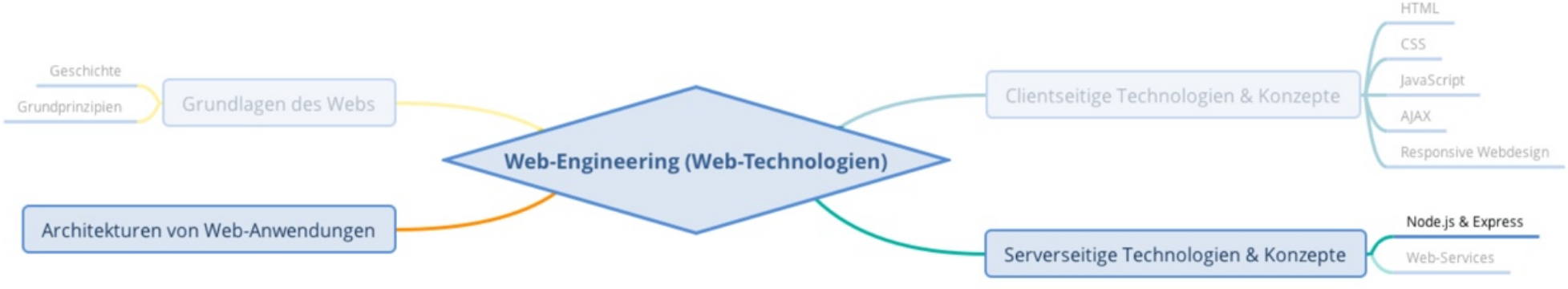
TRENNUNG VON ZUSTÄNDIGKEITEN

- Englisch *Separation of Concerns* (SoC)
- Jede Komponente eines Softwaresystems hat **eine** klar festgelegte Zuständigkeit
- Vorteile u.A.:
  - ⊕ Klarere Organisation und Struktur des Softwaresystems
  - ⊕ Verbesserte Änderbarkeit und Austauschbarkeit einzelner Aspekte des Softwaresystems
  - ⊕ Bessere Lokalisierung von Fehlern

THEMEN DER VERANSTALTUNG

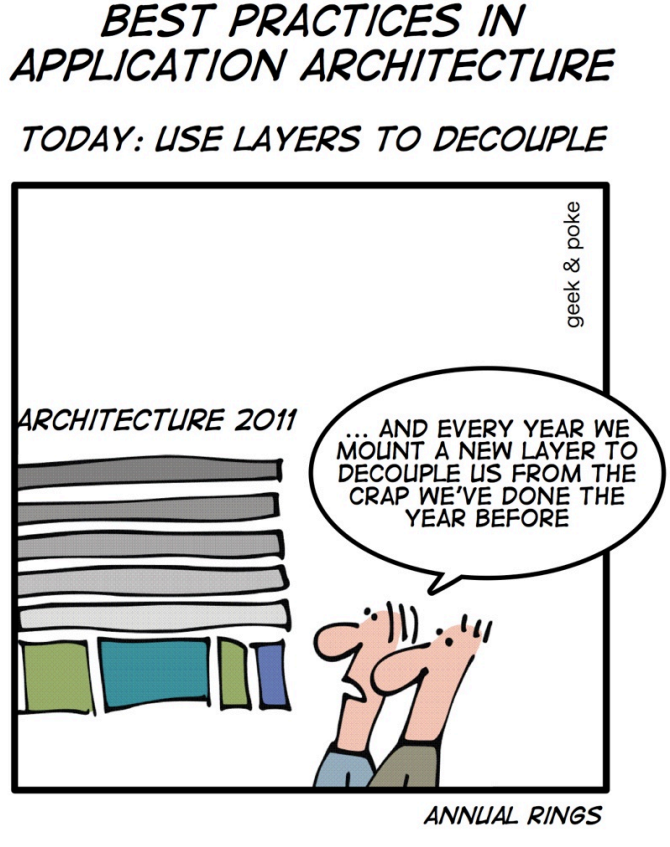


MEHRSCHICHTENARCHITEKTUR

- Ein klassisches Architekturmuster zur Trennung von Zuständigkeiten ist die Aufteilung des Softwaresystems in **Schichten**
- Jede Komponente wird genau einer Schicht zugeordnet
- Dabei abstrahiert jede Schicht von der darunter liegenden Schicht
- Typischerweise darf eine Schicht nur auf die nächst niedrigere Schicht zugreifen (*Schichten mit linearer Ordnung*)

MEHRSCHICHTENARCHITEKTUR (2)

Herausforderung: Wie viele Schichten braucht mein Softwaresystem?

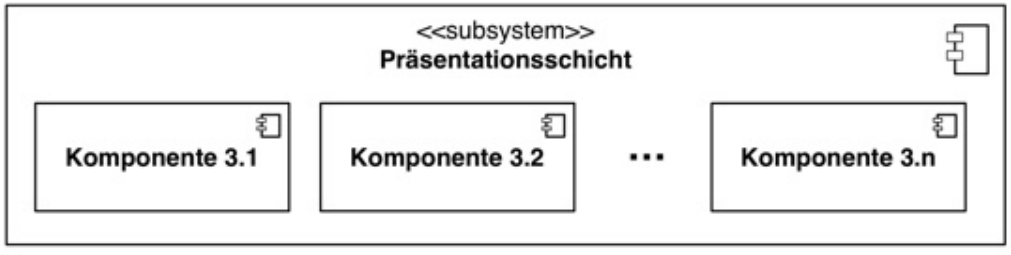


Quelle: Geek & Poke

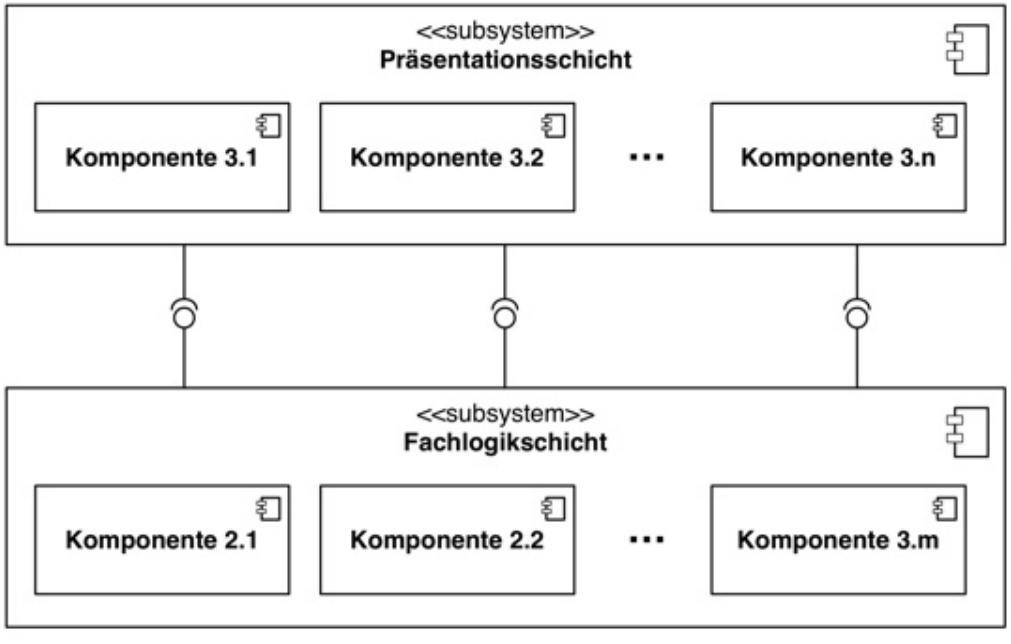
DREI-SCHICHTENARCHITEKTUR

- Englisch *three tier architecture*
- Klassische Ausprägung einer Mehrschichtenarchitektur
- Beinhaltet folgende drei Schichten:
  1. Präsentationsschicht (auch: GUI-Schicht)
  2. Fachlogikschicht (auch: Geschäftslogikschicht)
  3. Datenhaltungsschicht

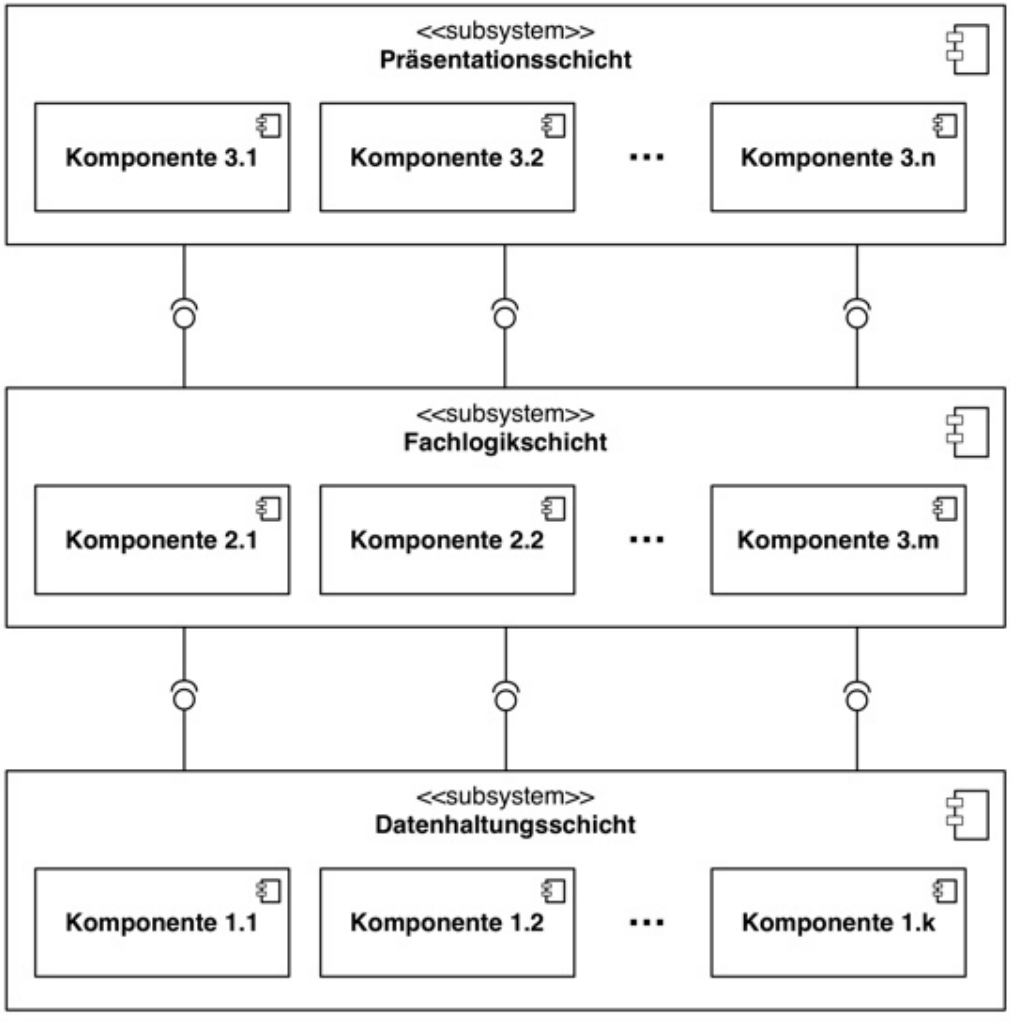
DREI-SCHICHTENARCHITEKTUR (2)



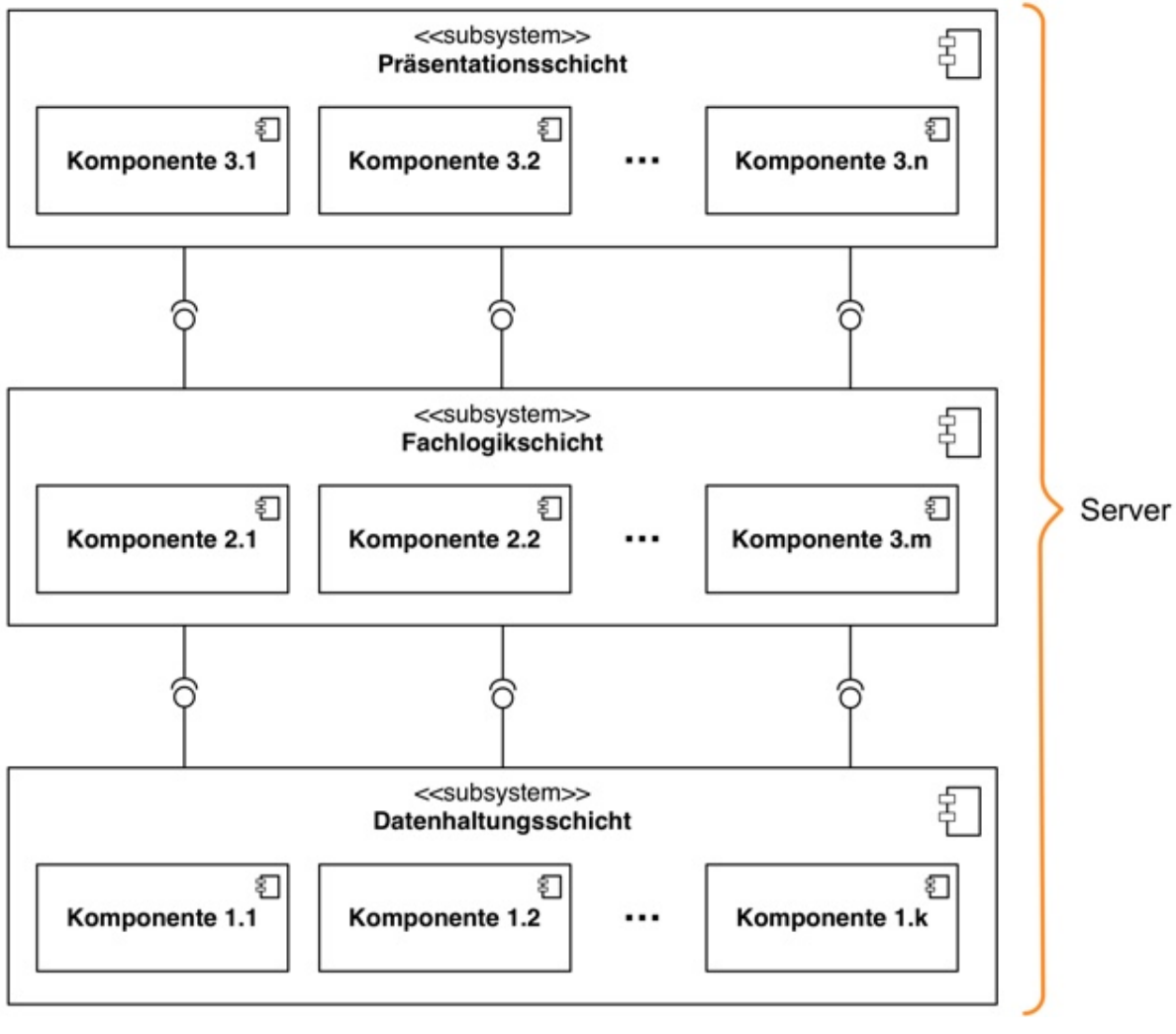
DREI-SCHICHTENARCHITEKTUR (2)



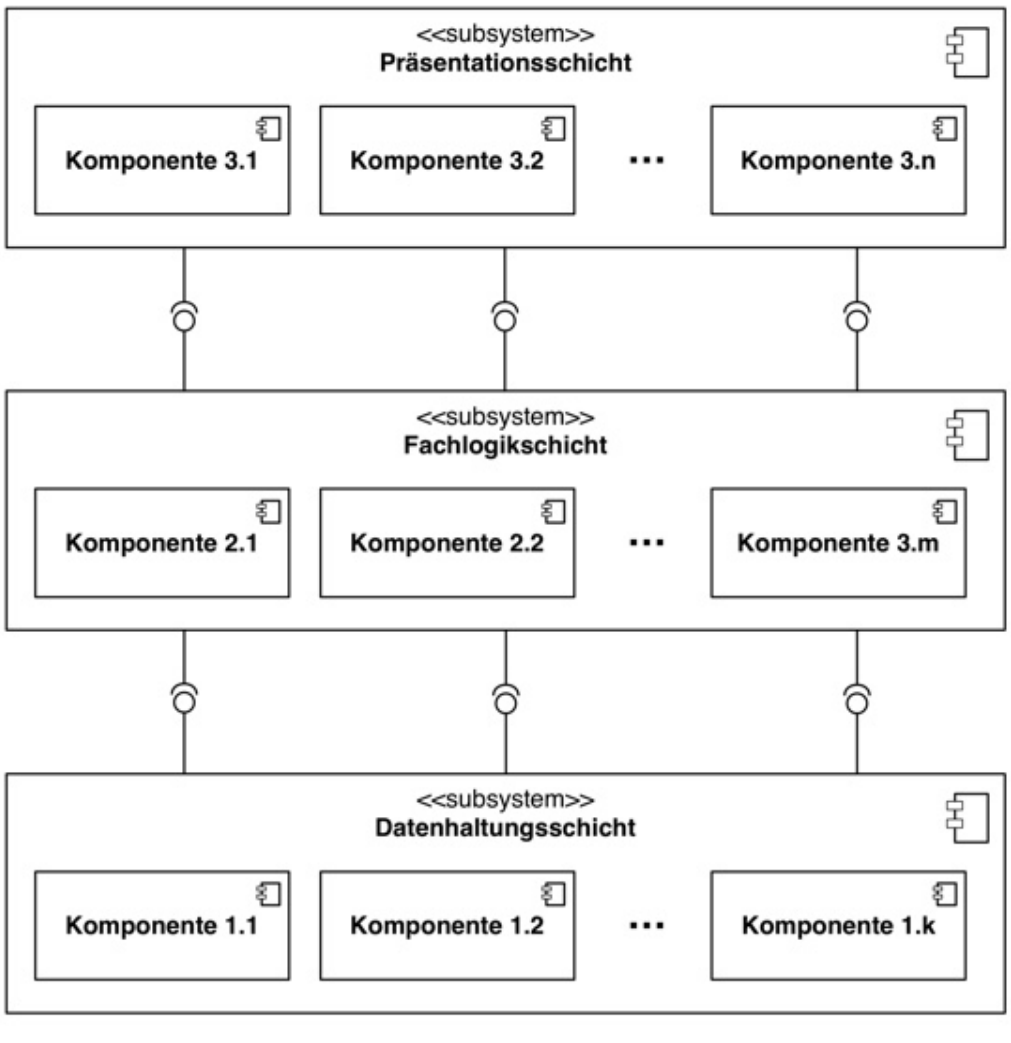
DREI-SCHICHTENARCHITEKTUR (2)



DREI-SCHICHTENARCHITEKTUR (2)

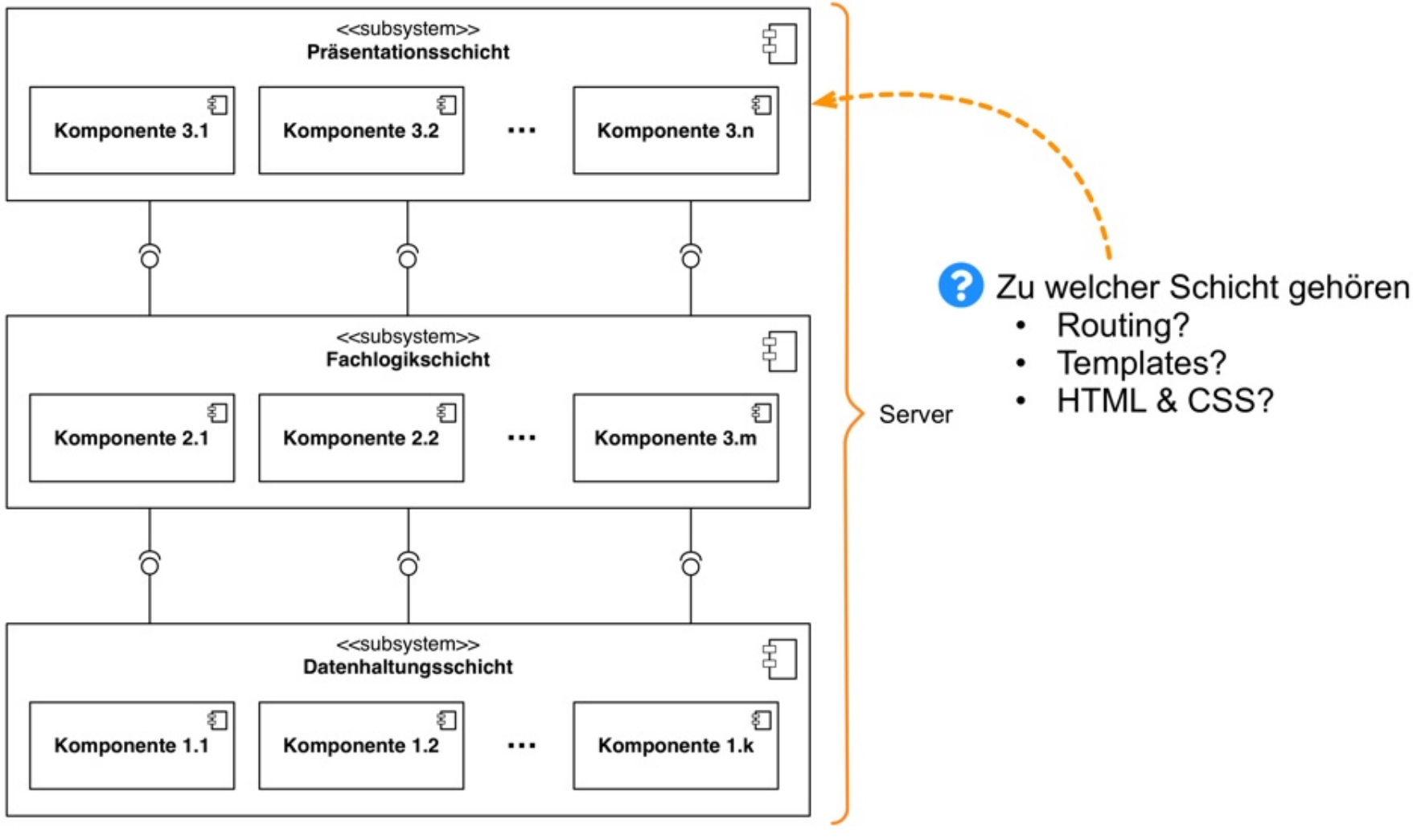


DREI-SCHICHTENARCHITEKTUR (2)

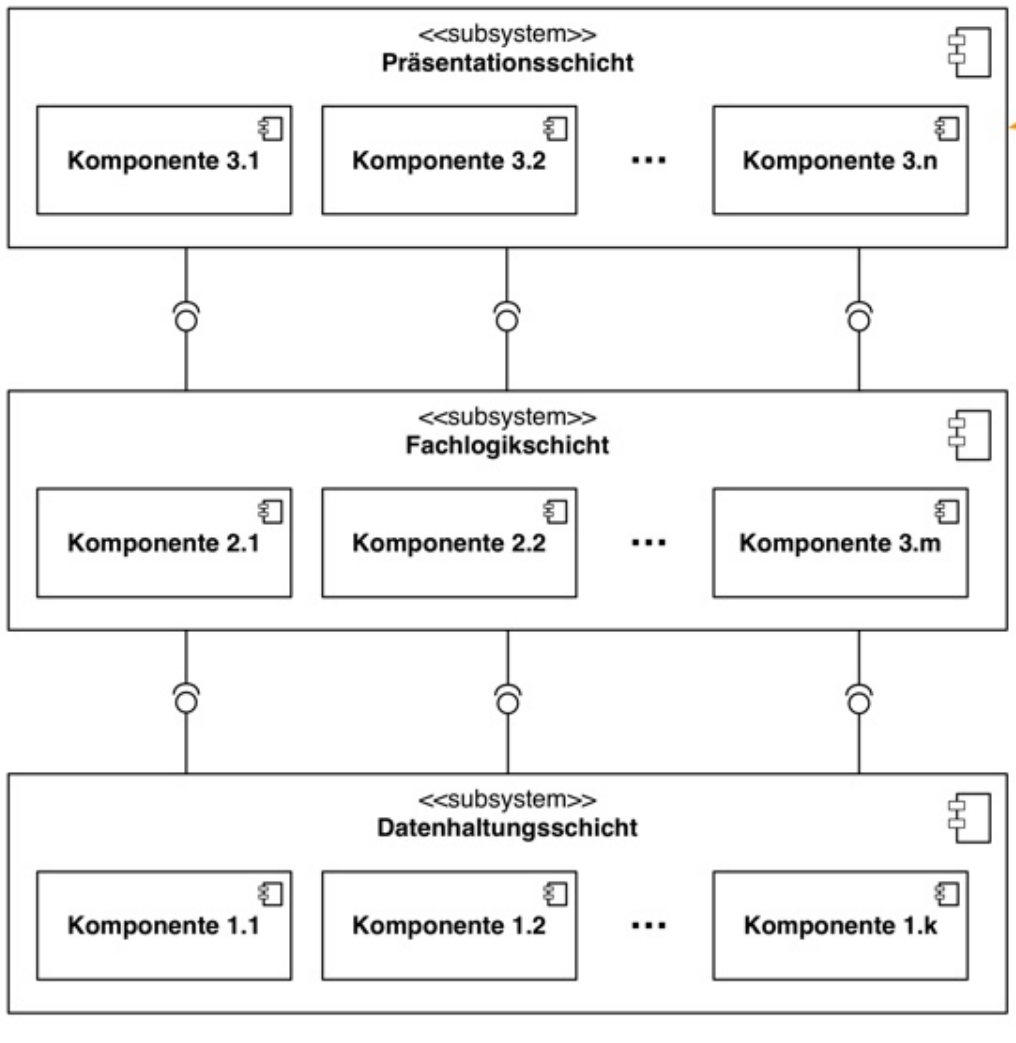


Zu welcher Schicht gehören
- Routing?
- Templates?
- HTML & CSS?

DREI-SCHICHTENARCHITEKTUR (2)



DREI-SCHICHTENARCHITEKTUR (2)



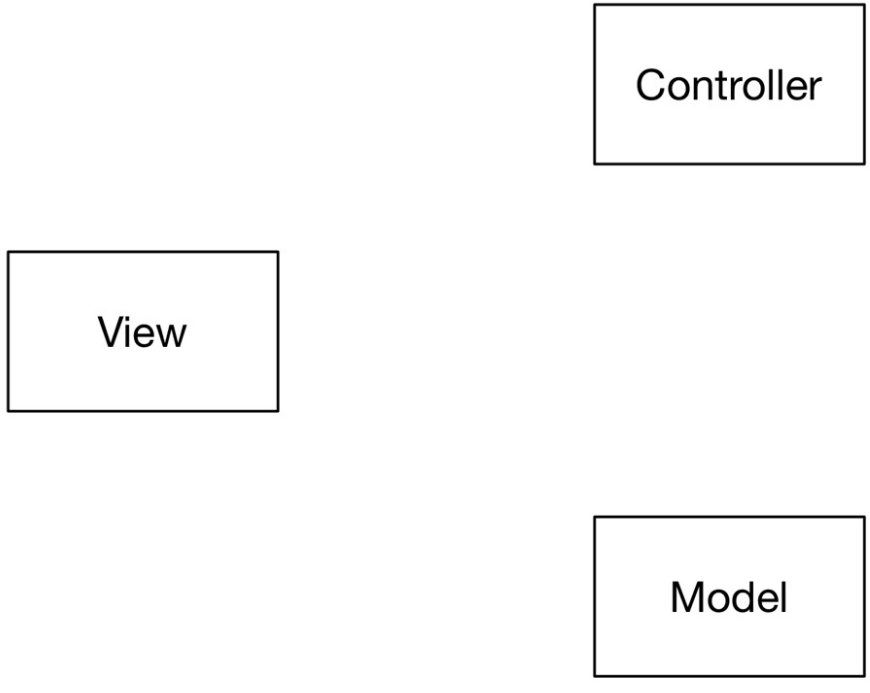
Zu welcher Schicht gehören
- Routing?
- Templates?
- HTML & CSS?

Unser Gesamtsystem ist zwar modularisiert, jedoch: Innerhalb der Präsentationsschicht fehlt uns noch eine saubere Trennung der Zuständigkeiten.

MODEL-VIEW-CONTROLLER (MVC)

- Architekturmuster zur Strukturierung der Präsentationsschicht
- Ziele:
  - Trennung von Zuständigkeiten
  - Anpassbarkeit und Erweiterbarkeit
  - Wiederverwendbarkeit (z.B. gleiches Model für andere Controller und Views)

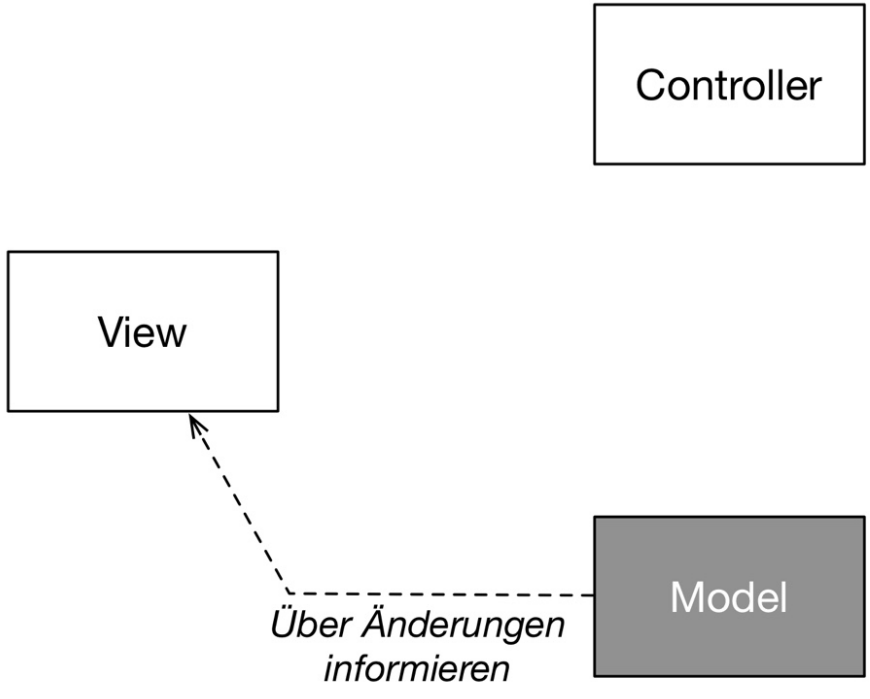
MVC: AUFBAU



MVC unterteilt die Komponenten der Präsentationsschicht in drei Typen:

1. Model
2. View
3. Controller

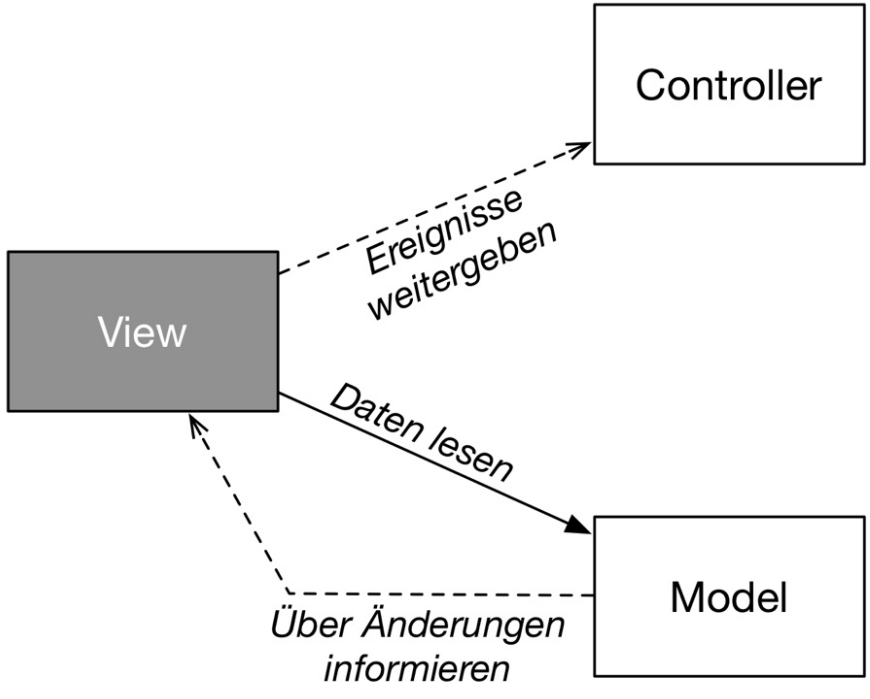
MVC: AUFBAU



Model:

- Datenmodell für die Oberfläche
- Informiert den View über Änderungen an den Daten
- Aufruf von Fachlogik (d.h. Kommunikation mit der Fachlogikschicht)

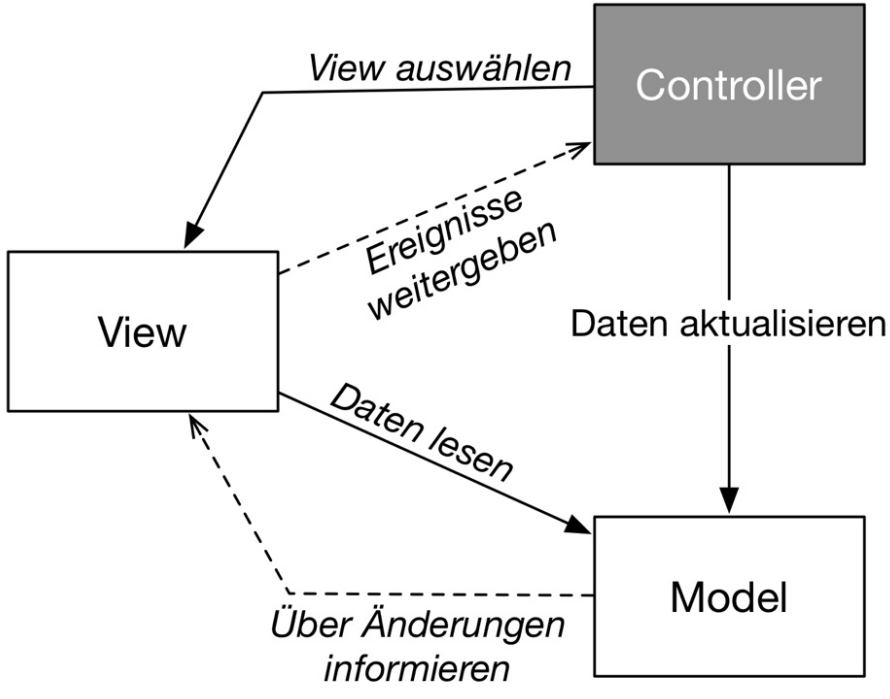
MVC: AUFBAU



View:

- Darstellung der Oberfläche (GUI)
- Weitergabe von Ereignissen an den Controller (z.B. Eingaben, Klick auf einen Button)
- Anwenden von Änderungen im Model

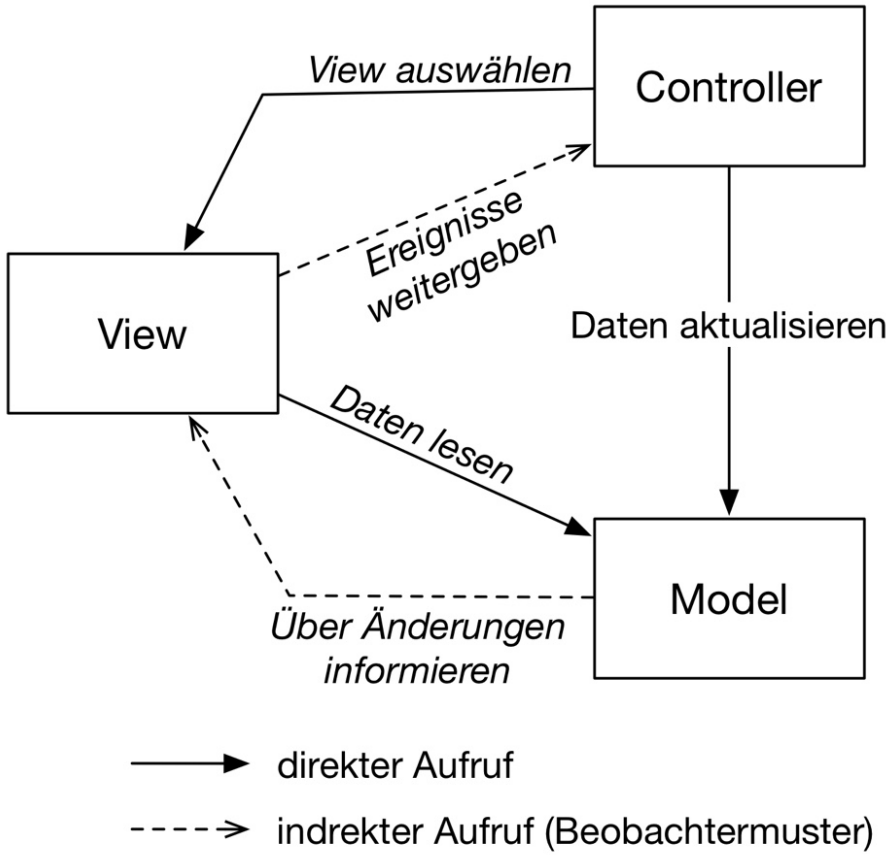
MVC: AUFBAU



Controller:

- Auswertung von Ereignissen und Durchführen entsprechender Aktionen
- Ablaufsteuerung (z.B. Öffnen einer View)

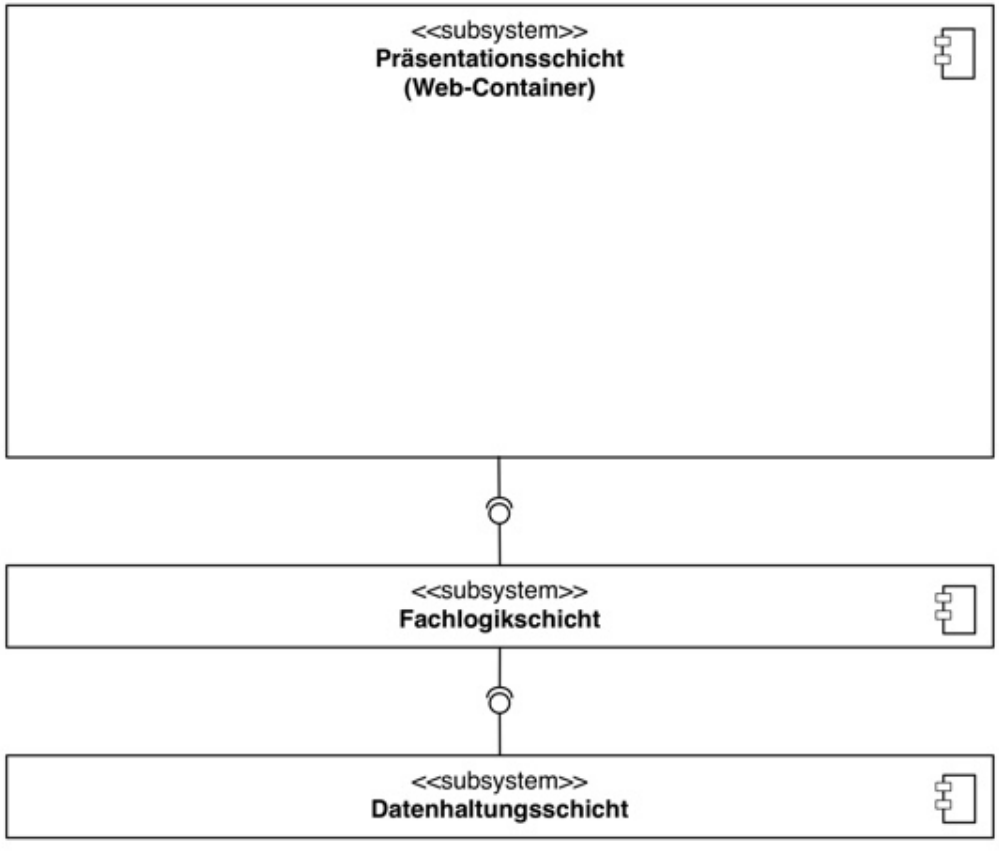
MVC: AUFBAU



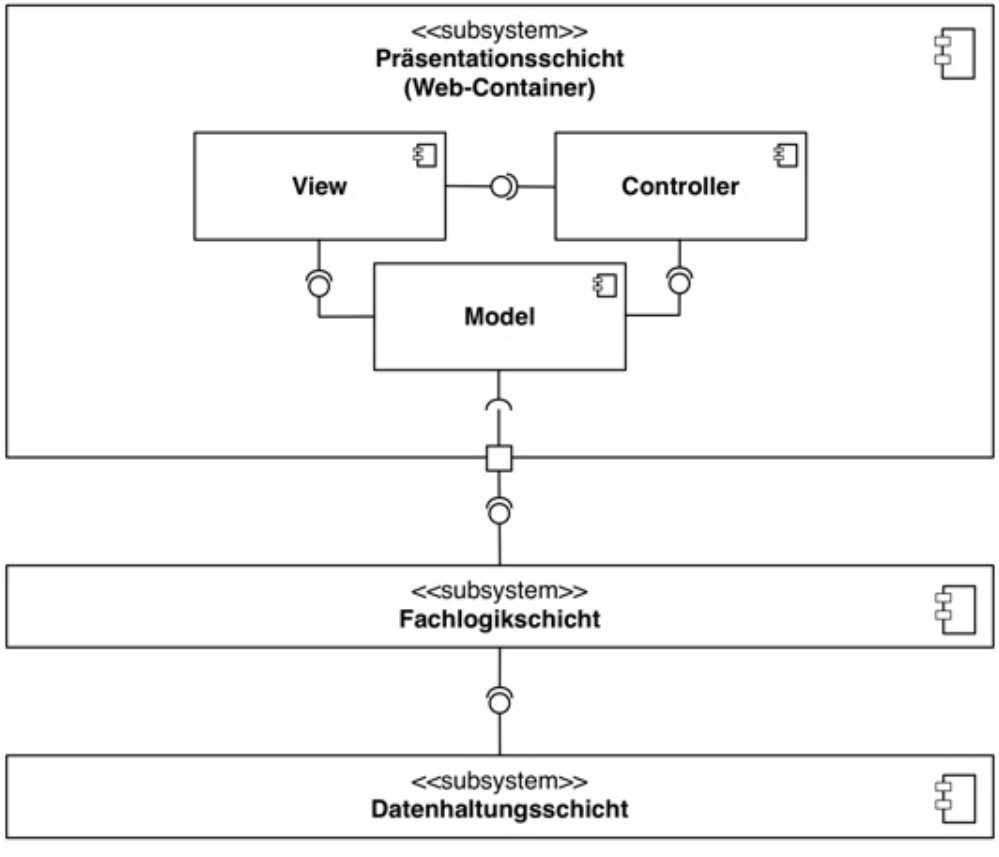
- Häufig erfolgt eine Entkopplung von Model und View durch Einsatz des **Beobachter-Musters** (*observer pattern*)
- Hier registriert sich die View beim Model als Beobachter, um über Änderungen am Model informiert zu werden
- Ebenso ist eine Entkopplung von View und Controller möglich

❗ Ergebnis: Model ist unabhängig von View und Controller, View ist unabhängig vom Controller (Austauschbarkeit, Wiederverwendbarkeit)

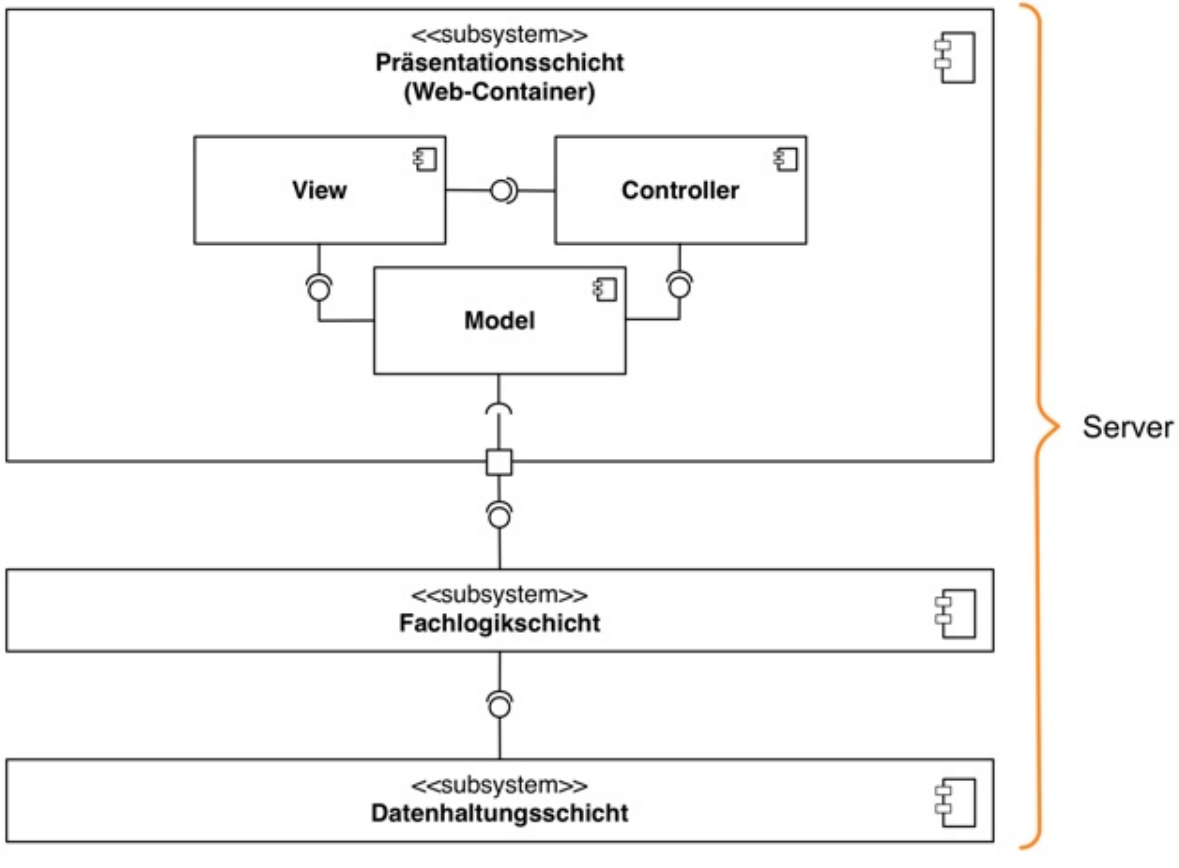
MVC IM WEB (SERVERSEITIG)



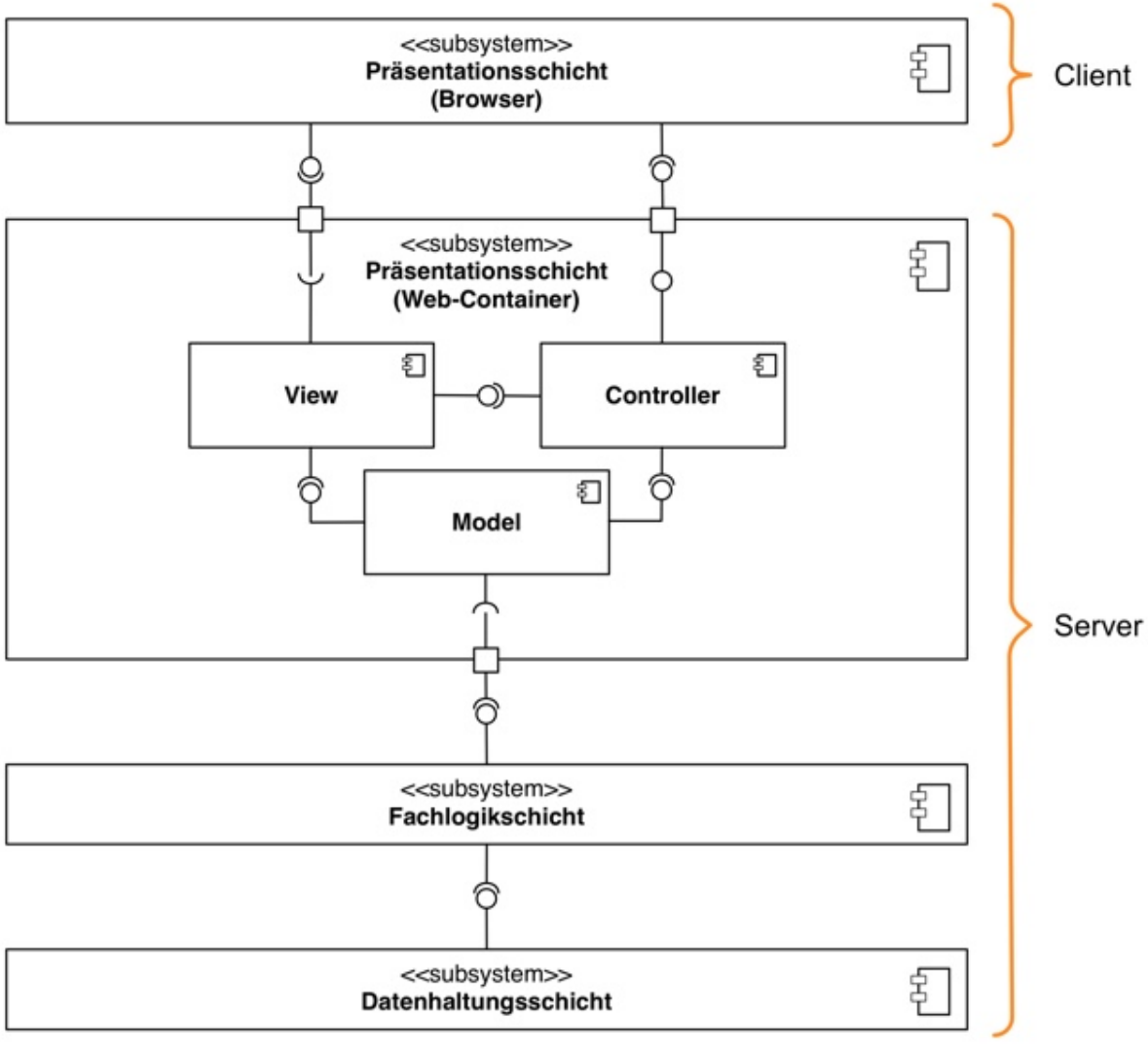
MVC IM WEB (SERVERSEITIG)



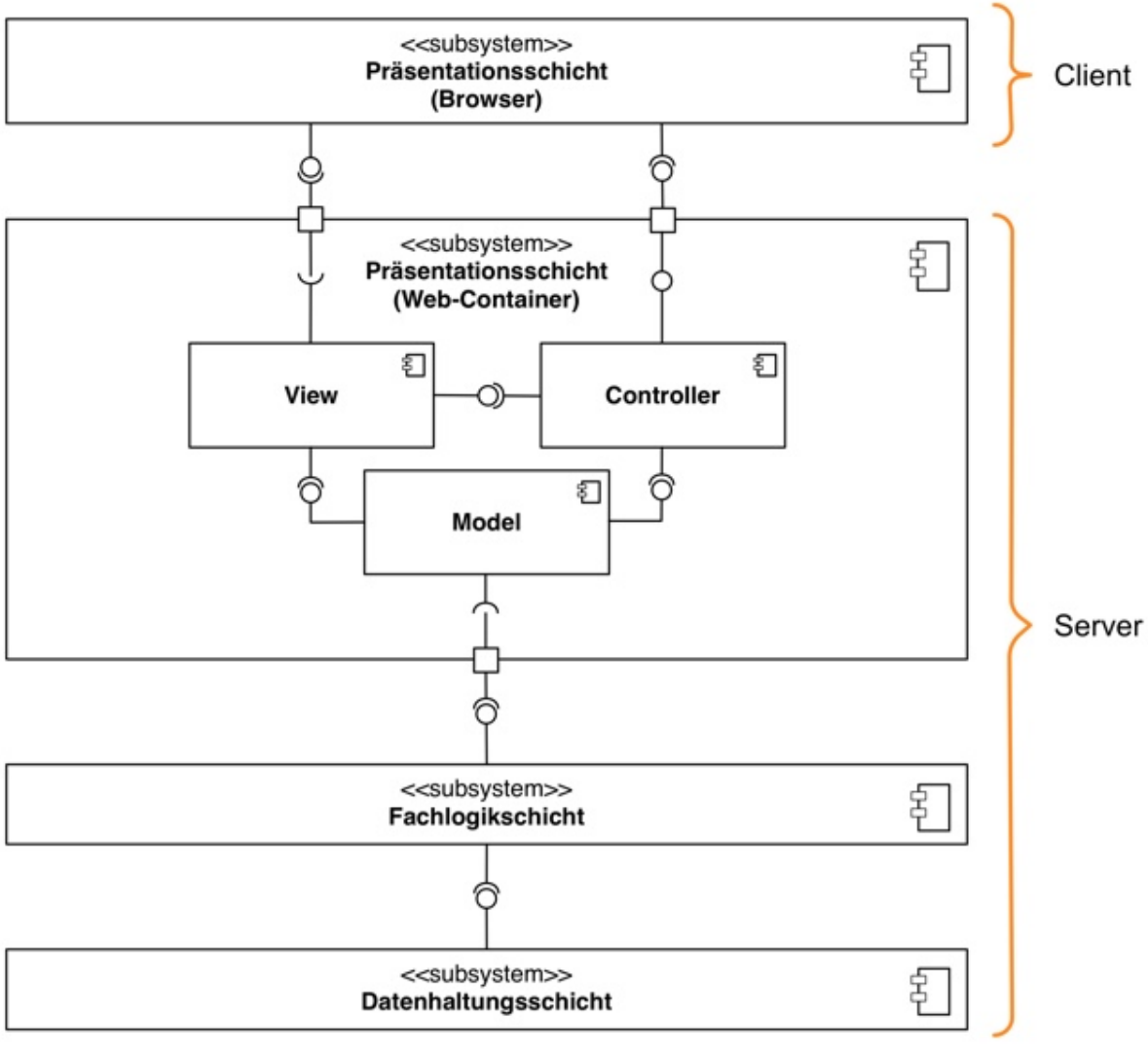
MVC IM WEB (SERVERSEITIG)



MVC IM WEB (SERVERSEITIG)



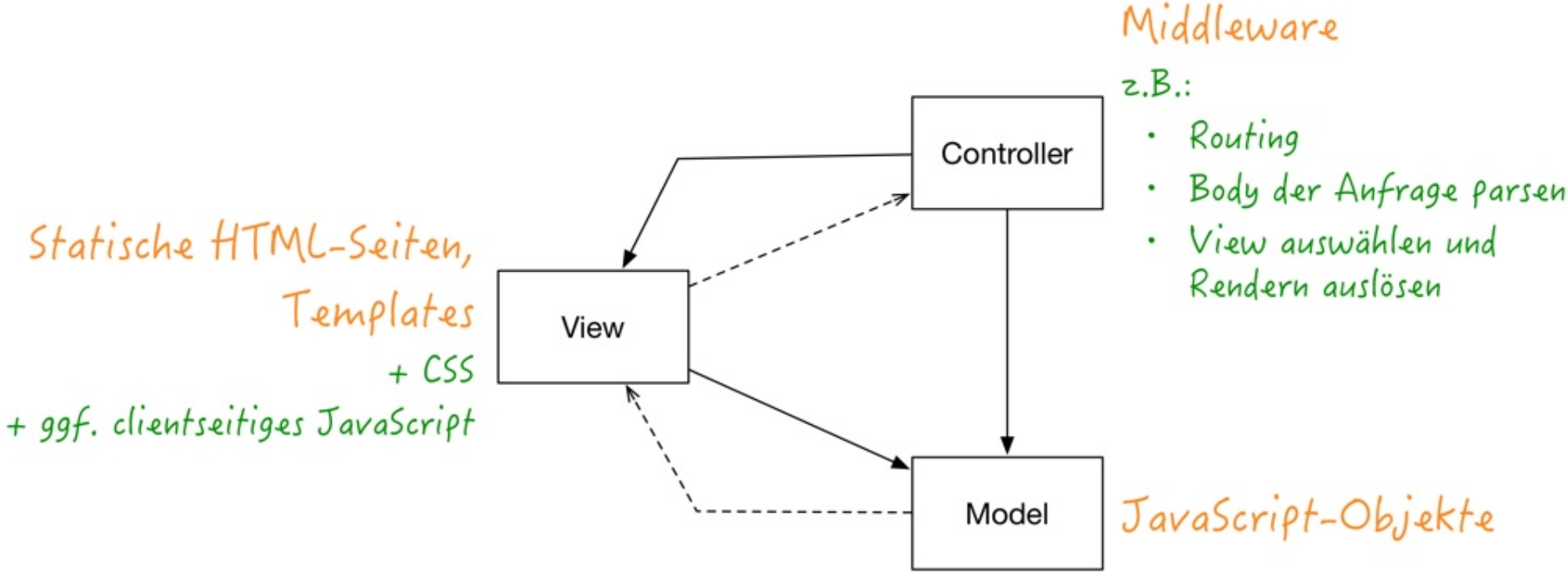
MVC IM WEB (SERVERSEITIG)



- Die Präsentationsschicht ist auf Client und Server verteilt:
  - Client: HTML-Darstellung
  - Server: HTML-Erzeugung, Präsentationslogik, Zustandsverwaltung, etc.

MVC MIT EXPRESS

Eine mögliche Umsetzung des MVC-Musters mit Express:



MVC: VARIANTEN

Es gibt weitere Implementierungsvarianten für MVC.

Express schreibt die Benutzung von MVC nicht vor und legt daher auch nicht fest, wie es zu implementieren ist.

Es gibt Web-Frameworks, die jeweils "ihre Sicht von MVC" vorgeben.

z.B. Spring MVC , JSF

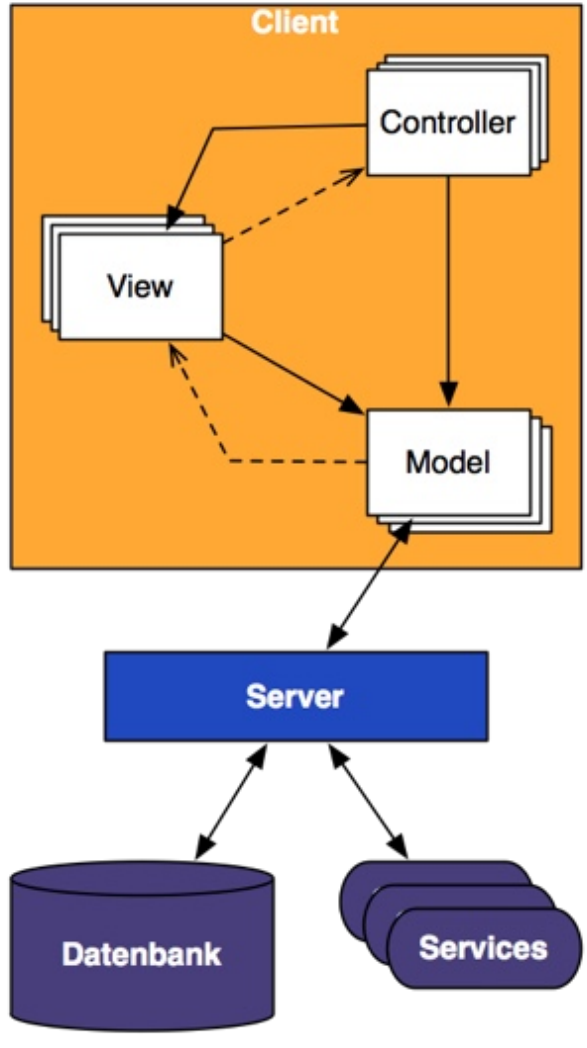
MVC: VARIANTEN (2)

MVC kann auch clientseitig angewendet werden!

z.B. JavaScript-Frameworks wie Ember.js

Es gibt zahlreiche Varianten des MVC-Musters.

z.B. Model-View-Viewmodel (MVVM), Model-View-Presenter (MVP)



ES GIBT AUCH GEGENLÄUFIGE TRENDS...

Beispiel: Komponenten in React

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}
```

Quelle: Intro to React