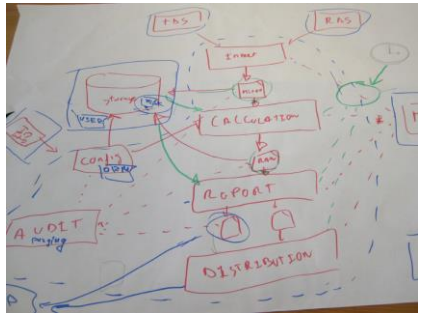


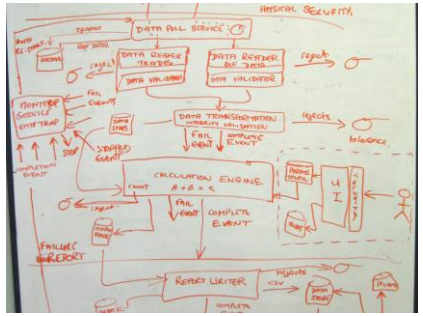


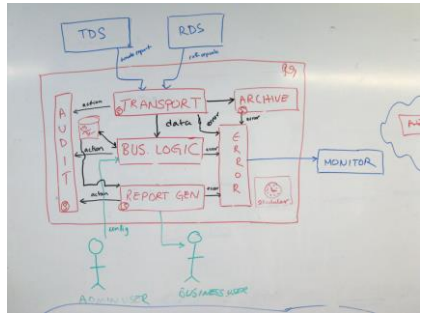
MOTIVATION
Kommunikation von Software-Architekturen

- Durcheinander von Boxen und Linien
- Inkonsistente Notationen, Farben, Formen und Linien
- Mehrdeutige Bezeichnungen
- Beziehungen nicht benannt
- Fehlende Technologieentscheidungen und Details
- Vermischte Abstraktionslevel









Quelle: http://c4model.com/

MOTIVATION

- Seiteneffekt aufgrund der Bewegung hin zu agilen Methoden
  - Weniger “Up Front Design
  - Weniger Software Diagramme

- Wenn Diagramme genutzt werden sind diese
  - Unklar
  - Mehrdeutig
  - Verwirrend
  - Vermischte Detailgrade

CODE LANDKARTEN

Kommunikation von Software-Architekturen

- C4 Model ist gedacht um Software-Architekturen zu modellieren und zu kommunizieren
- “Abstraction First“ Ansatz
- Reflektiert die Denkweise von Entwicklern und Software-Architekten
- Geeignet um Architekturen “Up Front” oder im Nachhinein zu dokumentieren
- Nutzt Code Landkarten (engl. code maps) um verschiedene Detaillevel zu beschreiben

[1] Bildquelle Peter Schmelzle - Self-photographed, CC BY-SA 3.0
[2] Bildquelle https://maps.google.com

ABSTRAKTIONSLEVEL

Übersicht: Statische Sichten auf Software-Systemen

- Kontext
  - Big Picture
  - Wie passen die Teile in den Gesamtkontext?
- Container
  - Einheiten, die Code “hosten“
- Komponenten
  - Gruppe von zusammengehörigen Funktionen
  - Verborgen hinter wohldefinierten Schnittstellen
- Code
  - Klassen, Schnittstelle, Objekte, Funktionen, Tabellen etc. innerhalb einer Komponente

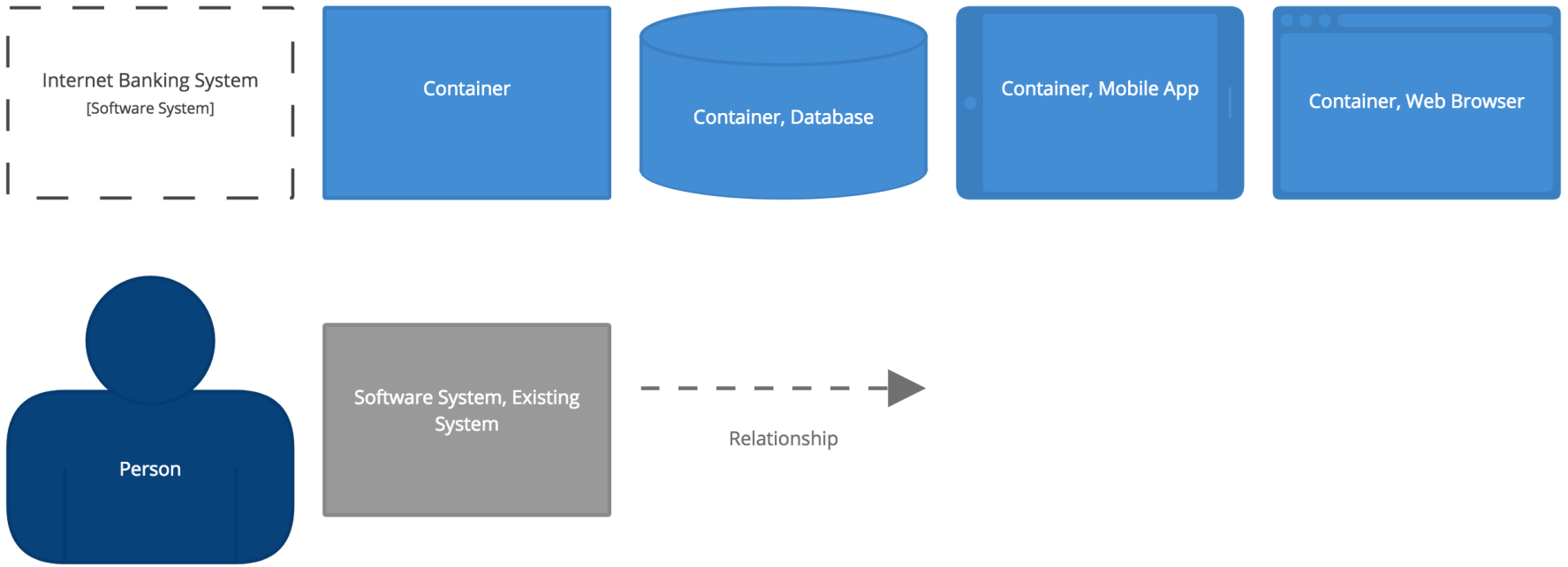
META MODELL
Elemente und Beziehungen

Element	Übergeordnetes Element	Eigenschaften
Person	keine	• Name • Beschreibung • Lage (Intern oder Extern)
Software System	keine	• Name • Beschreibung • Lage (Intern oder Extern) • Container aus denen das Software System besteht
Container	Software System	• Name • Beschreibung • Technologie • Komponenten aus denen der Container besteht
Komponente	Container	• Name • Beschreibung • Technologie • Code-Bestandteile aus denen die Komponente besteht
Code	Komponente	• Name • Beschreibung • Voll qualifizierter Typ
Beziehung		• Beschreibung • Technologie

META MODELL
Views (4C)

View	Scope	Eigenschaften
System Context	Software System	• Software Systeme • Personen
Container	Software System	• Software Systeme • Personen • Container innerhalb des im Scope befindlichen Software Systems
Component	Container	• Software Systeme • Personen • Weitere Container im übergeordneten Software System des im Scope befindlichen Containers • Komponenten innerhalb des im Scope befindlichen Containers
Code	Component	• Code-Elemente (z.B. Klassen, Schnittstellen), die zur Implementierung der im Scope befindlichen Komponente genutzt werden

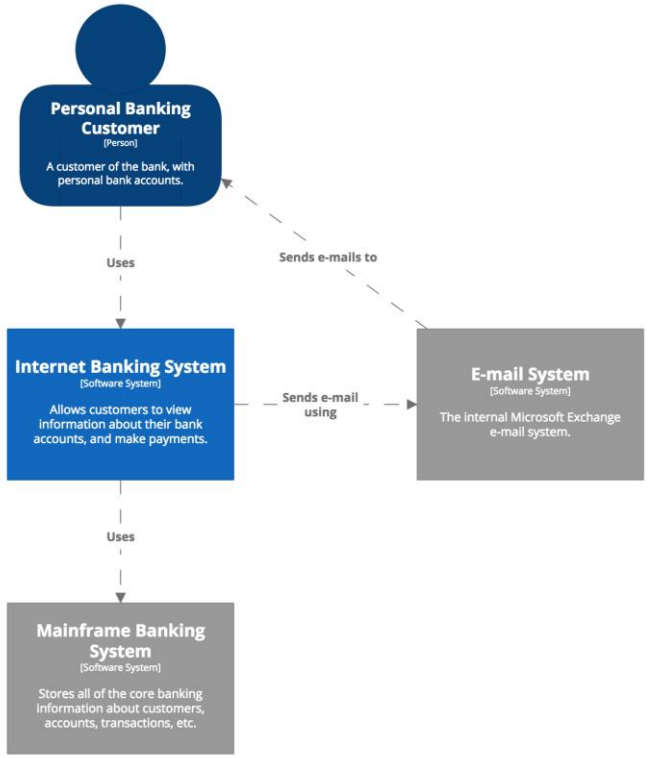
META MODELL
Diagrammelemente



ABSTRAKTIONSLEVEL

Level 1: System Context Diagram

- **Scope**: Ein einzelnes Software System
- **Primäre Elemente**: Das im Scope befindliche Software System
- **Unterstützende Elemente**: Personen und Software Systeme, die direkt mit dem im Scope befindlichen Software System in Verbindung
- **Zielgruppe**: Technische und nicht-technische Personen innerhalb und außerhalb des Entwicklungs-Teams



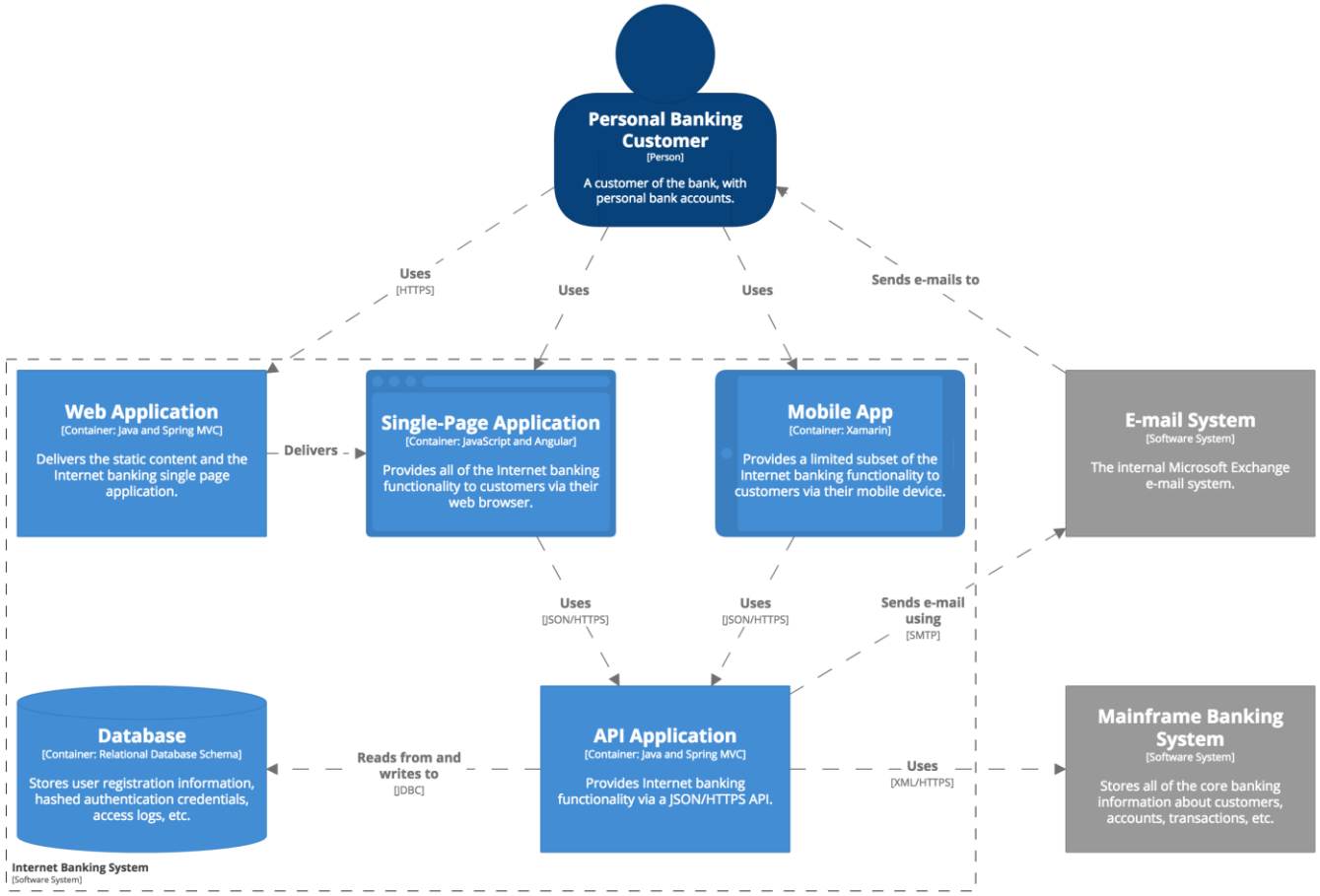
ABSTRAKTIONSLEVEL

Level 2: Container Diagram

- **Scope**: Ein einzelnes Software System
- **Primäre Elemente**: Der im Scope befindliche Container
- **Unterstützende Elemente**: Personen und Software System, die mit dem im Scope befindlichen Container in Verbindung stehen
- **Zielgruppe**: Technische Personen innerhalb und außerhalb des Software Entwicklungs-Teams; einschließlich Software Architekten, Entwickler und Betriebs- und Supportpersonal
- **Hinweis**: Container Diagramme treffen keine Aussage über die Deployment Szenarien, eventuelles Clustering, Replikations- und/oder Failover-Szenarien etc.

ABSTRAKTIONSLEVEL

Level 2: Container Diagram Beispiel

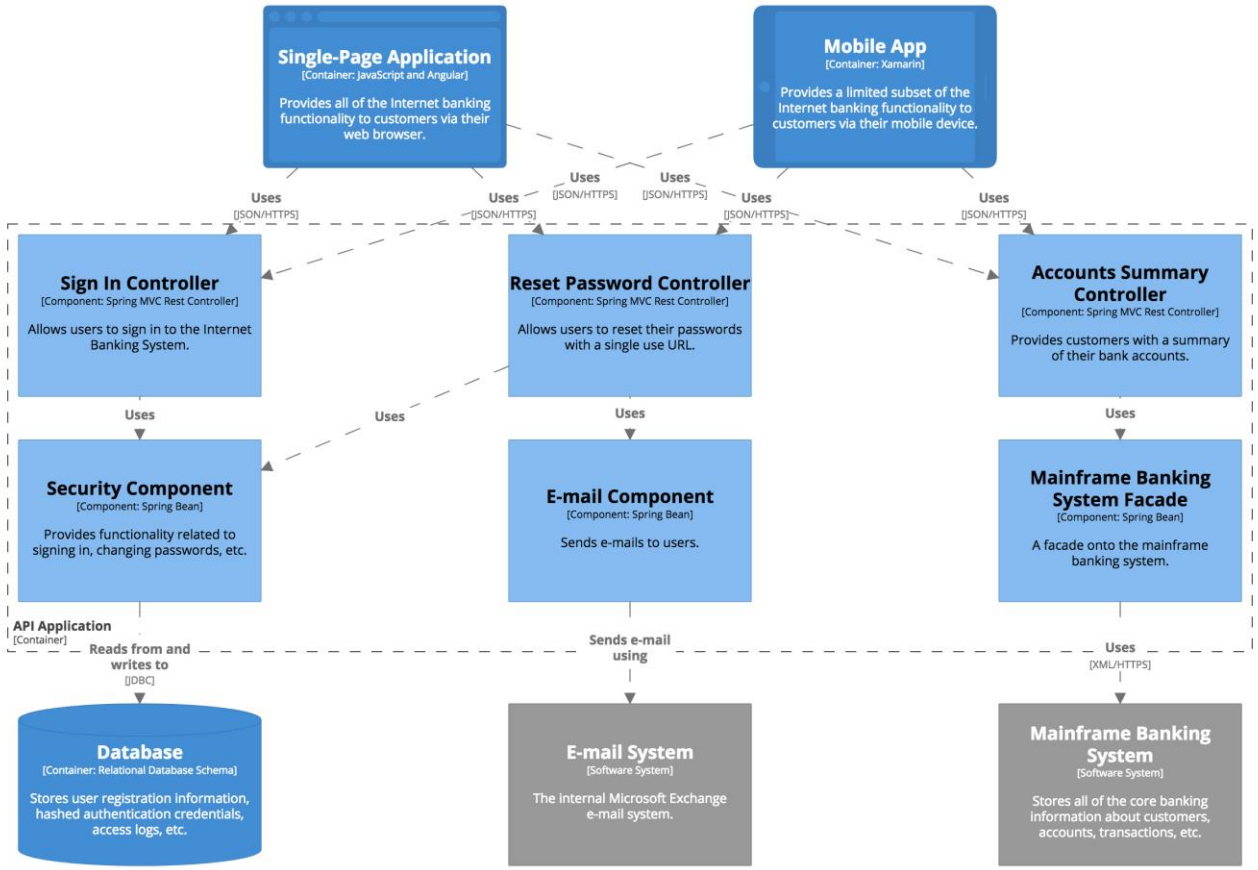


ABSTRAKTIONSLEVEL

Level 3: Component Diagram

- **Scope**: Ein einzelner Container
- **Primäre Elemente**: Komponenten innerhalbs des im Scope befindlichen Containers
- **Unterstützende Elemente**: Containers (innerhalb des im Scope befindlichen Software Systems) + Personen und Software Systeme, die direct mit den Komponenten in Verbindung stehen
- **Zielgruppe**: Software Architekten and Entwickler

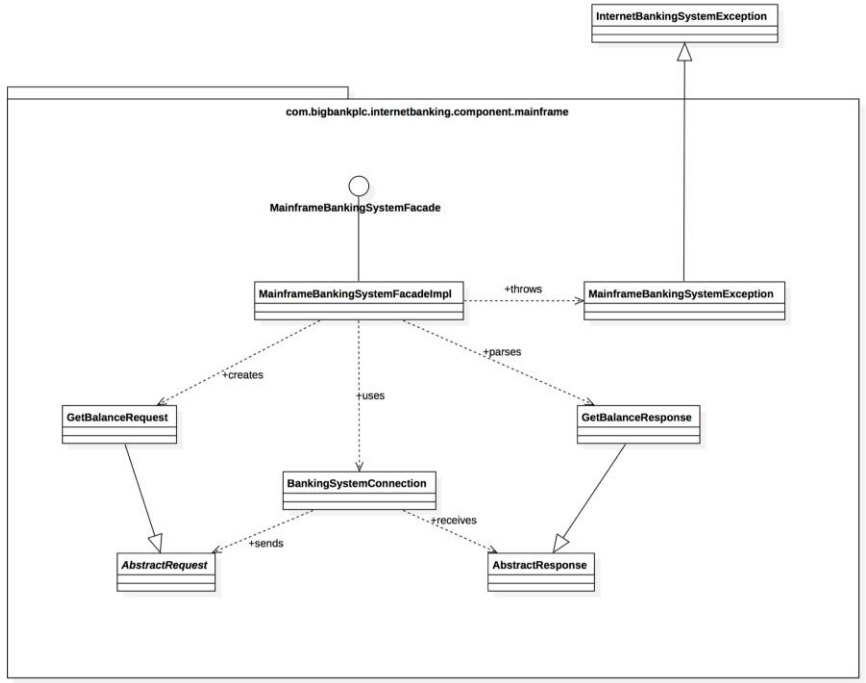
ABSTRAKTIONSLEVEL
Level 3: Component Diagram



ABSTRAKTIONSLEVEL

Level 4: Code Diagram

- **Scope**: Eine einzelne Komponente
- **Primäre Elemente**: Code Elemente (z.B. Klassen, Interfaces, Objekte, Methoden und Funktionen, Datenbanken und Tabellen etc.) innerhalb der im Scope befindlichen Komponente
- **Zielgruppe**: Software Architekten und Entwickler



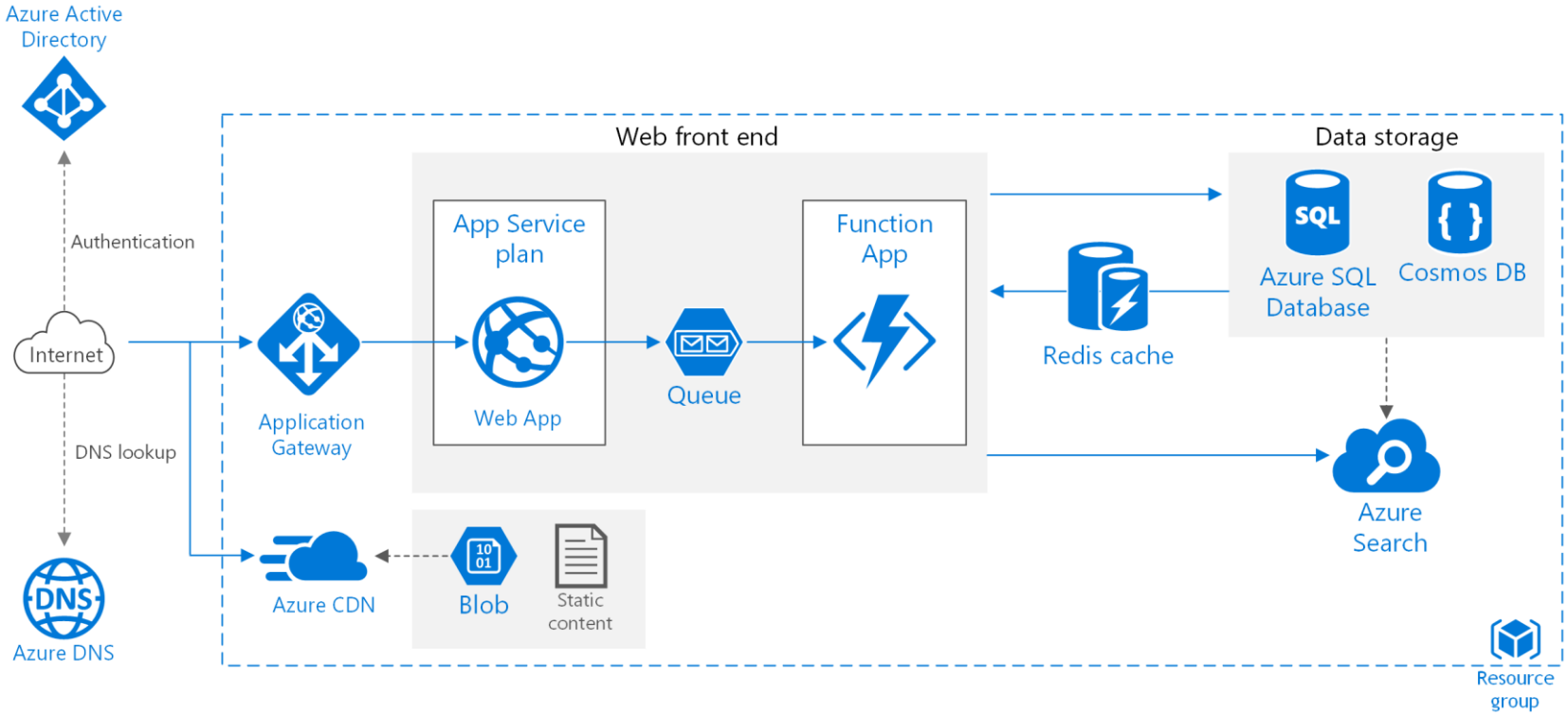
ABSTRAKTIONSLEVEL
Zusätzliche Diagramme

- Die 4 Level sind nicht immer ausreichend
- Wo nötig mit Komplementären ergänzen

- Systemlandschaft
  - In der echten Welt sind Software Systeme schon lange nicht mehr voneinander isoliert
  - Zeigen z.B. wie Systeme innerhalb eines Unternemens zusammenspielen
- Dynamische Diagramme
  - UML Kollaborationsdiagramme
  - UML Sequenzdiagamme
- Deployment Diagramme
  - UML Deployment Diagramme
    einschließlich Containers and Deployment Nodes

REALES BEISPIEL

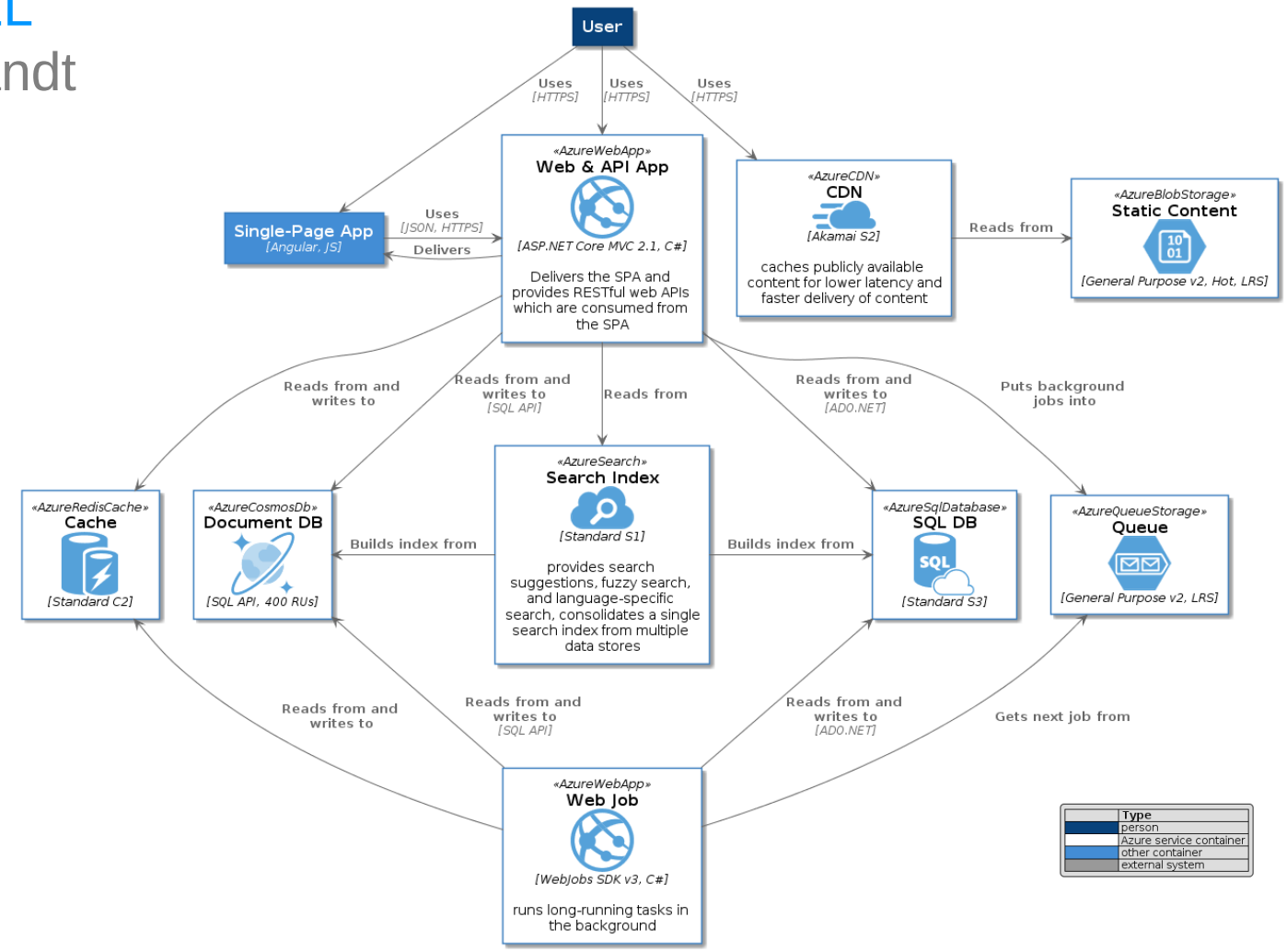
Referenz Architektur für Microsoft Azure App Service



Wüssten Sie wie das System zu bauen ist?

Quelle: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/app-service-web-app/scalable-web-app

REALES BEISPIEL
C4 Model angewandt



WEITERFÜHRENDE LITERATUR

Simon Brown
Software Architecture for Developers Volume 2
Leanpub Books, 2018
https://leanpub.com/visualising-software-architecture

Fragen bis hier her?