

Praktikum: XML Technologie

Gruppe DocBook

Patryk Brzoza, Moritz Issig, Fami Mahmud

Technische Universität München

Fakultät für Informatik

Lehrstuhl für Angewandte Informatik /

Kooperative Systeme

Garching b. München, 09. März 2020



Agenda

1. Einführung
2. Technologien
3. Projektanforderungen
4. Design und Implementierung
5. Architektur
6. Spiel-Demo
7. Teamorganisation
8. Fazit

Einführung - Blackjack

Regeln:

- **Ziel:** Kartenhand möglichst nah an 21 aber nicht mehr
- Jede*r Spieler*in spielt für sich gegen den Dealer
- Das Kartendeck besteht aus 6x 52 Karten
- Bildkarten zählen 10. Zahlenkarten (2-10) den Zahlenwert und Asse 11 oder 1

Spielablauf:

1. **Chips setzen:** Spieler*innen setzen Chips für die kommende Runde
2. **Karten erhalten:** der Dealer und jede*r Spieler*in erhält 2 Karten. Nur eine Dealerkarte offen
3. **Spielerzug:** nacheinander haben die Spieler*innen nun die Möglichkeit zu:
 - **hit:** weitere Karte erhalten (mehrfach möglich)
 - **stand:** Zug beenden
4. **Dealerzug:** Dealer zieht bei 16 und beendet Zug ab 17
5. **Ergebnisermittlung:** Vergleich von Dealerhand und Spielerhand → Auszahlung

Technologien - XML

XML steht für *Extensible Markup Language*

→ definiert einen Regelsatz für die
plattformunabhängige Speicherung von
Dokumenten

```
<player id="1522" name="Nelson">
  <hand sum="20">
    <card hidden="false">
      <type>Diamond</type>
      <value>10</value>
    </card>
    <card hidden="false">
      <type>Diamond</type>
      <value>K</value>
    </card>
  </hand>
  <wallet>300</wallet>
  <pool locked="true">
    <chip>
      <value>100</value>
    </chip>
  </pool>
</player>
```

Technologien - DocBook

DocBook ist eine *Semantic Markup Language*

- vereinfacht die Erstellung von Dokumenten
- Transformierung zu HTML, PDF, ... möglich

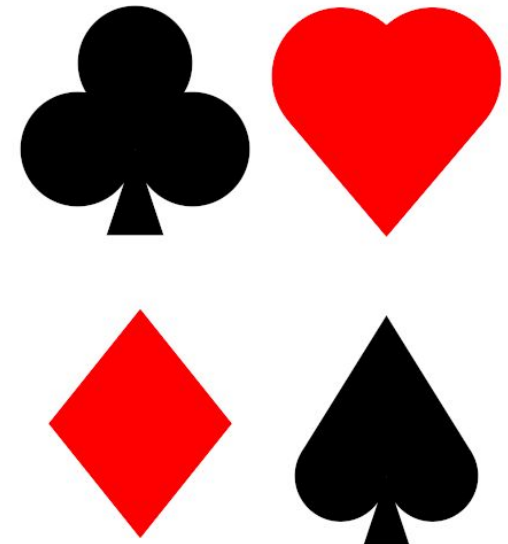
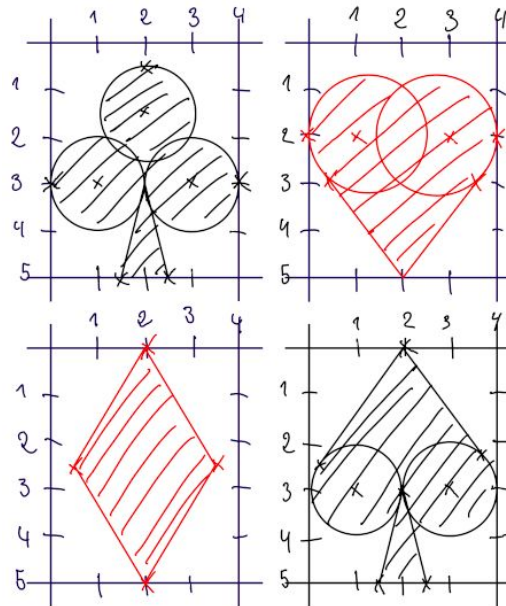
```
<chapter id="architecture">
  <title>Architecture</title>
  <para>This chapter describes the connections between the different hardware-components which
    are basing on a Model-View-Controller architecture pattern. The second section presents
    the data-models of the lobby, containing the scores and the games, and the card-values,
    which will be used to generate the card-svgs.</para>
  <sect1>
    <title>MVC Architecture</title>
    <para>The program is separated in the separate sub-components: the Model, View and Control
      <listitem>
```

Technologien - SVG

SVG steht für *Scalable Vector Graphics*

→ ermöglicht skalierbaren Einsatz von Grafiken unabhängig von Größe

- Grundbaustein aller GUI-Elemente
- Übertragen von Skizzen in Grafiken mit ViewBox und SVG-Elementen



Technologien - XSLT

XSLT steht für *Extensible Stylsheet Language Transformations*

XSLT ermöglicht Transformieren von XML-Dokumenten
in andere XML-Dokumente (wie HTML oder SVG)

```
<card hidden="true">  
  <type>Spade</type>  
  <value>10</value>  
</card>
```



Technologien - XQuery

XQuery ermöglicht die Interaktion mit XML-Datenbanken.

- replace / insert / delete node

Knoten können mit XPath adressiert werden.

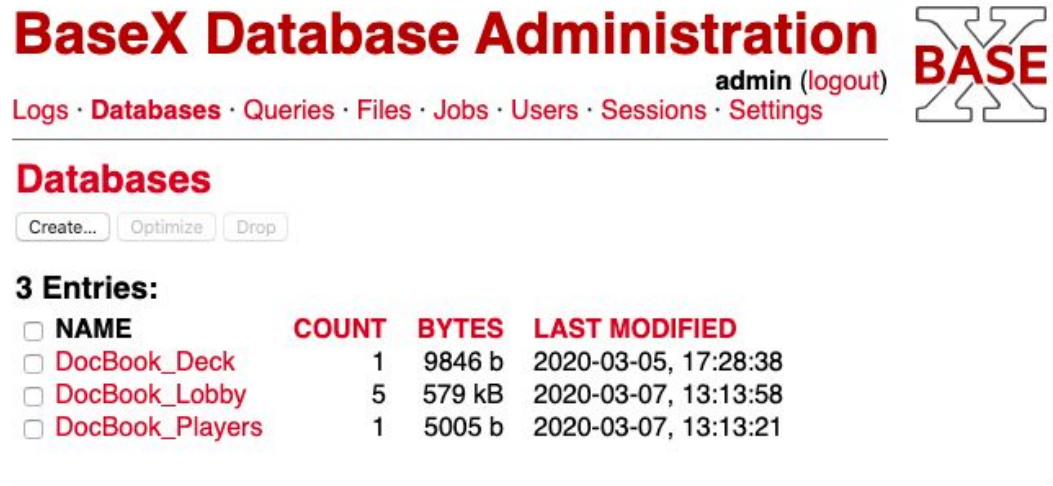
```

declare
%rest:path("docbook_blackjack/setup")
%output:method("xhtml")
%updating
%rest:GET
function blackjack-controller:setup() {
    let $lobby_model := doc(concat($blackjack-controller:staticPath, "db/Lobby.xml"))
    let $deck_model := doc(concat($blackjack-controller:staticPath, "db/Deck.xml"))
    let $players_model := doc(concat($blackjack-controller:staticPath, "db/Players.xml"))
    let $redirectLink := "/docbook_blackjack"
    return (
        if (not(db:exists("DocBook_Lobby")))
        then (db:create("DocBook_Lobby", $lobby_model)),
        if (not(db:exists("DocBook_Deck")))
        then (db:create("DocBook_Deck", $deck_model)),
        if (not(db:exists("DocBook_Players")))
        then (db:create("DocBook_Players", $players_model)),
        update:output(web:redirect($redirectLink))
    );
};

```


Technologien - BaseX

- XML-basiertes Datenbankmanagementsystem
- Unterstützung verschiedener APIs:
 - HTTP Server als Schnittstelle für Clients
 - RESTXQ für REST-Methoden
 - STOMP für Websocket-Kommunikation



BaseX Database Administration admin (logout)

Logs · **Databases** · Queries · Files · Jobs · Users · Sessions · Settings

Databases

Create... Optimize Drop

3 Entries:

<input type="checkbox"/> NAME	COUNT	BYTES	LAST MODIFIED
<input type="checkbox"/> DocBook_Deck	1	9846 b	2020-03-05, 17:28:38
<input type="checkbox"/> DocBook_Lobby	5	579 kB	2020-03-07, 13:13:58
<input type="checkbox"/> DocBook_Players	1	5005 b	2020-03-07, 13:13:21

BaseX Team, 2014-18

Projektanforderungen

Offizielle Anforderungen:

- Spielkonzept nach Blackjack-Regeln aus der Einführung
- Webanwendung
- Ausschließlich XML-Technologien
- Dynamische Erzeugung von Karten und Chips
- MultiClient: mehrere Spieler*innen mit eigenen Clients in einem gemeinsamen Spiel
- Lobby-Funktion mit Anmeldung für die Spieler*innen
- Highscore-Board in der Lobby
- Architektur nach Model-View-Controller (MVC)-Muster
- Nutzung von UML-Diagrammen zur Beschreibung von Architektur, Daten und Verhalten
- Beschreibung aller REST-Schnittstellen
- Dokumentierung des Projekts

Projektanforderungen

Zusatzfunktionen:

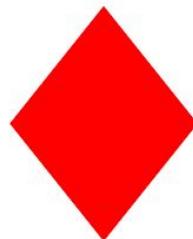
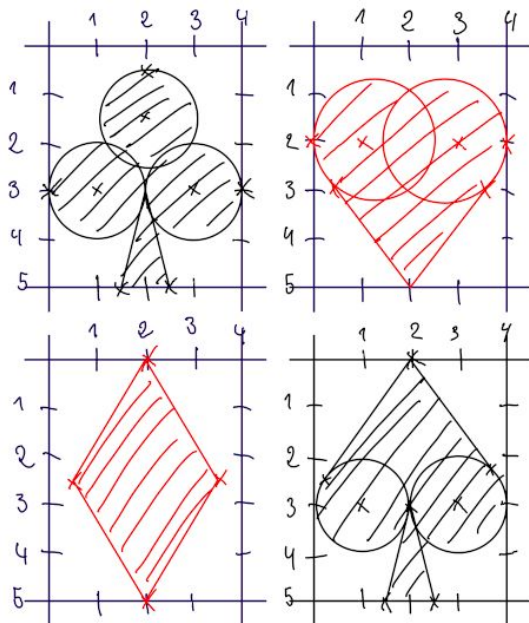
- Mehrere MultiClient-Spiele parallel spielbar
- SinglePlayer-Spiele möglich
- Bereitstellung auf einem Webserver

Design und Implementierung - Card Template

Card Template nimmt die Parameter Wert und Typ an

→ Zieht aus Values.xml die Punkte für Verteilung der Icons

→ Lädt SVG des Icons



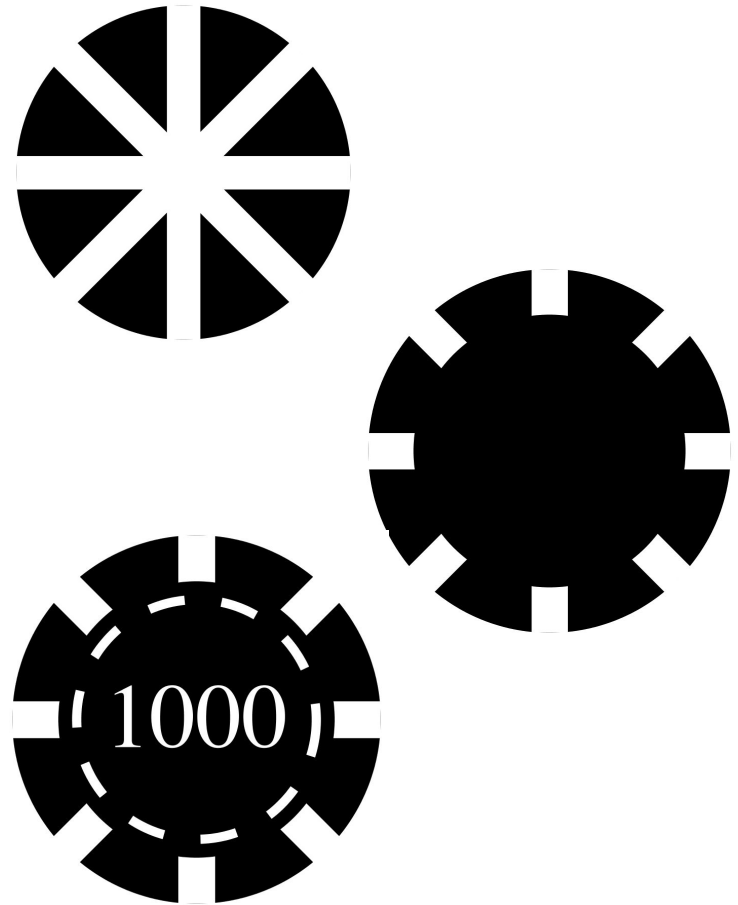
Value					
Type	0	1	2	3	4
	5	6	7	8	9
	10	11	12	13	14
	15	16	17	18	19
	20	21	22	23	24
	25	26	27	28	29
	30	31	32	33	34
					Type
					Value

Design und Implementierung - Chip Template

Template erhält Wert als Parameter
→ determiniert Wert des Chips und Farbe

Chips werden in 3 Schritten mit Hilfe des
<symbol> und <use> Elemente erstellt:

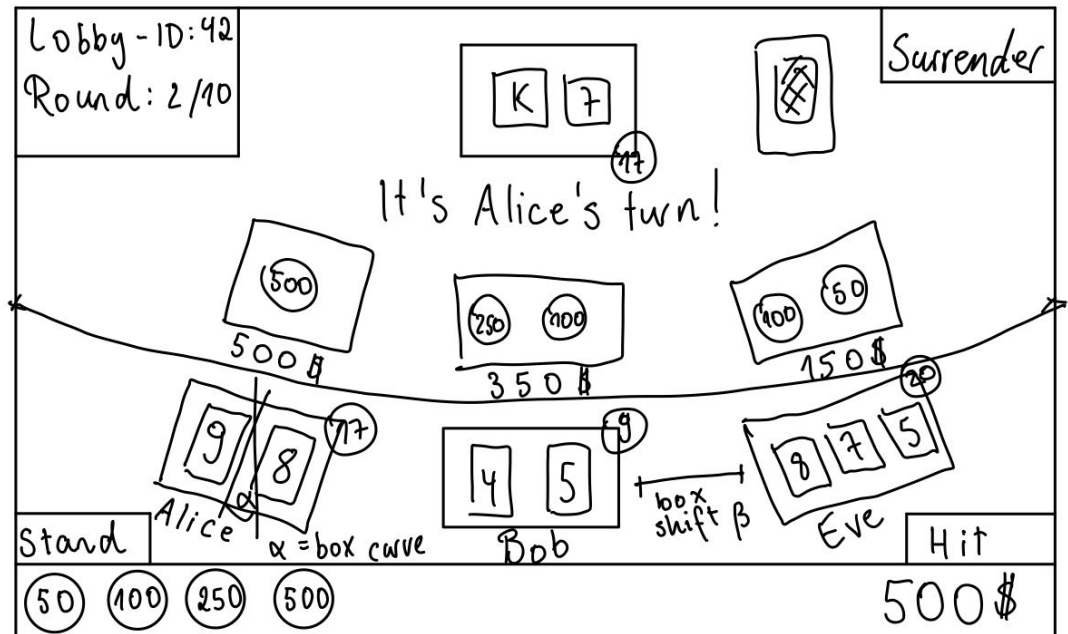
1. (schwarzer) Kreis mit weißen Linien
2. (schwarzer) innerer Kreis
3. gestrichelter weißer Kreis mit Wert



Design und Implementierung - Game Template

Game Template nimmt Parameter *playerName*, *playerId* und *gameld* an:

- Generierung von Chips und Karten mit XSL-Calls
- Spieleranzahl dynamisch skalierbar
- Karten- und Chipabstände dynamisch skalierbar



pC = player Count

x = box position

$$\alpha = \frac{10 \cdot pC - 2x + 1}{pC - 1} \quad (\text{curvature of player box})$$

$$\beta = \left(\frac{7}{pC} \right) \left(x - pC + \frac{pC}{2} \right)^2 \quad (\text{distance between player boxes})$$

Design und Implementierung - Game Template

Game Template nimmt Parameter *playerName*, *playerId* und *gameId* an:

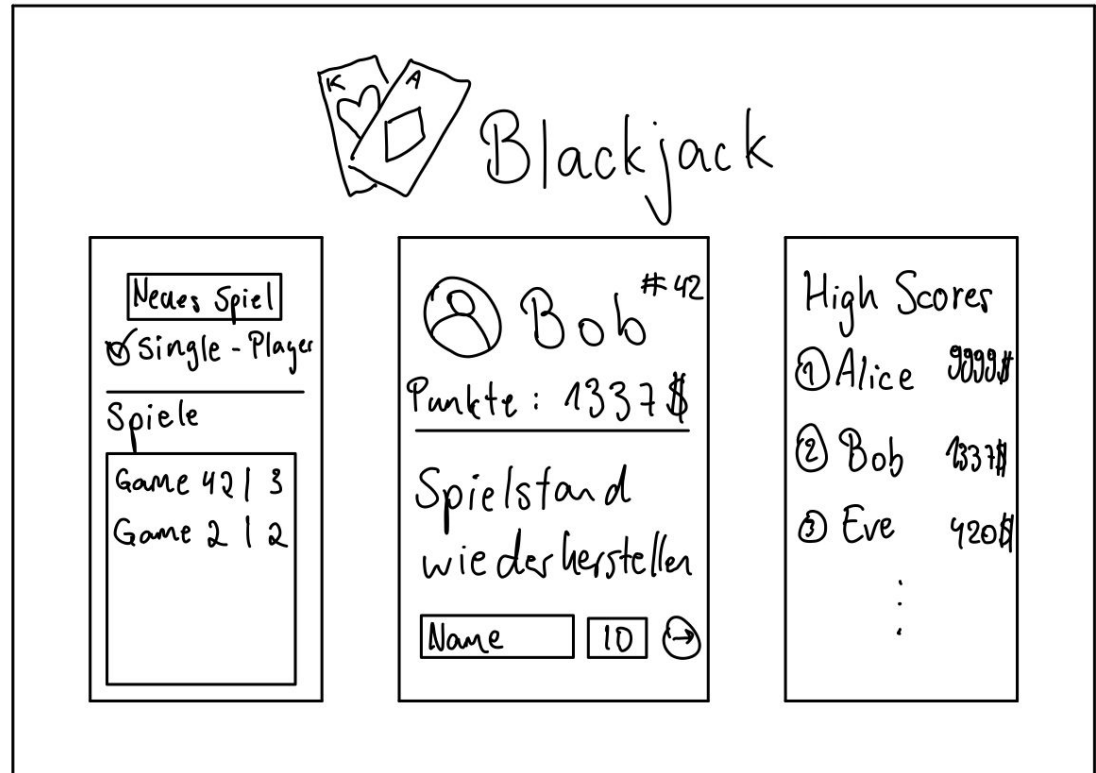
- Generierung von Chips und Karten mit XSL-Calls
- Spieleranzahl dynamisch skalierbar
- Karten- und Chipabstände dynamisch skalierbar



Design und Implementierung - Lobby Template

Lobby Template nimmt Parameter *isLoggedIn*, *playerName*, *playerId* und *playerHighscore* an:

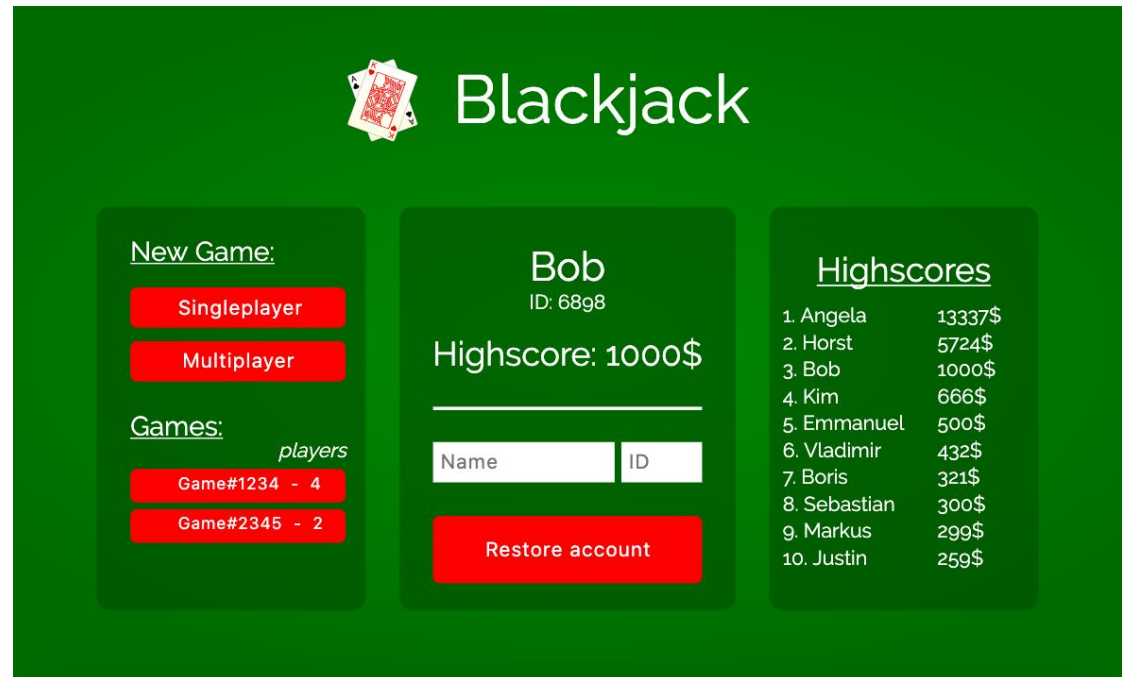
- Spielermanagement mit Name und ID
- Anzeige von verfügbaren Spielen
- Anzeige von persönlichen und globalen Highscores



Design und Implementierung - Lobby Template

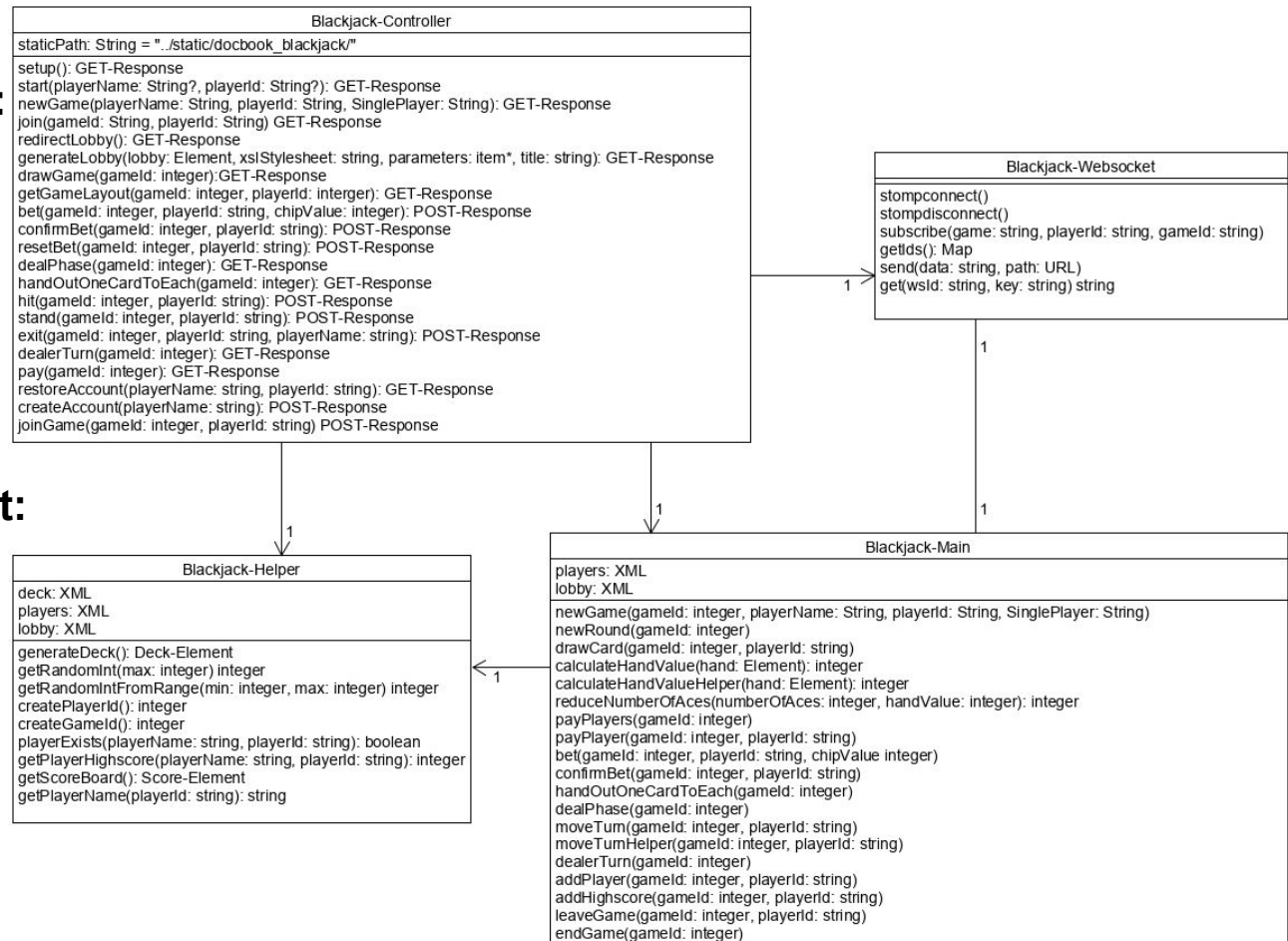
Lobby Template nimmt Parameter *isLoggedIn*, *playerName*, *playerId* und *playerHighscore* an:

- Spielermanagement mit Name und ID
- Anzeige von verfügbaren Spielen
- Anzeige von persönlichen und globalen Highscores



Implementierung - Module

- **Blackjack-Controller:**
REST-Schnittstelle
- **Blackjack-Main:**
Spiellogik,
Zugriff auf Datenbank
- **Blackjack-Helper:**
Hilfsfunktionen
- **Blackjack-Websocket:**
Verwalten der
Websockets



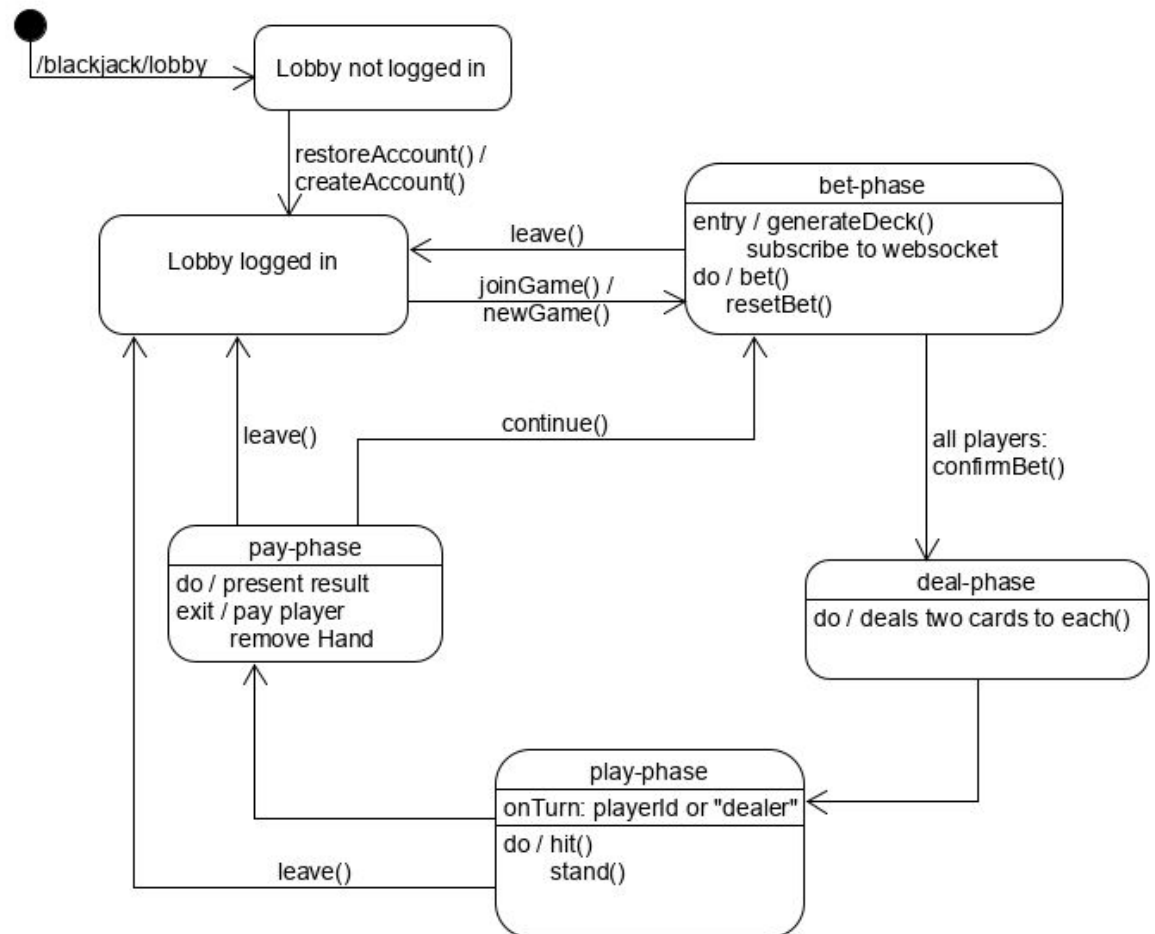
Implementierung - Phasen

- **Lobby:**

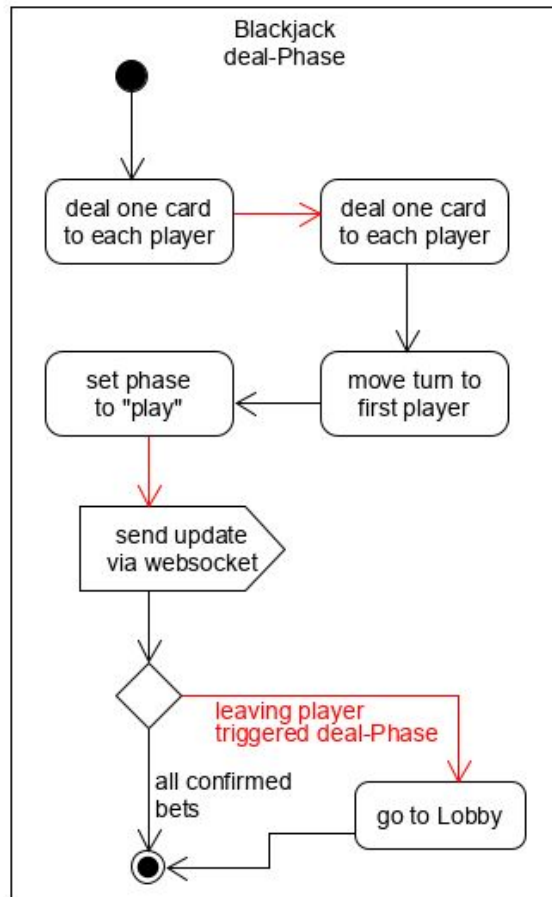
1. not Logged in
2. Logged in

- **Spielphasen:**

1. Bet-Phase
2. Deal-Phase
3. Play-Phase
4. Pay-Phase



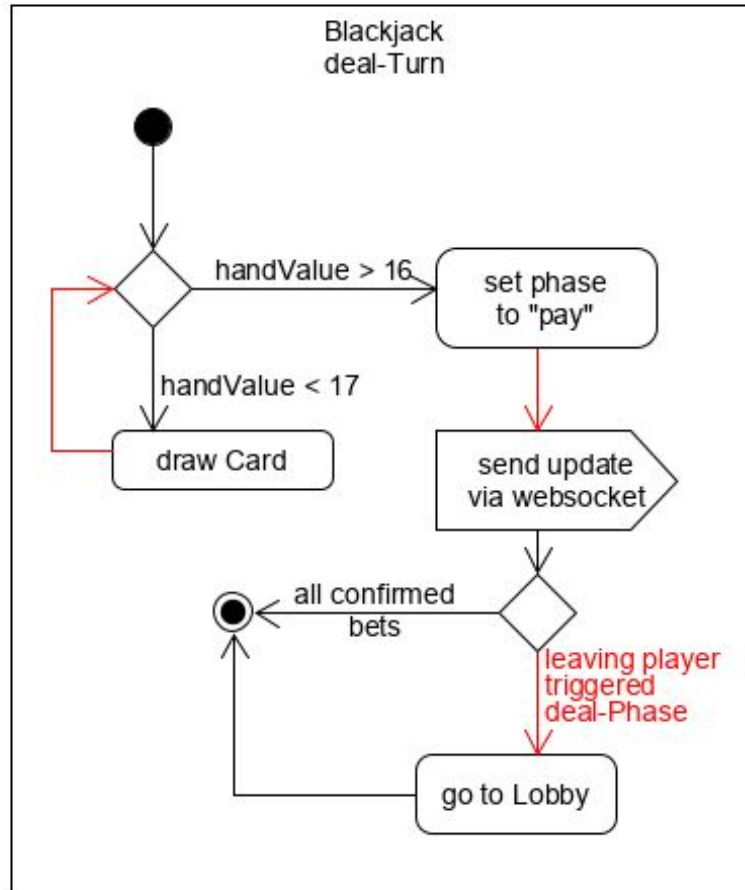
Implementierung - Deal-Phase



1. Eine Karte an alle
(Dealerkarte verdeckt)
→ DB-Update
2. Noch eine Karte an alle
(Dealerkarte offen)
3. Erste*r Spieler*in als
am Zug setzen
4. setze Play-Phase
→ DB-Update
5. Websocket-Update
6. *Optional*: Return to Lobby

Notation	
REST-Call:	
Function-Call:	

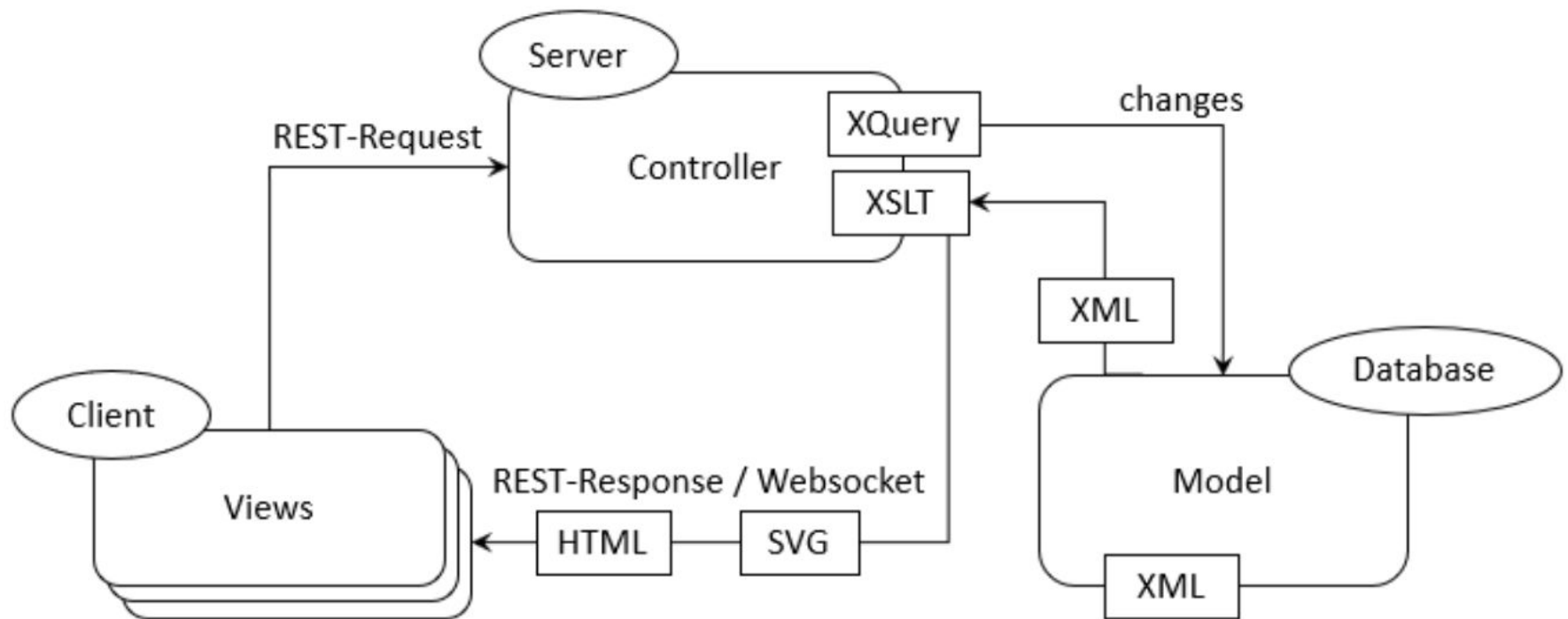
Implementierung - DealerTurn



1. Falls unter 17:
ziehe Karte
→ **DB-Update**
zurück zu 1.
2. sonst:
setze Pay-Phase
→ **DB-Update**
3. Websocket-Update
4. *Optional:* Return to Lobby

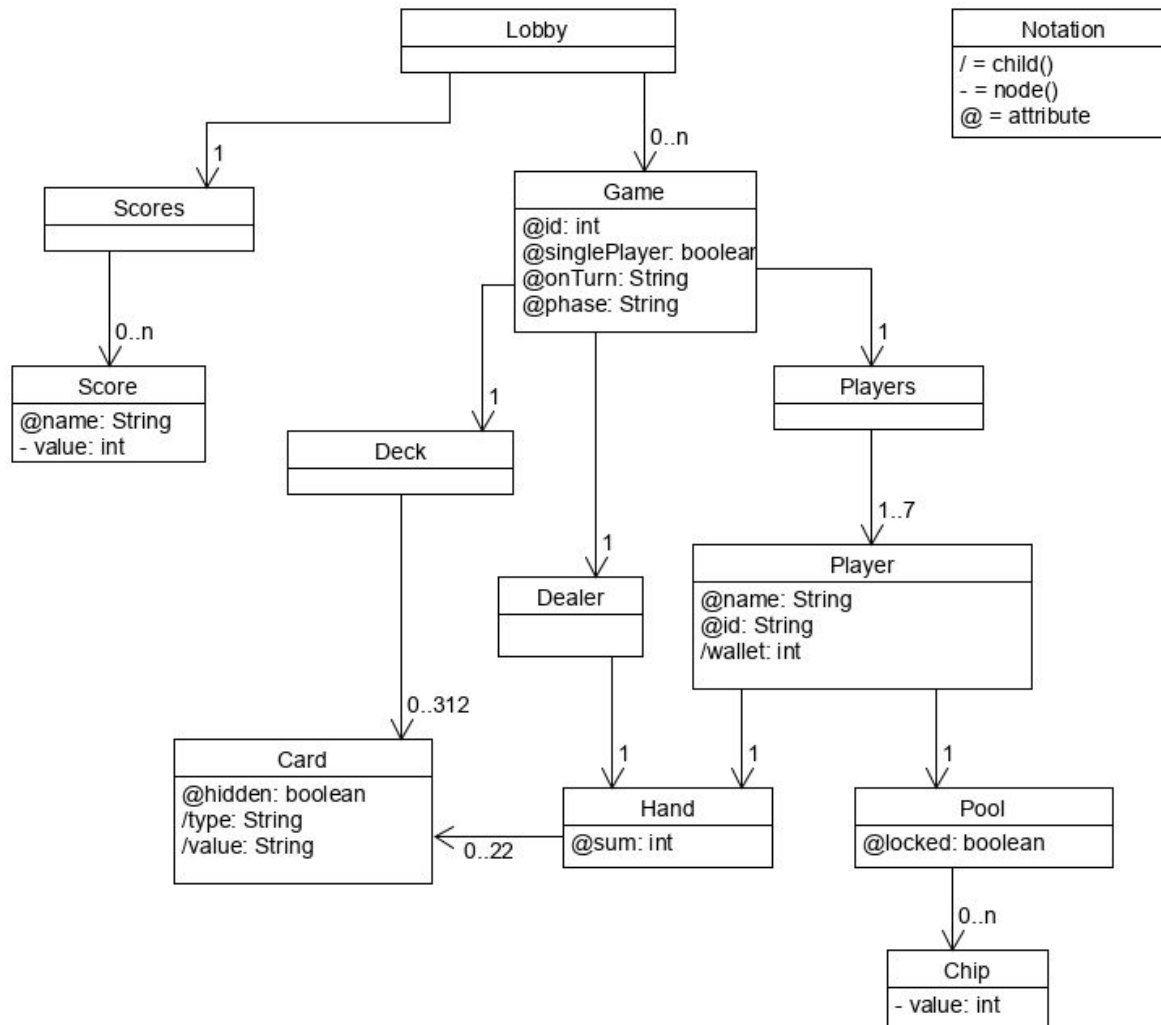
Notation	
REST-Call:	
Function-Call:	

Architektur - Modell - View - Controller



Architektur - Datenmodell

Lobby:



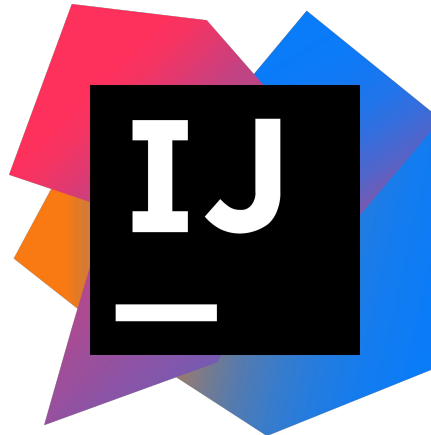
Spiel Demo



blackjack.brzoza.me

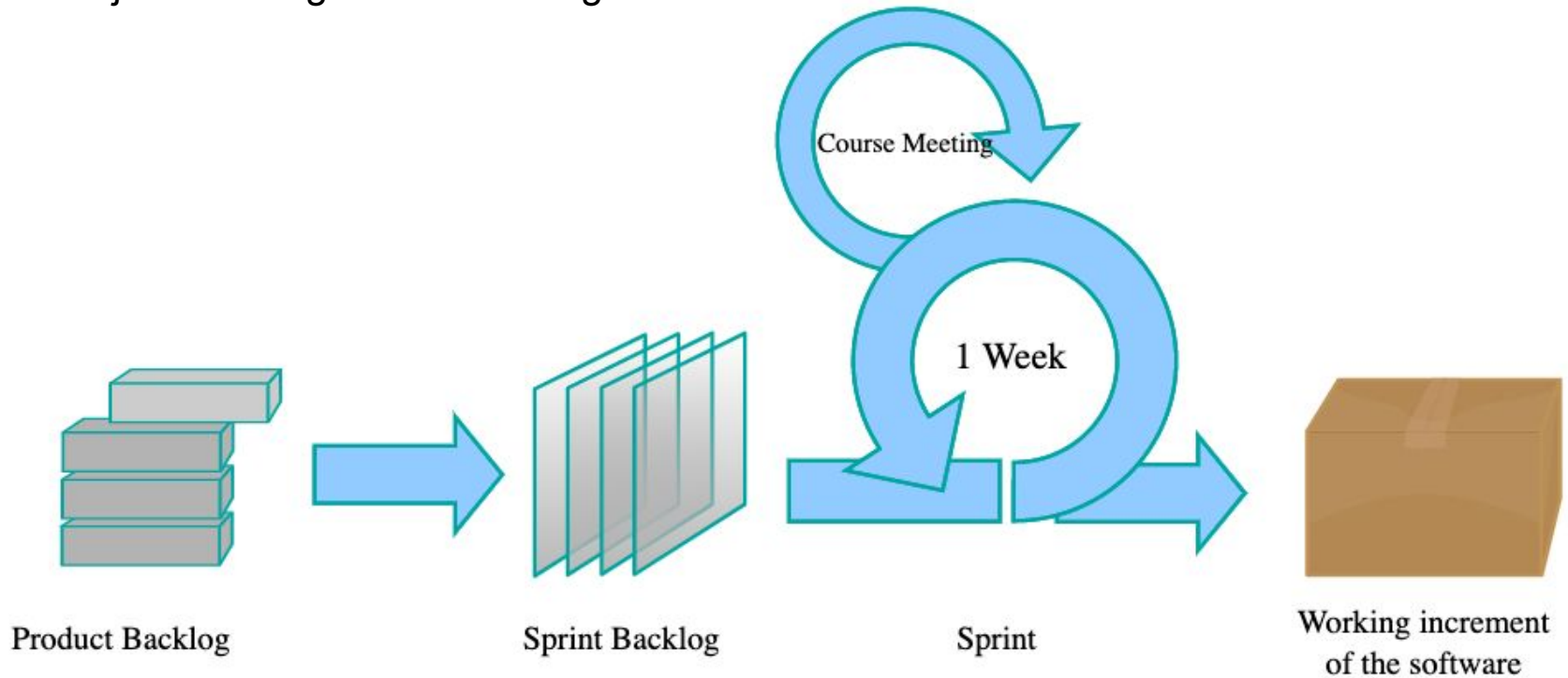
Teamorganisation: Entwicklungsumgebung

- Dateiverwaltung und Zusammenarbeit mit Git-Repository auf LRZ GitLab
- Implementierung der Systemkomponenten mit IntelliJ
- DocBook-Erstellung mit Oxygen



Teamorganisation: SCRUM

- Projekt nach agilem Vorbild organisiert



- Grundanforderungen im Product Backlog verankert
- 1 Arbeitsblatt = 1 Sprint, Aufgaben im Spring Backlog

Fazit

- Projekt nach agilem Vorbild organisiert: **SCRUM**
- Inkrementelle Entwicklung neben der Einführung von neuen Technologien sehr hilfreich

Herausforderungen

- Dynamische Erstellung der Karten
- Setup von Tools (DocBook, BaseX STOMP, xquerydoc)
- Unterschiede in der Darstellung bei Browsern und Betriebssystemen

Reflektion

- Nur XML-Technologien eher ungeeignet für ganze System
- Einzelne Technologien nützlich auch mit anderen Programmiersprachen:
 - dynamische Erzeugung von HTML und SVG mit XSLT
 - XML-Datenbank mit XQuery