New De[h]li - Course Management System

Projektteam:

- Tobias Arndt
- Marvin Gorecki
- Alexander Kastius
- Tobias Knöschke
- Melanie Schneider

Betreuer:

- Daniel Kurzynski

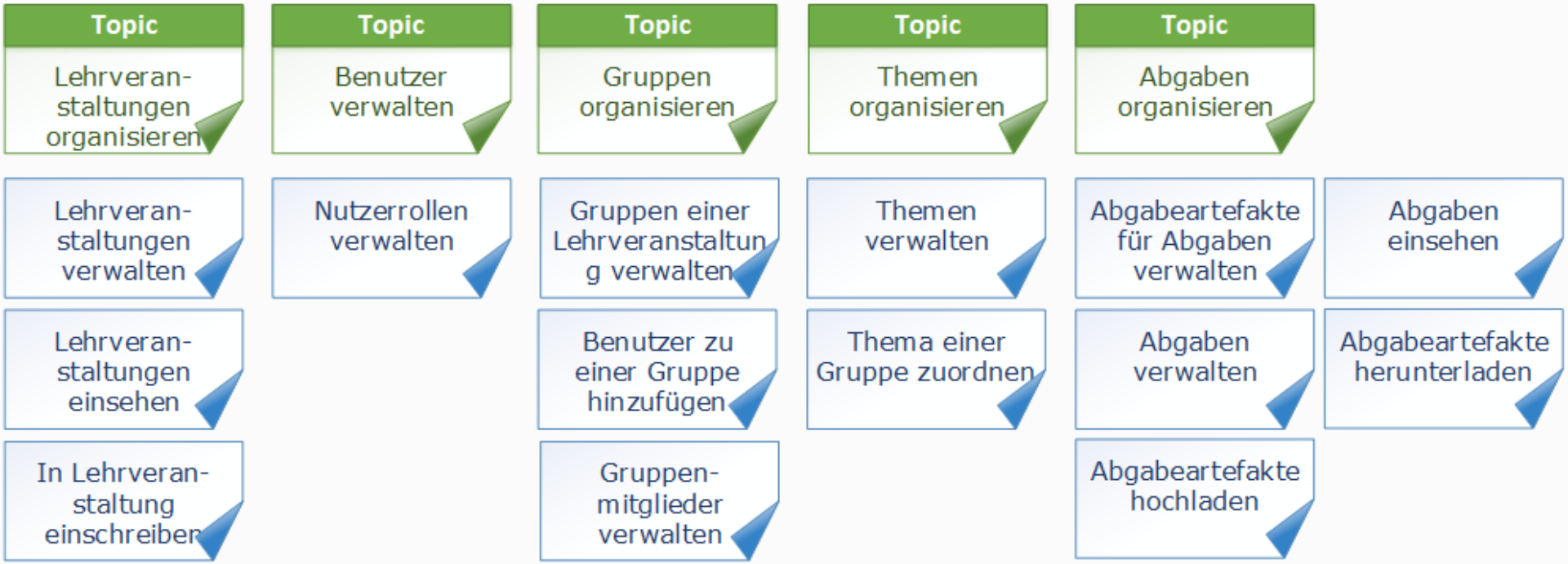
Softwaretechnik I - Gruppe 15

23.06.2015

Motivation

- Bedienung von Deli nur teilweise intuitiv
- fehlendes Feedback nach Nutzeraktionen
  z.B. beim Einschreiben eines Themas oder Gruppe
- Timeouts beim Upload von Dateien > 50 MB

Funktionale Anforderungen



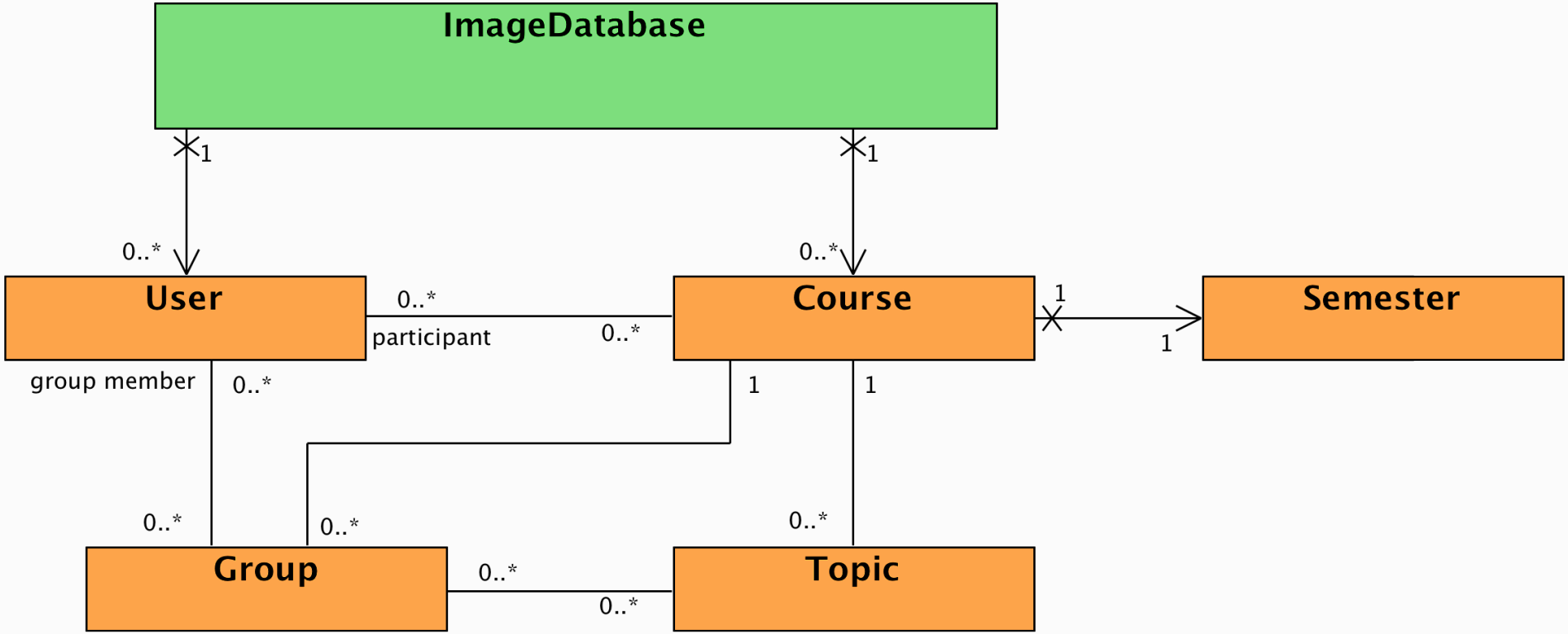
Nicht-funktionale Anforderungen

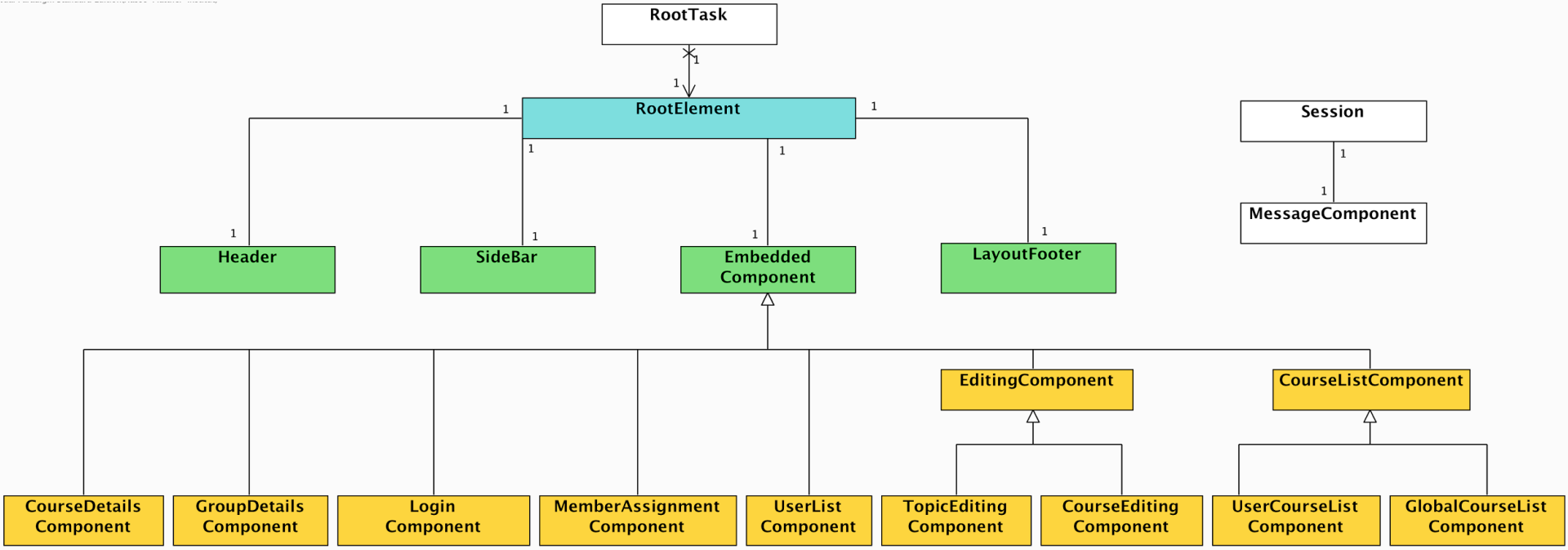
- einfache und benutzerfreundliche Bedienbarkeit

  nach 3-Klick-Regel
- Kommunikation mit dem Server soll gering gehalten werden
- Nutzer sollen nach einer Aktion ein Feedback erhalten
- Datei-Uploads via nginx-Webserver

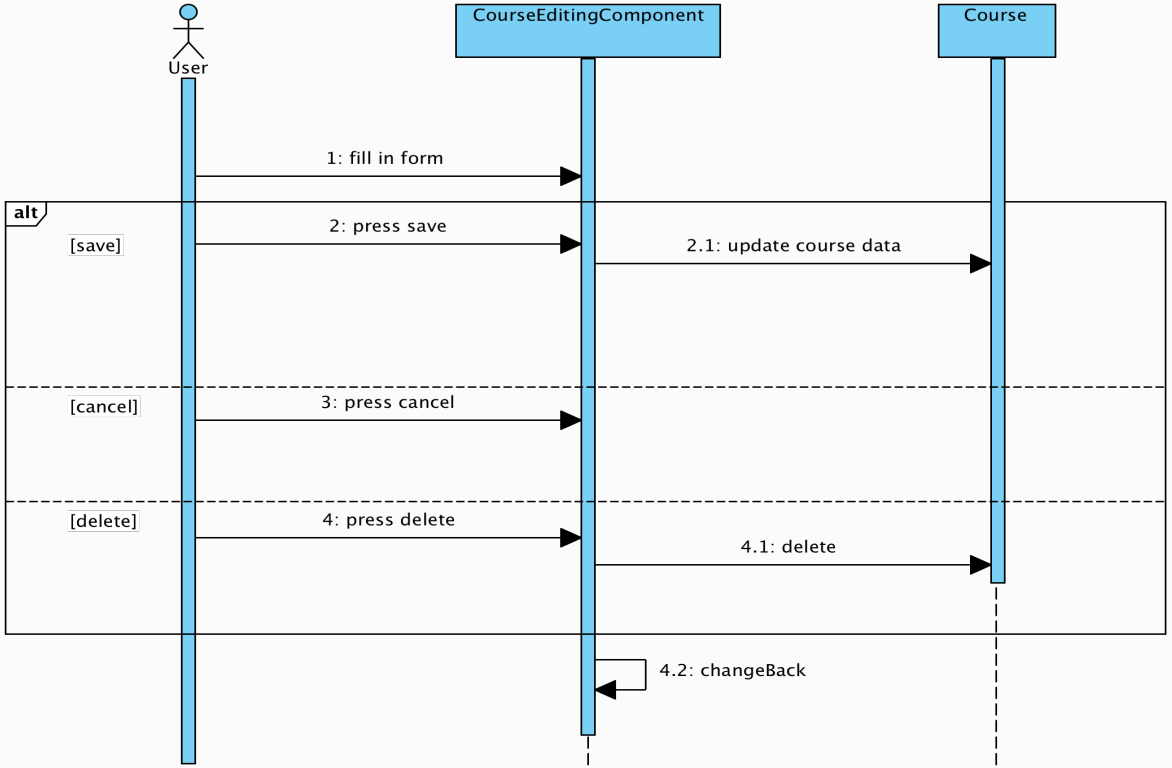
Demo

new De[h]li


ImageDatabase
1
1
0..*
0..*
User
0..*
participant
0..*
Course
1
1
Semester
group member
0..*
1
1
0..*
0..*
0..*
Group
0..*
0..*
Topic

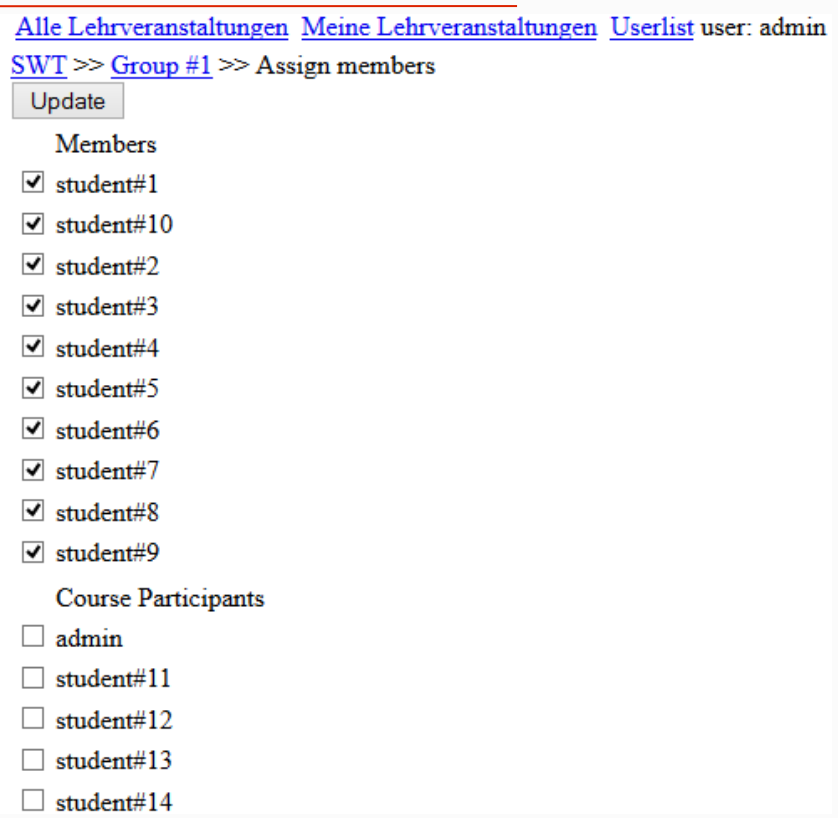

RootTask
RootElement
Session
MessageComponent
Header
SideBar
Embedded Component
LayoutFooter
EditingComponent
CourseListComponent
CourseDetails Component
GroupDetails Component
Login Component
MemberAssignment Component
UserList Component
TopicEditing Component
CourseEditing Component
UserCourseList Component
GlobalCourseList Component

User Story - Lehrveranstaltungen verwalten



User Story - Gruppenmitglieder verwalten

vs.



Komponenten laden

```
assignMembers

    self rootComponent
      setMainContentTo: CMSMemberAssignmentComponent
      preparedBy: [ :component | component group: self group]
```

```
setMainContentTo: aCMSEmbeddedComponentClass preparedBy: aBlock

    | loadedComponent |
    loadedComponent := self loadComponent: aCMSEmbeddedComponentClass.
    aBlock value: loadedComponent.
    self mainContent: loadedComponent
```

```
loadComponent: aComponentClass

    ^ self components at: aComponentClass
      ifAbsent: [self createComponent: aComponentClass].
```

UI Design & Usability

UI Design:

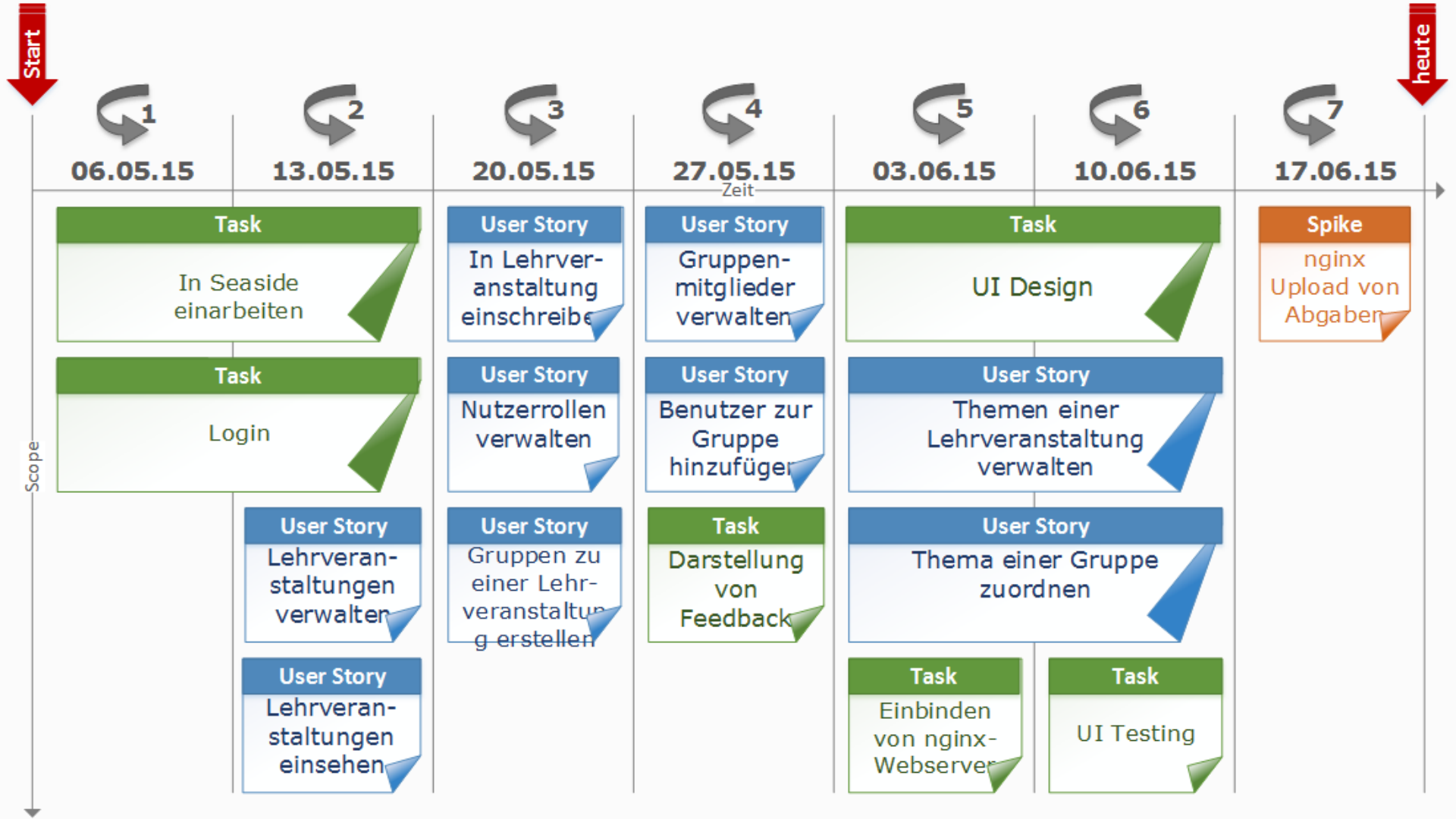
- Verwendung des CSS-Frameworks Bootstrap

Usability:

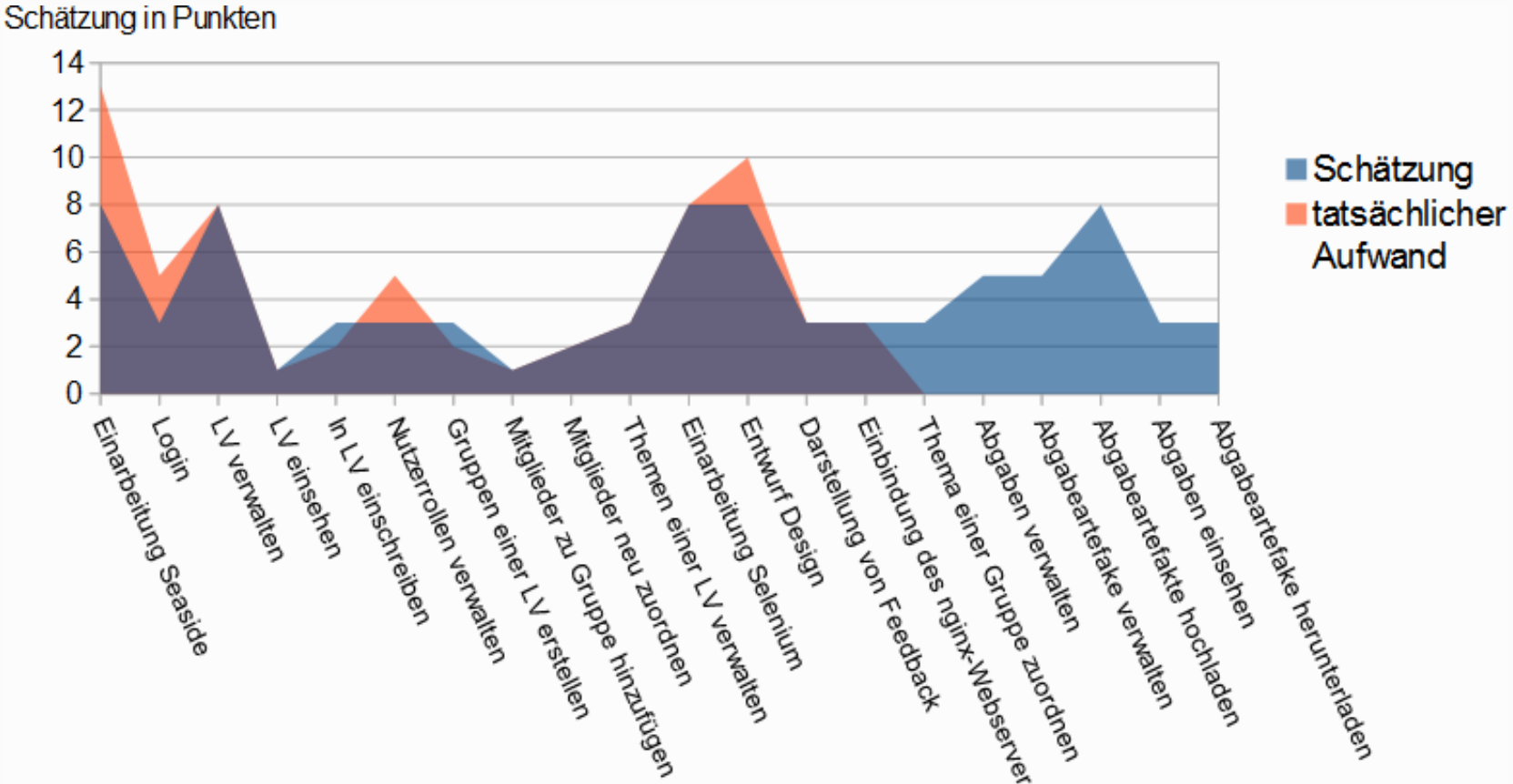
- 3-Klick-Regel
- einfaches Bedienung durch minimales Design
- Feedback

Projektverlauf

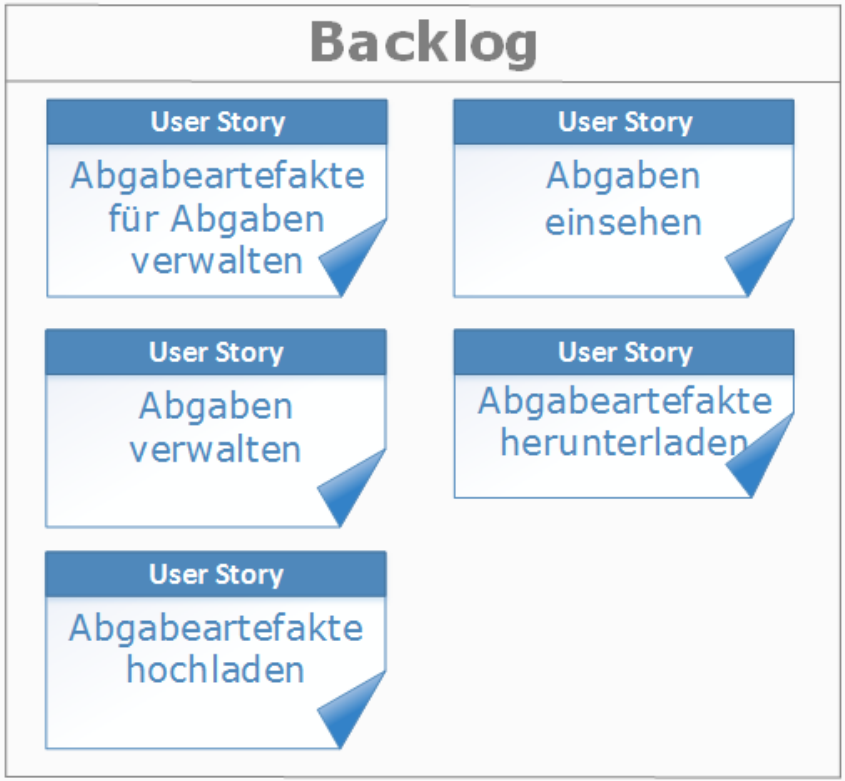
- Projektstart am 06. Mai
- wöchentliche Iterationen
- Workload von 8 bis 10 Punkten pro Iteration
- 2 Teammeetings pro Woche
- Projektorganisation mit Hilfe von Redmine
- Kommunikation außerhalb der Teammeetings via Slack
- Umsetzung der User Stories nach Priorisierung


Start
heute
1
2
3
4
5
6
7
06.05.15
13.05.15
20.05.15
27.05.15
03.06.15
10.06.15
17.06.15
Zeit
Scope
Task
In Seaside einarbeiten
Task
Login
User Story
Lehrveran-staltungen verwalten
User Story
Lehrveran-staltungen einsehen
User Story
In Lehrver-anstaltung einschreibe
User Story
Nutzerrollen verwalten
User Story
Gruppen zu einer Lehr-veranstaltu g erstellen
User Story
Gruppen-mitglieder verwalten
User Story
Benutzer zur Gruppe hinzufüge
Task
Darstellung von Feedback
Task
UI Design
User Story
Themen einer Lehrveranstaltung verwalten
User Story
Thema einer Gruppe zuordnen
Task
Einbinden von nginx-Webserve
Task
UI Testing
Spike
nginx Upload von Abgaben

Aufwandsschätzung



Planung bis Projektende



Extreme Programming - im Projekt

- schnelle Entwicklung
- Code früh lauffähig machen und darauf aufbauen
- kurze Iterationen, kleine Schritte
- eingesetzte Praktiken: TDD und Pair Programming
- 2 Teammeetings pro Woche
- Projektorganisation mit Hilfe von Redmine
- Kommunikation außerhalb der Teammeetings via Slack

Test Driven Development

- Erst die Tests,
  dann der Code!
- Tests sollten erwartete Funktionalität
  exakt und vollständig
  beschreiben (Randfälle!)
- Code soll Tests erfüllen

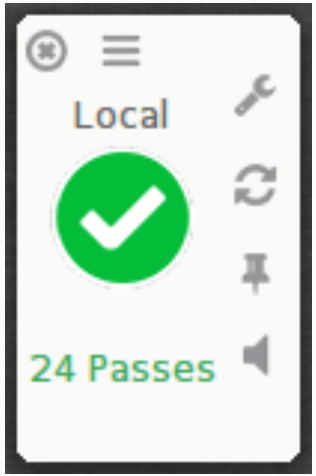
testAutoLeavingGroup

```
| course group member |
course := CMSCourse new.
group := course createNewGroup.
member := CMSUser new.
member addCourse: course.
group addNewMember: member.

self assert: (member participatesIn: group).
self assert: (group isMember: member).

member leaveCourse: course.

self deny: (member participatesIn: group).
self deny: (group isMember: member)
```

Test Driven Development

TDD in Smalltalk: Test Auto Runner

- Wähle zu überwachenden Code & zugehörige Tests
- Bei Änderung werden Test erneut ausgeführt



Test Driven Development - im Projekt

Unsere Erfahrung:

- Anfangs ungewohnt (typisch für neue Methodik), später Routine
- Bestätigung (erfolgreiche Tests) wirkt positiv

Pair Programming

Pro:

- zu zweit vor einem Rechner arbeiten
- können voneinander lernen
- verbesserte Fehlererkennung und Code-Optimierung

Contra:

- zwei erledigen, was einer erledigen könnte

Pair Programming - im Projekt

- 2 Teammitglieder pro zu entwickelnde Funktionalität
- schnellere Einarbeitung durch direkten Austausch
- Entscheidungsfindung dauert länger

Ausblick

- deploybare Anwendung
- automatisierte UI-Tests mit Selenium
- mögliche Erweiterbarkeit: Mehrsprachigkeit durch Spracheinstellung im Browser

Unser Fazit

- viel Kommunikation mit dem Kunden und innerhalb des Teams durch wöchentliche Iterationen
- schnelle Entwicklung von Funktionen, die wirklich gebraucht werden
- Aber! das Team wird verleitet Funktionen sehr schnell fertig zustellen. Dadurch leidet auch mal die Qualität.

Vielen Dank
für die Aufmerksamkeit

Fragen?