REST: Theoretische Grundlagen (3 Punkte)

1. Lesen Sie Abschnitt 2.3: Erklären Sie die Eigenschaften nach denen ein Architektur-Stil in Fielding's Arbeit beurteilt wird.
2. Lesen Sie Abschnitt 5.1: Welches sind die Beschränkungen für den REST-Stil und welche Auswirkungen haben Sie auf die Eigenschaften von Architekturen?

Aufgabe 1

Payload = Body

Aufgabe 2

REpresentational State Transfer ist ein Architekturstil für verteilte Systeme, insbesondere Webservices. Es ist eine Abstraktion der Struktur und des Verhaltens des WWW (HTTP). Fielding gibt nur abstrakte Bedingungen vor und keine expliziten Prozesse, Protokolle oder Medien. Der Fokus liegt auf Skalierbarkeit, Erweiterbarkeit und Interoperabilität.

1. Client-Server Modell

- Anforderung, dass alle Eigenschaften der Client-Server-Architektur gelten
- Server stellt einen Dienst bereit, der vom Client angefragt werden kann
- durch Kapselung können Client und Server unabhängig vomneinander entwickelt werden
- Server fällt schlanker aus und gewinnt an Skalierbarkeit

2. Stateless

- jeder Request ist komplett isoliert voneinander und in sich abgeschlossen
- jeder Request hat somit alle notwendigen Informationen um den Request serverseitig zu verstehen
- weder Server noch Client speichert Zustandsinformationen zwischen Nachrichten
- positive Wirkung auf Stabilität und Skalierbarkeit, da keine seperaten Sitzungsdaten benötigt werden
- HTTP ist ein zustandsloses Protokoll

3. Caching

- senkt die Latenz und das aufkommene Datenvolumen
- erlaubt es Daten einer Response für equivalente Requests wiederzuverwenden

4. Uniform Interface

Ziel ist die Einheitlichkeit der Schnittstelle und somit ihre einfache Nutzung

Adressierbarkeit von Ressourcen

- da REST eine ressourcenorientierte Architektur ist, muss man durch Ressourcen über ein schlankes, einheitliches Interface navigieren können
- jede REST-Schnittstelle hat eine eindeutige Adresse (URL - Uniform Resource Locator) um auf das Angebot des Webservices zugreifen zu können
- jede Information, die über einen URI kenntlich gemacht wurde, wird als Ressource gekennzeichnet

- Ressourcen müssen identifizierbar bzw. adressierbar gemacht werden über URI

Selbstbeschreibende Nachrichten

- REST Nachrichten sollen selbstbeschreibend sein z.B. über die Verwendung von HTTP Verben
- darüber lassen sich Ressourcen manipulieren
- Applikationen sind nicht an ein Protokoll gebunden
- im Falle von HTTP wäre Uniform Interface die HTTP-Verben GET, POST, PUT, DELETE
- sehr einfache und übersichtliche Schnittstelle

Repräsentationen zur Veränderung von Ressourcen

- die unter einer Adresse zugänglichen Dienste können unterschiedliche Darstellungsformen haben
- der REST-konforme Server kann dann, je nachdem der Client anfordert, verschiedene Repräsentationen einer Ressource ausliefern
- Formate können z.B. JSON, XML, HTML etc. sein

„Hypermedia as the Engine of Application State“ (HATEOAS)

- der Begriff HATEOAS bezieht sich auf jeglichen Content welcher Links zu anderen Formen von Medien wie Bilder oder Text enthält
- dadurch kann der Client über Ressourcen traversieren
- ähnlich wie im WWW, wo man sich durch Links zum Ziel klickt

```
{
    "departmentId": 10,
    "departmentName": "Administration",
    "locationId": 1700,
    "managerId": 200,
    "links": [
        {
            "href": "10/employees",
            "rel": "employees",
            "type" : "GET"
        }
    ]
}
```

---

5. Layered System

- die Systeme sollen mehrschichtig aufgebaut sein
- reicht dem Anwender lediglich eine Schnittstelle anzubieten
- dahinterliegende Ebenen können verborgen bleiben und somit die Architektur insgesamt vereinfacht werden

6. Code on Demand

- optional

- erst im Bedarfsfall kann an den Client Code zur lokalen Ausführung übertragen werden