

Methodik, die verfolgt wurde, um Lösung zu entwickeln -> ich glaube, den Hevner Artikel (Design Science in Inf Sys research) habe ich dir eh geschickt, eignet sich als Methode. Verbinde einfach sein abstraktes Konzept mit dem, was du tatsächlich gemacht hast (was war die Ausgangssituation (Legacy code mit Angular), wie hat dein Suchprozess ausgesehen (wie bist du auf react gekommen, was waren die Alternativen,...), Was kam aus der globalen Knowledge Base (react), wie kamst du zu deiner Synthese aus react+rdfstore-js+angular)

Design science in information systems research

Hevner et al (2004)

Meta: COIN-Antrag hat das Paper hergenommen und interpretiert was das im Falle des konkreten Projektes bedeutet.

see `./hevner_summary.md`

Process

1. peacemems notifies me of react
2. reading that, i also stumbled across flux
 - flux-article talks about problems with angular/bi-directional data-bindings resonates (same problems when debugging prev prototype) (?)
3. new ulf screens → we'll need to rewrite (?)
4. rewriting with the old angular setup (angular 2 isn't production ready)
5. pre-compilation (js, scss) and bundling setup
 - we also switched away from bootstrap, as we'd need to modify it's styles that heavily anyway
6. actually: when stores and synching via the ws became a thing, started researching flux, ended up stumbling across redux (#342)
7. read up on redux and ways to integrate with existing code-base
8. implementation
 - Frankangular - the Migration Process. Reducing angular to a rendering stage.
 - Frankangular - Duplicate imports :{
 - Migration:
 - Reducing bootstrap usage.
 - Promises: \$q to native.
 - started with router and core reducers(?)
 - a lot of mocking → smooth collaboration
9. usability tests (?) not really part of architecture

10. Meta: möglichst so, dass ich nichts mehr machen soll

11. Meta: Write in a way that large parts can be used as WoN-documentation(?)

12. Meta: always spell in full first names of female authors for references

13. Meta: repeat important points with different wordings.

“iteratively identifying deficiencies in prototypes and creatively developing solutions to address them”
(Markus et al., 2002)

es6-includes make bundling a lot easier (no endless include lists in index.jsp anymore)

how did we migrate, step-by-step (central redux architecture first, then add components, write import wrapper for won.js, restructure linkeddata-service.js)

medium.js text field

rdfstore-js:

- use it for caching but not as redux store
- accessing it is asynch (reducers are synchronous)
- not all app data is described in rdf

compare with other architectures (angular 1.X, flux, cyclejs' mvi, elm,...)


how did co-workers deal with it? ease of use?

interaction/integration with project mngmt workflows. e.g. pull-requests, mocking,...

more difficult architectural decisions:

- Routing
- Rdf-store
- Access Management

Relevant Github-Issues

- [The New Code Base Structure - Structure Diagrams, Refactoring and more](#) (#151)
- [Map widget](#) (#222) and [marker clustering](#) (#227)
 - leafletjs and osm
- [Address forms](#) (#226)
- [Password-retyping unnecessary if reset-via-email works](#) (#264)
- [Experiences with contenteditable](#) (#278)
- [Angular 2.0](#) (#300)
- [Precompilation and Tooling \(Bundling, CSS, ES6\)](#) (#314)
 - bundling, svg sprites, sass, es6,  - why and how?

- SASS and BEM. Also address Semantic CSS (!)
- [SVG-sprites](#) (#318)
- [Template Parsing Performance](#) (#319) - jsx
- [Speech-Bubble-CSS](#) (#333)
 - Afaire we now use a better version by simply rotating a div with a border.
- [Documentation-generator](#) (#337)
- [Actions/Stores and Synching](#) (#342)
 - meta: figures in issue need updates
 - this is the issue that triggered the redux research.
 - Redux based on Elm's Model-View-Intent. Also used by CycleJS. The parts (Action-Creators / Actions, Reducers, Views). Insights on handling side-effects (e.g. server-side interaction)
 - dealing with rdf-store
- [Routing and Redux](#) (#344)
- [chrome's security](#) (#372)
- [WebSocket only created before login](#) (#381)
- [Direct link to need.](#)
- [nicer urls via html5mode in ui-router](#) (#520)
- [Speed up build](#) (#577) aka "`jspm install`" is slow when you need to run it on every build"
- [Page-load performance optimisation](#) (#546)
- [Human-friendly timestamps](#) (#549) → tick actions
- [Load data selectively](#) (#623) – Paging
- [Flatten content-node of needs](#) (#719)
- [direct link to conversation not working](#) (#728)
- Unify Directives: Overview » Incoming Requests and Matches List
- Unify Directives: Chat and Incoming Request and Outgoing Request
- [Usability Tests of Demonstrator](#) (#752)

Future work

- [HTTP Batching](#)
- [Slim Owner-Server](#)
- [Make WoN-app pinnable to home screen](#)
- web-workers / caching
- accessibility