Moduldokumentation

Modul Workshop Distributed Software Systems (wodss)

Simon Wächter

2019

Inhalt

[1 Einleitung 2](#_Toc4433546)

[1.1 Einleitung 2](#_Toc4433547)

[1.2 Lernziele 2](#_Toc4433548)

[1.3 Prüfungen 2](#_Toc4433549)

[2 Woche 1 3](#_Toc4433550)

[2.1 Übersicht Workshop Verteilte Systeme, Frühjahrssemester 2019 3](#_Toc4433551)

[2.1.1 Dozenten 3](#_Toc4433552)

[2.1.2 Ziele 3](#_Toc4433553)

[2.1.3 Vorgehen 3](#_Toc4433554)

[2.1.4 Funktionale Anforderungen 4](#_Toc4433555)

[2.1.5 Nicht-funktionale Anforderungen 4](#_Toc4433556)

[2.2 Zeitplan 5](#_Toc4433557)

[3 Woche 2 6](#_Toc4433558)

[4 Woche 3 7](#_Toc4433559)

[5 Woche 4 8](#_Toc4433560)

[6 Woche 5 9](#_Toc4433561)

[7 Woche 6 10](#_Toc4433562)

[8 Woche 7 11](#_Toc4433563)

# Einleitung

## Einleitung

Dieses Dokument stellt die Moduldokumentation für das Modul wodss dar. Allfällige Unterlagen sind im Modulordner zu finden.

## Lernziele

Die Studierenden arbeiten in Gruppen an einer gegebenen Aufgabenstellung. Ziel ist es, ein relativ einfaches Geschäftsmodell mit Hilfe von Internettechnologien komponentenbasiert abzubilden. Technologien wie Servlets, JSP & JSF, Spring MVC, Ajax, Enterprise Java Beans (SessionBeans) oder Spring, O/R Mapping (JPA oder Hibernate), Authentifizierung & Autorisierung und WebServices werden eingesetzt.

## Prüfungen

Die Arbeit wird benotet und ergibt die Modulnote. Die Bewertung erfolgt aufgrund:

* des Projektberichtes
* des Codes
* der Präsentation
* einer mündlichen Prüfung

# Woche 1

## Übersicht Workshop Verteilte Systeme, Frühjahrssemester 2019

### Dozenten

* Dominik Gruntz
* Dierk König
* Jürg Luthiger
* Arno Wagner

### Ziele

Ziel des Workshops verteilte Systeme ist die Anwendung des Wissens aus den Modulen zu den Themen verteilter Programmierung, Web Technologien, Security und Architektur in Unternehmensanwendungen in einem übergreifenden Projekt. Auf diese Art soll das Wissen vertieft und durch praktische Erfahrungen verankert werden.

Bei der Umsetzung eines verteilten Systems ist es wichtig, dass der Entwickler auch folgende funktionsübergreifende Fragen beantworten kann:

* Wie gestaltet man das Remote API und welche Protokolle kommen zum Einsatz?
* Welche Technologien wählt man für Client und Server?
* Wie aufwändig sind Einarbeitung, Implementierung und Test?
* Wie sicher ist das erstellte System?
* Wie gut ist das System mittel- und langfristig erweiterbar, wartbar und betreibbar?
* Wie gestaltet man eine angenehme Benutzeroberfläche?
* Wie baut man eine effiziente Datenablage?
* Welche Arten von Nutzern gibt es und wer darf das System wie verwenden?

### Vorgehen

Für die Beantwortung der obigen Fragen teilen wir den Workshop in 6 Projektgruppen mit je 4 Studierenden. Jede Gruppe erstellt eine voll funktionsfähige Lösung mit Client und Server.

Die Lösung soll so gebaut werden, dass der Client von jeder Gruppe mit dem Server jeder Gruppe kombiniert werden kann.

* Phase 1: Technologieauswahl: Jede Gruppe wählt einen Technologie-Stack sowohl für Cient wie auch für den Server. Ziel des Workshops ist es, eine möglichst grosse Vielfalt an Technologien zu erreichen.
* Phase 2: Schnittstellen: Die Studierenden analysieren die funktionalen Anforderungen und einigen sich auf ein gemeinsames remote (REST) API. Dieses API gilt für alle Gruppen.
* Phase 3: Implementierung: Jede Gruppe implementiert ihre Lösung.
* Phase 4: Präsentation (bewertet): Jede Gruppe stellt ihre Lösung vor
* Phase 5: Kombination (bewertet): Jede Gruppe kombiniert den Server Teil ihrer Lösung mit dem Client Teil einer anderen Gruppe.
* Phase 6: Diskussion (bewertet): Die Studierenden können einen Kriterienkatalog für die gegenseitige Bewertung der Lösung vorschlagen, die Lehrpersonen geben den endgültigen Kriterienkatalog bekannt.
  + Jede Gruppe gibt eine Selbsteinschätzung bezüglich des Kriterienkatalogs ab.
  + Jede Gruppe für jede andere Gruppe eine Bewertung deren Lösung bezüglich des Kriterienkatalogs ab.
* Phase 7: Abschlussbesprechung: Die Dozierenden leiten die Abschlussdiskussion und geben die finalen Bewertungen bekannt.

### Funktionale Anforderungen

Das zu erstellende System soll die Einsatzplanung von Mitarbeitern eines Unternehmens erlauben. Dabei gilt:

* Es gibt nur ein Unternehmen. Eine Hierarchie des Unternehmens muss nicht modelliert werden.
* Das Unternehmen hat viele Mitarbeiter.
* Jeder Mitarbeiter hat einen Beschäftigungsgrad (max: 100%).
* Jeder Mitarbeiter kann Administrator, Projektleiter oder Entwickler sein. Diese Zuordnung ist einmalig und kann nachträglich nicht verändert werden.

Das Unternehmen bearbeitet Projekte. Für diese Projekte gilt:

* Ein Projektleiter kann innerhalb seines Beschäftigungsgrades mehrere Projekte leiten.
* Jedes Projekt hat genau einen Projektleiter und viele zugeordnete Entwickler. Projekte haben einen gegebenen Bedarf an Entwicklern, z.B. 5 FTE (full-time-equivalents) also 5 Entwickler zu je 100% oder 10 Entwickler zu 50%.
* Projekte haben ein Anfangs- und ein End-Datum. Vor dem Start und nach dem Ende sind die Projekte für die Entwickler inaktiv.
* Jeder Entwickler kann mehreren Projekten zu einem bestimmten Prozentsatz zugeordnet sein. Die Summe aller seiner Zuordnungen darf seinen Beschäftigungsgrad nicht überschreiten.

Rolle "Administrator":

* Administratoren sind vorkonfiguriert.
* Administratoren dürfen Projekte, Projektleiter und Entwickler sehen, anlegen, löschen/deaktivieren und verändern.
* Administratoren bekommen eine Übersicht über alle Projekte, deren Bedarf an Entwicklern und die Summe der zugeordneten Entwickler.
* Administratoren bekommen eine Übersicht über alle Entwickler, deren Beschäftigungsgrad und deren Summe an Projekt-Zuordnungen.

Rolle "Projektleiter":

* Projektleiter dürfen alle Projekte mit allen ihren Eigenschaften sehen.
* Projektleiter dürfen ihr eigenes Projekt verändern, insbesondere die Zuordnung von Entwicklern zu ihrem Projekt.
* Projektleiter bekommen eine Übersicht über alle Entwickler, deren Beschäftigungsgrad und deren Summe an Projekt-Zuordnungen.

Rolle "Entwickler":

* Entwickler dürfen ihre eigenen Eigenschaften und die Projekte, zu denen sie zugeordnet sind, sehen.

### Nicht-funktionale Anforderungen

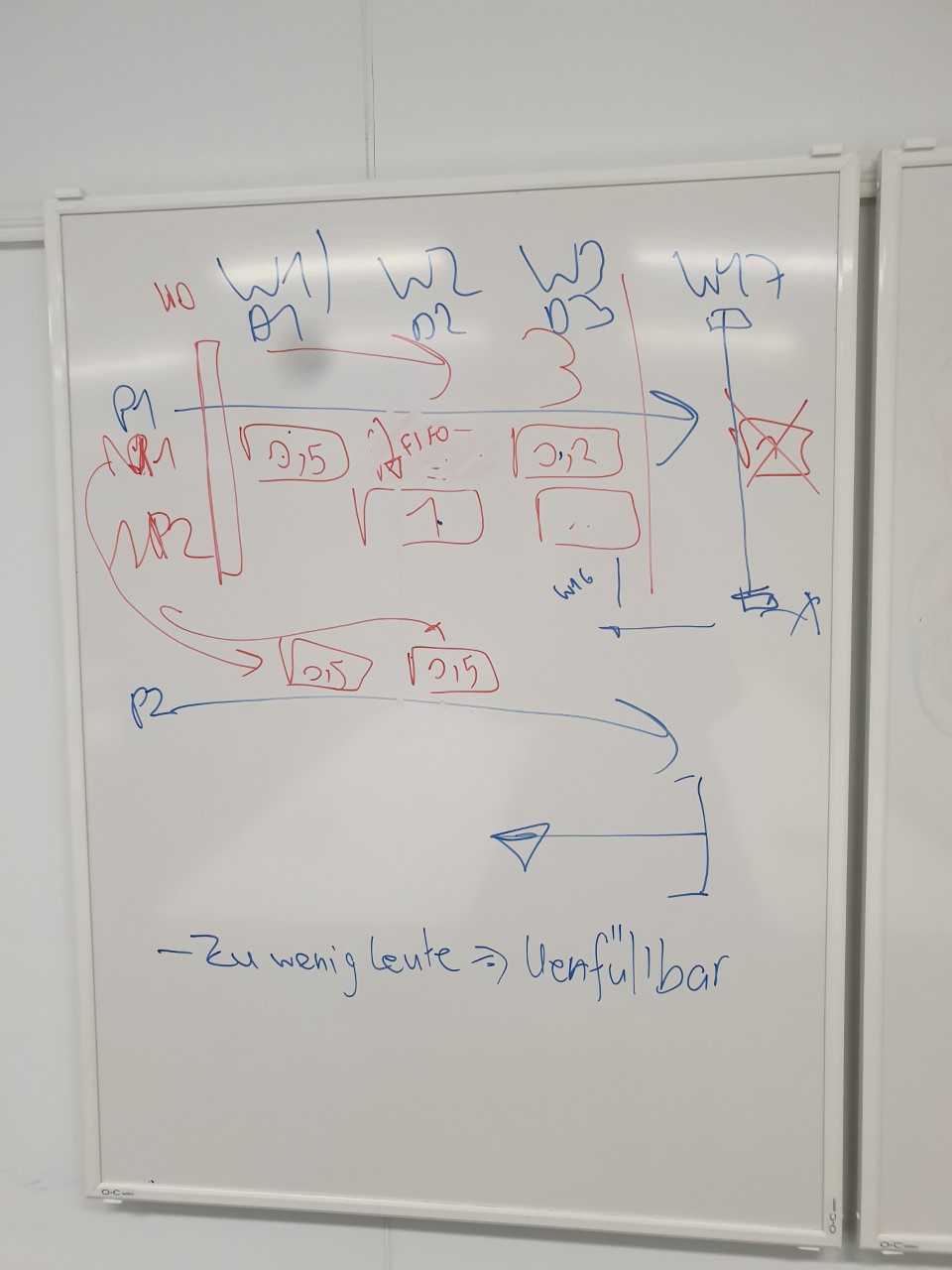
* Für den Server existiert eine Testsuite mit einigen, repräsentativen Tests.
* Die Inbetriebnahme und der Betrieb von Client und Server sind jeweils klar beschrieben.
* Der Server stellt eine Demoversion bereit, die automatisch mit Demodaten initialisiert wird.
* Die Authentifizierung nutzt z.B. JWT.

## Zeitplan



# Woche 2

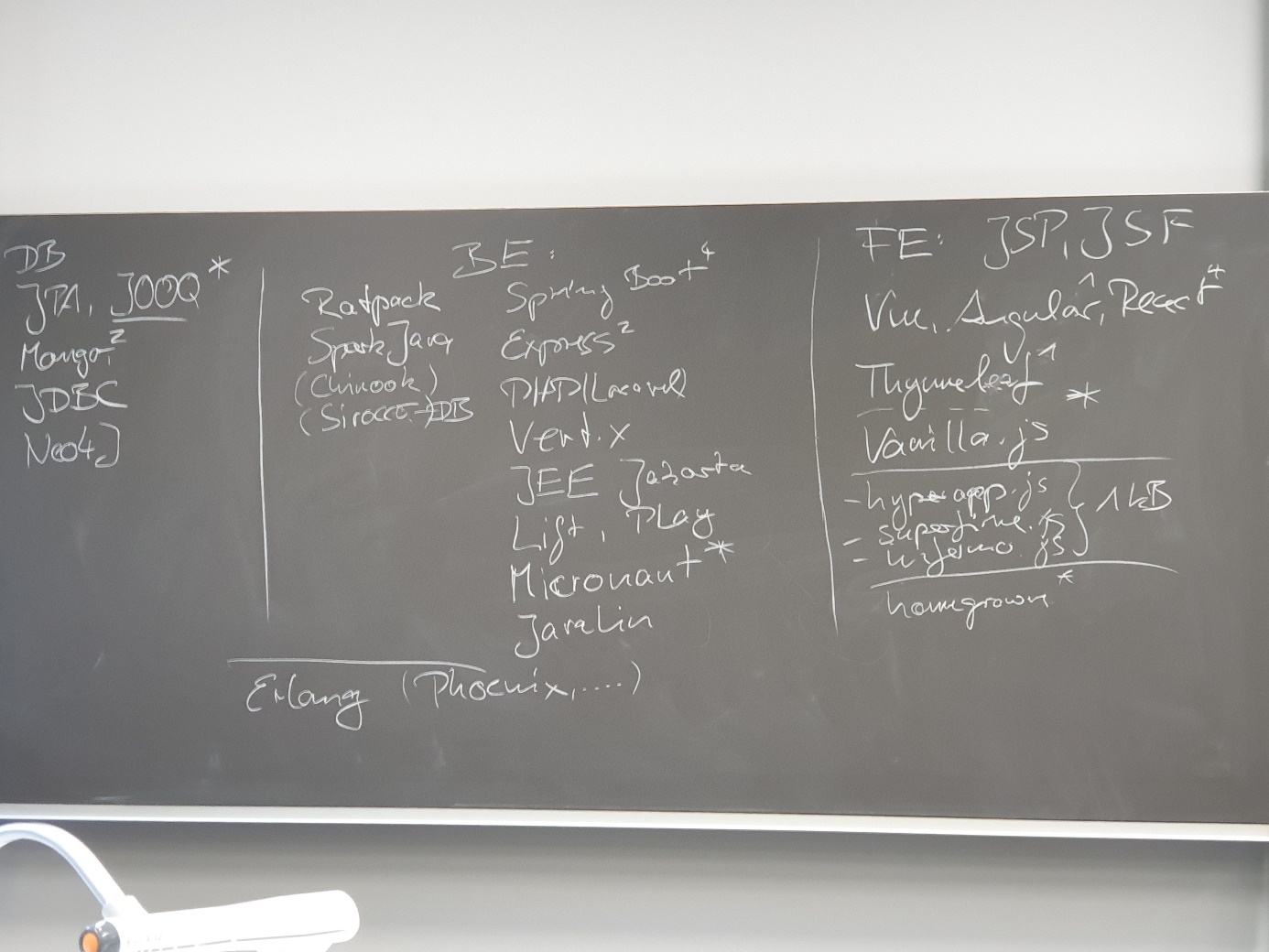
In Woche 2 wurde zusammen mit Herrn Gruntz ein möglicher Aufbau besprochen:



# Woche 3

In Woche wurde die API von David weiter verfeinert.

# Woche 4



# Woche 5

In Woche 5 wurde die API finalisiert und an die Dozenten geschickt.

# Woche 6

In Woche 6 wurde der weitere Projektablauf bis zum 29. April geplant.

# Woche 7