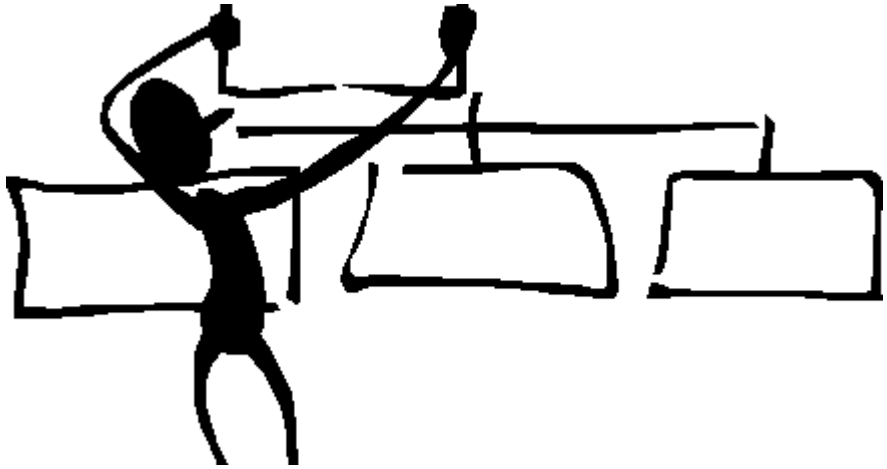


Exkurs

MVC UND „ZWIEBEL-ARCHITEKTUR“

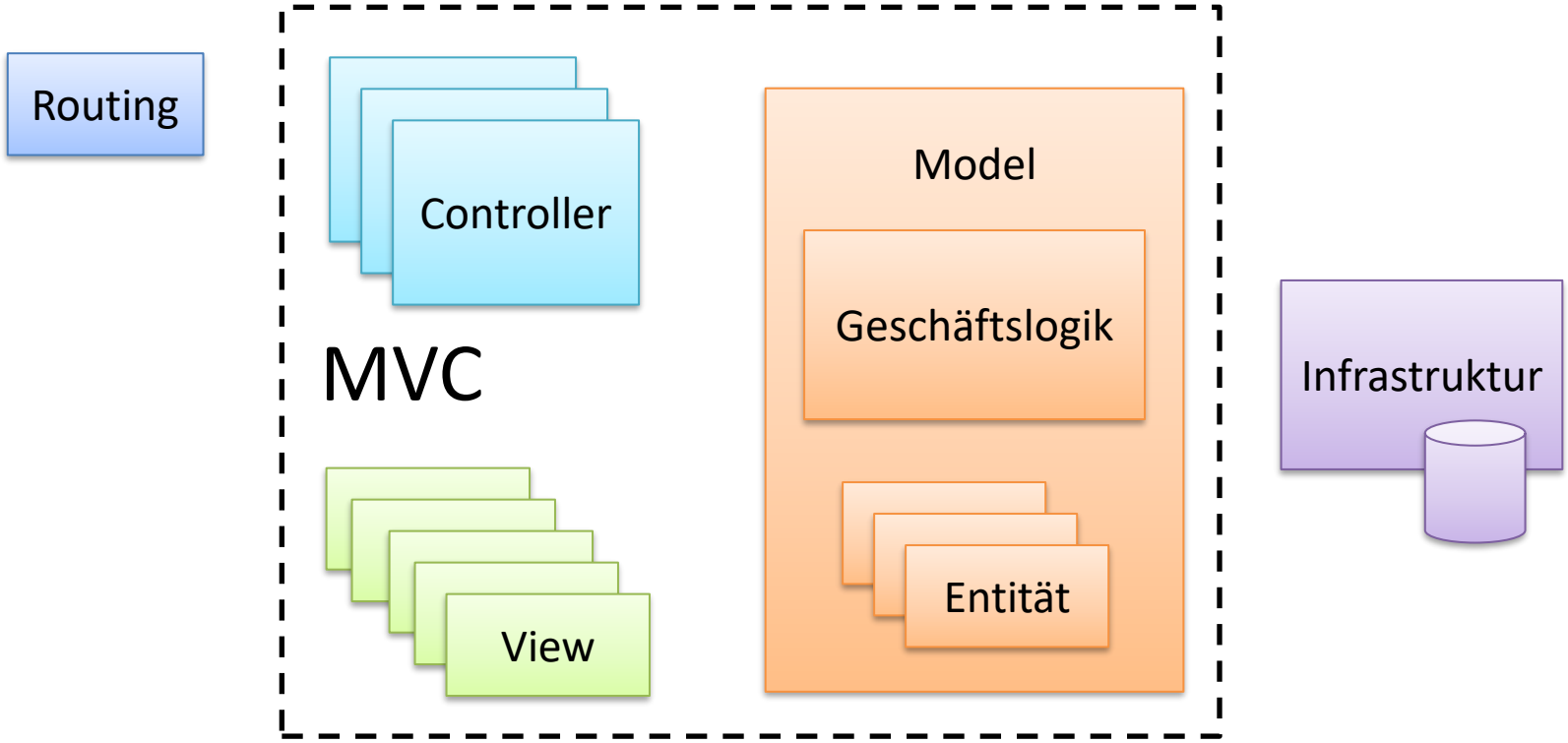
Model-View-Controller (MVC)

- Bewährtes Muster zur besseren Strukturierung von Anwendungen
  - Model: Definiert Geschäftslogik und Struktur der Daten
  - View: Definiert, wie Daten angezeigt werden
  - Controller: Verbindet Model und View und regelt logischen Fluss
- Ziele / Sinn:
  - Vereinfachung der Implementierung Trennung von Aufgaben: Logik vs. Anzeige vs. Aufruf
  - Möglichkeit zur Wiederverwendung einzelner Komponenten (z. B. selbes Model auf verschiedenen Plattformen)

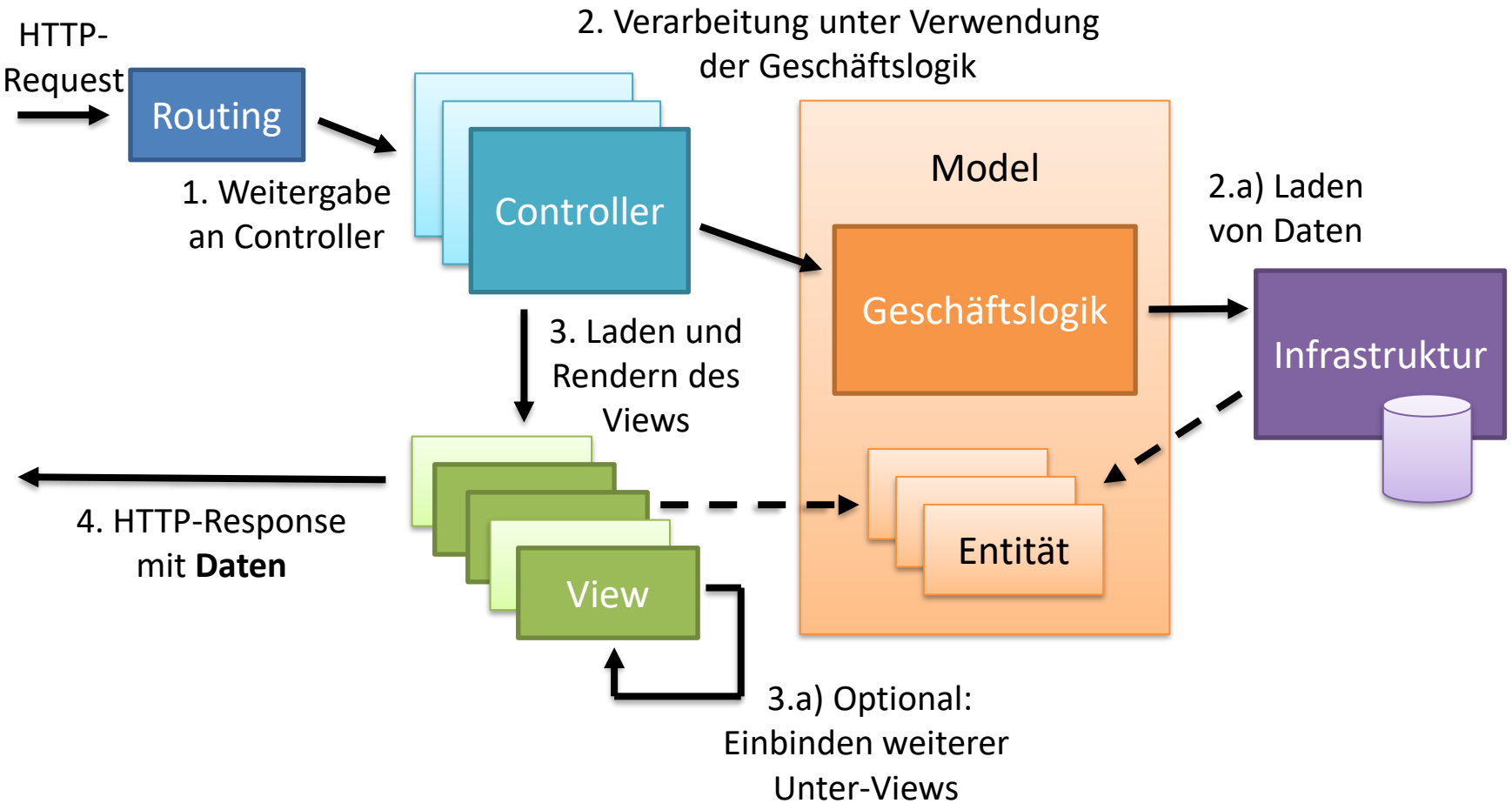
MVC im Web

- Model
  - Implementierung von Entitäten und Geschäftslogik
  - Technologieagnostisch (kein HTTP, kein HTML, keine Datenbanklogik, …)
- View
  - Template mit HTML, das von Controller aufgerufen wird und dynamisch Daten für Client rendert
- Controller
  - Nimmt Request entgegen, verarbeitet ihn unter Verwendung von Geschäftslogik und erzeugt Response
  - Moderator zwischen Model und View
- Routing-Logik
  - Stellt Requests auf Basis von URI an entsprechende Controller zu
- Infrastruktur
  - Plattformspezifische Implementierung von Funktionalität für Geschäftslogik
    - Benutzerauthentifizierung z. B. über Session oder Cookie
    - Speicherung von temporären Daten z. B. in Session von Webserver
    - Persistente Datenspeicherung z. B. in Datenbank oder externem System (über Webservice o. Ä.)

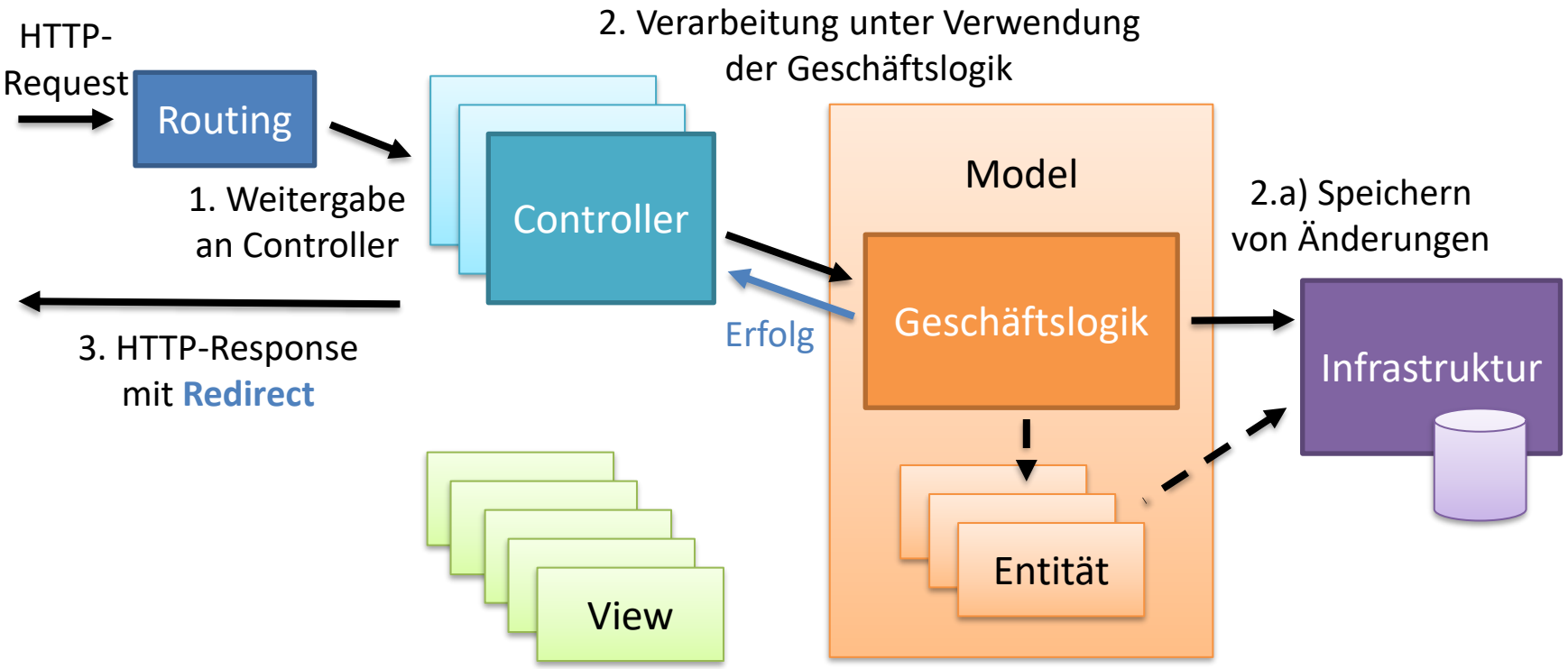
Komponenten einer Webanwendung auf Basis von MVC



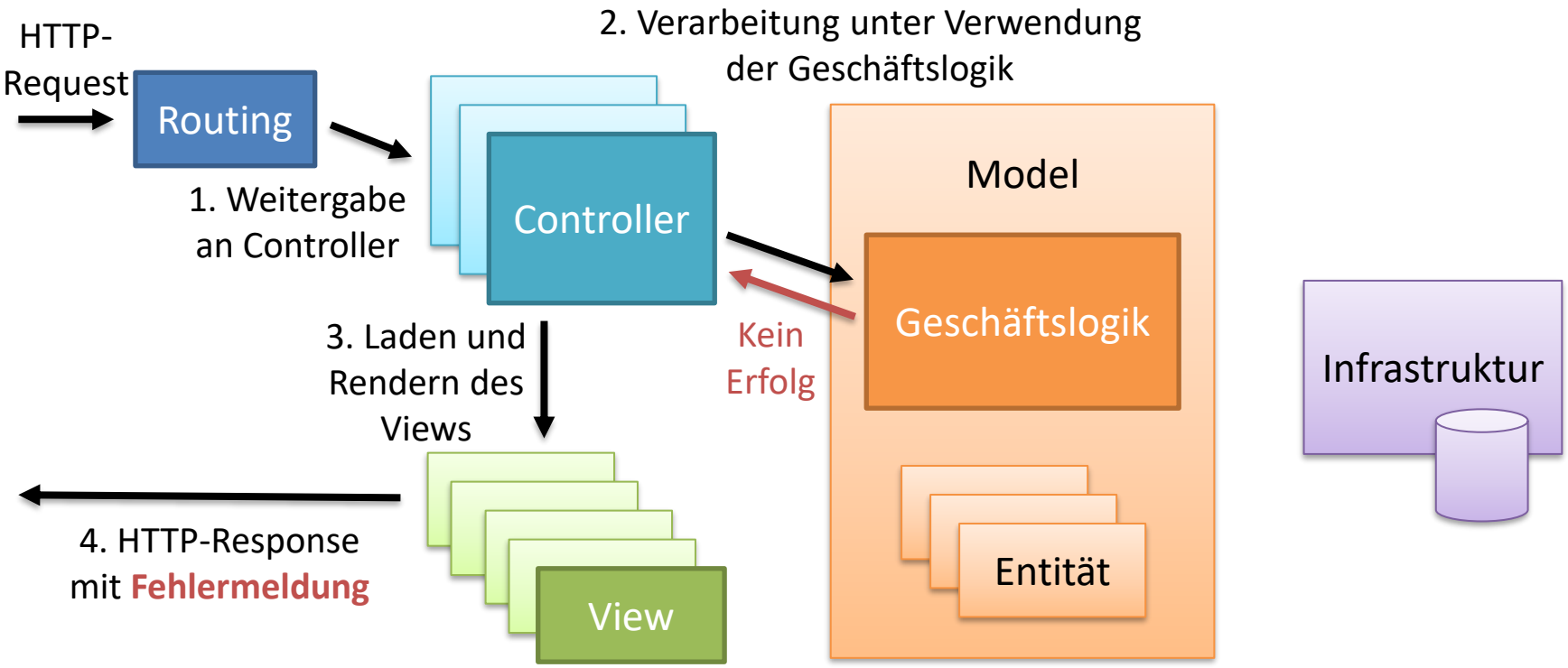
Beispiel: GET-Request



Beispiel: POST-Request

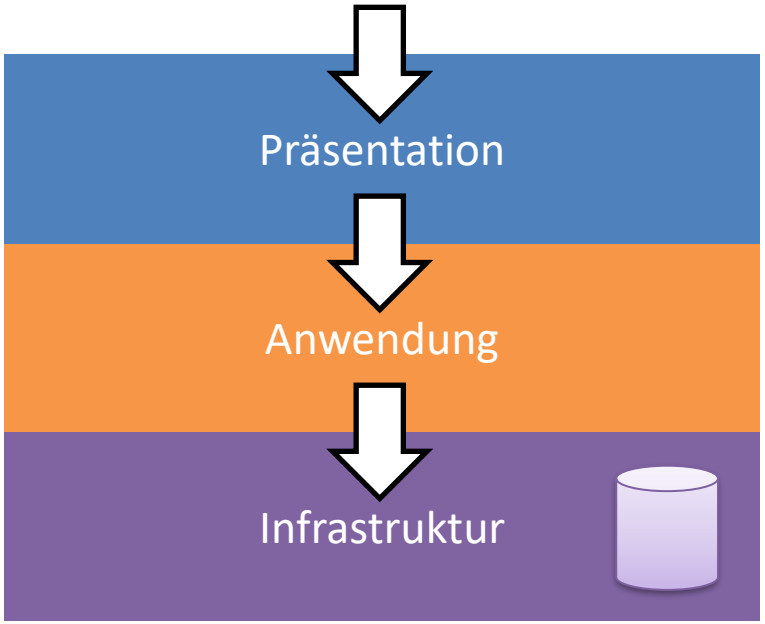


Beispiel: POST-Request (2)



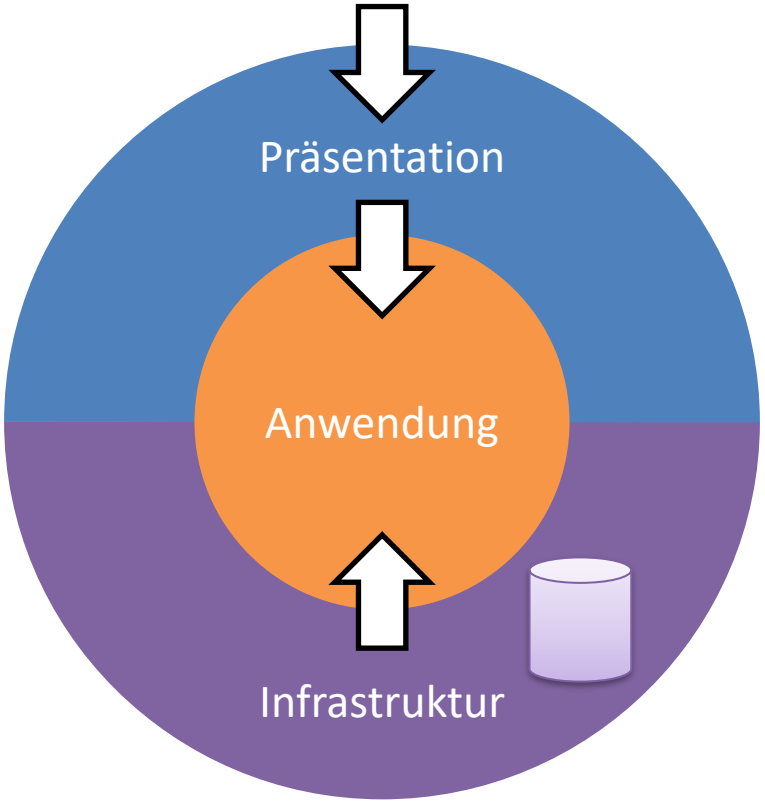
„Klassische“ Schichten-Architektur

- Hierarchisch
  - Präsentationslogik ist abhängig von Anwendungslogik
  - Anwendungslogik ist abhängig von Infrastruktur
- Anwendungslogik hat somit oft auch technologische Abhängigkeit zu Infrastruktur
  - Relationale Datenbank in Infrastruktur → OR-Mapper in Anwendungslogik usw.
- Änderungen in Infrastruktur können Auswirkungen auf Anwendungsschicht haben
  - Wechsel des Datenbanksystems etc.
- Testbarkeit eingeschränkt
  - Um Anwendungslogik zu testen muss entsprechende Infrastruktur verfügbar sein



Zwiebel-Architektur

- Anwendungslogik ist zentrale Komponente
  - Unabhängig von Präsentation **UND** Infrastruktur
  - Somit auch unabhängig von eingesetzten Technologien
    - UI-Frameworks, Datenbanken, …
- Umkehrung der Abhängigkeit zwischen Infrastruktur und Anwendungslogik
  - **„Inversion-of-Control“ (IoC)**
  - Erreichbar durch Einsatz von **Dependency-Injection**
- Erhöht Austauchbarkeit und erleichtert das Testen
  - Fakes für Infrastruktur, …



Komponenten einer Webanwendung – Perspektive „Zwiebel-Architektur“