Protokoll Team-Meeting

Thema	Datum	Ort	Anwesende Mitglieder	Abwesende Mitglieder	Protokollführer
Technologierecherche	25.11.2019	Li107	Jonas Gwozdz Erik Heldt Linus Herterich Julius Hohlfeld Lennart Buchmann Nils Buxel Tim Henning David Koch Matthias Berger Alaa Aldin Karkoutli Manuel Eckert Julius Jolig Alex Hofmann	Martin Bingel	Matthias Berger

Zusammenfassung:

1. Canvas
2. Django/Python
3. Ruby on Rails
4. Angular.js
5. Coding Guidelines
6. mermaid.js
7. Rete.js
8. cytoscape.js
9. bpmn.js
10. D3.js
11. InfoVis
12. Diskussion
13. Rechercheverteilung

Canvas

- HTML5 basiert
- Klasse in Java

- Einfach, jedoch limitiert
- Zeichnung von einfachen Formen mittels Abfolge von Anweisungen
- Keine Interaktivität
- Performancelastig
- → kaum geeignet

Django/Python

- Python
    - MultiParadigmensprache
    - Skriptsprache
    - Datentypen von Variablen sind an ihre Werte geknüpft
    - Richtige Formatierung von Code ist essenziell
    - Einarbeitungszeit ca. 10-15 Stunden
- Django
    - Sehr Backendlastig
      → Perfomancelastig
    - Integration von Datenbanken ist möglich
    - Gute Dokumentation bei Stackoverflow
    - Simpel und wenig interaktiv
    - Export problematisch
- → Vermutlich nicht sinnvoll

Ruby on Rails

- Framework für Ruby
- Speziell für Webanwendungen
- Keine Vorkenntnisse von Ruby nötig
- Schneller Einstieg möglich
- Einfach, jedoch sehr eingeschränkt
  → „Rails Way“
- Schlechte Dokumentation
- Schwierige Verwaltung von großen Projekten
- Mittlere bis Lange Einarbeitungszeit

- → Wenig geeignet

Angular.js

- Angular mit Typescript (Google) möglich
- Änderungen werden sofort übernommen (WYSIWYG)
- Viel Automatisierung
- Sehr Modular
- Gute Dokumentation (Tutorials)
- Singlepage Design
- Probleme:
  - Keine freie Software für IDE
  - Kostenpflichtige Plugins
  - Eingeschränkte Browserwahl
- Einarbeitung mit JS-Kenntnissen mittel

Coding Guidelines

- s. Wiki

Mermaid.js

- Framework zum Erstellen von Diagrammen und Graphen
- Mittelmäßig große Community
- Graphen können in verschiedenen Formaten exportiert werden
- Java Script und HTML Kenntnisse nötig
- Sehr einfachen
- Nicht interaktiv
- Für Graphische Darstellung geeignet

Rete.js

- Webbasierter Editor für Graphen
- Spärliche Dokumentation
- kleine Community
- Schwierige Einarbeitung
- Funktionsansprüche sind erfüllt

Cytoscape.js

- Reines Java Script Framework

- Voll umfassende Library
- Stark optimiert
- Gute Browserunterstützung
- Sehr gute Dokumentation
- Import/Export möglich
- hohe Aktualität
- Kantengewichte bereits implementiert
- Algorithmen für Graphenanalyse bereits implementiert
- Kaum Nachteile
  - Kein Editor

bpmn-js

- Zwei Versionen:
  - pre-packaged
    → ungeeignet
  - npm (relevant)
- Bietet drag-and-drop Funktionalität
- Im-/Export mittels XML möglich
- geringer Umfang
- einfache Erweiterung

D3-js

- Gute Dokumentation
- Einbindung mittels HTLM sehr einfach
- Nicht sehr umfangreich
  → Plugins
- Nur Darstellung
  → Kantengewichte mittels Plugin
- Im-/Export mittels JSON

InfoVis

- Statische JSON-Graph Struktur
- Mangels Redner keine weitere Betrachtung

Diskussion

- Cytoscape und D3-js scheinen am vielversprechendsten
  → Genauere Betrachtung beider Frameworks
- Clientbasierte Entwicklung erscheint sinnvoller als Serverbasierte

    - evtl. Templates auf Server
- Eine Einheitliche IDE sollte genutzt werden
    - Mittels Containervirtualisierung
- Sprachen:
    - Java Script
    - CSS
    - HTML
- IDE:
    - Webstorm
    - Visual Studio Code
- Containervirtualisierung:
    - Docker
    - Vagrant
- (Docker/Vagrant)→ (Webstorm/Visual Studio Code)

Rechercheverteilung

- Webstorm - Alaa Aldin Karkoutli
- Visual Studio Code - Jonas Gwozdz
- Docker - Tim Henning
- Vagrant - Linus Herterich
- Cytoscape.js - Lennart Buchmann, Nils Buxel, Matthias Berger
- D3-js - Erik Heldt, Julius Hohlfeld, David Koch