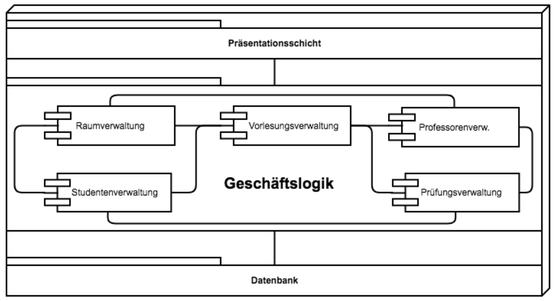
Microservices CheatSheet

Max Jando, Joshua Vécsei

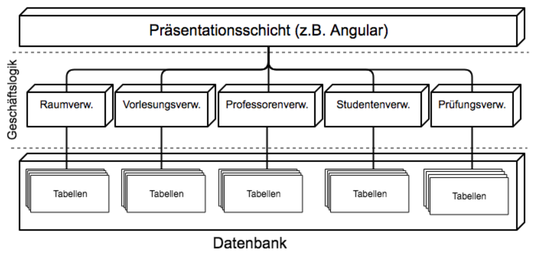
Hochschule Mannheim, Fakultät für Informatik

Monolitische Hochschulanwendung



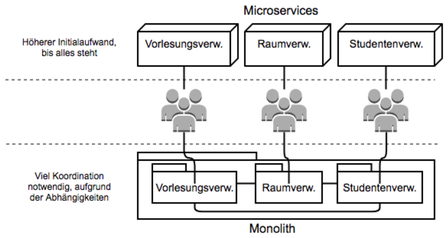
- Datenbank, UI und Anwendungslogik in einem Programm - *Full Stack Anwendung*
- Bei einem Update eine Teilkomponente muss der ganze Monolith aktualisiert werden
- Nur der ganze Monolith kann skaliert werden
- Weiterentwicklungen können unter Umständen schwer getestet werden, da die Komponenten miteinander verzahnt sind.

Microservices Hochschulanwendung



- Datenbank, UI und Anwendungslogik von einander getrennt - *No Stack Anwendung*
- Bei einem Update einer Teilkomponente muss nur die Teilkomponente aktualisiert werden
- Einzelne Komponenten können gut skaliert werden
- Weiterentwicklungen können gut getestet werden, da die Komponenten voneinander gekoppelt sind

Wie groß ist ein Microservice?



Unterschiedliche Faktoren spielen eine Rolle:

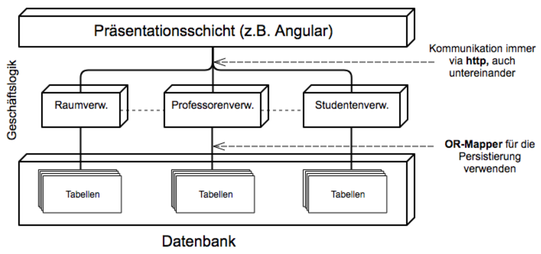
- Teamgröße
- Unternehmensstruktur
- Abhängigkeiten der Komponenten untereinander

Gefahr der Entstehung von "Nanoservices"

Erst einen Monolithen entwickeln und anschließend in Microservices aufteilen
Einschätzung der Komplexität ermöglichen

(Martin Fowler)

Microservice im Detail



Die Umsetzung von Microservices erfolgt i.d.R. mit einem Framework
Keine Vorgaben bezüglich der eingesetzten Technologien
REST hat sich als Architekturstil bewährt

- Nutzt Standard HTTP-Methoden
- JSON als schlankes Austauschformat
- Schnittstellen sind Interoperabel

REST Web Services

- Mit dem **RE**presentational **S**tate **T**ransfer (REST) Architekturstil lassen sich Web-Schnittstellen realisieren
- Verwendet Standard HTTP-Methoden um CRUD-Operationen auf Ressourcen durchzuführen
- Die vier HTTP-Verben beschreiben die durchzuführende Aktion

HTTP-Verb	Aktion
GET	SELECT
POST	INSERT
PUT	UPDATE
DELETE	DELETE

Welches Framework für Microservices?

spring boot Spark VERT.X Restlet

Welches Framework nun eingesetzt werden soll, ist stark abhängig davon welche Sprache man einsetzen möchte und in welcher die Entwickler am meisten Erfahrung haben. Ist die eingesetzte Sprache **Java** und persönliche Erfahrungen haben die Auswahl noch nicht weiter eingegrenzt, sollte **Spring-Boot** eingesetzt werden:

- Weit verbreitetes Framework im Bereich der Java-Entwicklung
- Embedded-Tomcat für jeden Microservice,
  - dadurch Einfacher Start des erstellten Artefakts ohne zusätzlichen Application-Server
- Schneller Einstieg möglich
- Gute Integration in Eclipse/IntelliJ
- Ausführliche Dokumentation und Tutorials zur weiteren Vertiefung im Anschluss des Workshops

Annotationen in Spring-Boot

Spring-Boot verwendet Annotationen um eine Microservice-Applikation zu realisieren. Nachfolgend ein Auszug einiger häufig benutzen Annotationen:

- **@RestController**: Kurzschreibweise um @Controller und @ResponseBody zur Klasse hinzuzufügen
- **@Controller**: Definiert die Klasse als Controller und sorgt dafür, dass Sie von Spring als solcher erkannt wird
- **@ResponseBody**: Rückgabewerte von Methoden werden an die Web-Antwort angehängt
- **@RequestMapping**: Weist Anfragen an den Controller (URI), den entsprechenden Methoden zu
- **@Autowired**: Teilt Spring mit, dass wir ein Objekt des Typs brauchen, worüber die Annotation steht. — @Inject für nicht Spring-spezifisch
- **@PostMapping / @GetMapping** Weist HTTP-POST / -GET Anfragen der entsprechenden Methode zu
- **@RequestBody**: Transformiert Informationen aus dem Body der HTTP-Anfrage in das nachstehende Objekt
- **@SpringBootApplication**: Notwendig damit Spring-Boot Controller und andere Komponenten finden kann. Identisch zu **@Configuration**, **@EnableAuto Configuration** und **@ComponentScan**

Max Jando

Joshua Vécsei