Entwurfsbeschreibung
Nachrichtenkommunikation für das THW

na17b

Inhaltsverzeichnis

1 Allgemeines

Die Anwendung soll den gegebenen Nachrichtenfluss in einer Notfallzentrale des THW nachbilden.

2 Produktübersicht

Im oberen Bereich befindet sich eine Kopfleiste in THW-Farben, welche eine von zunächst drei verfügbaren Rollen anzeigt. Am linken Bildrand befindet sich ein Menü, welches den Wechsel zwischen Übersicht, Maske zur Erstellung eines neuen Formulars, sowie der Rollenauswahl ermöglicht. Die Übersichtsseite vermittelt einen schnellen Überblick über alle hinterlegten Vordrucke, eingeschränkt auf Verfasser, Datum, Uhrzeit und eine gekürzte Version des Inhalts. In der ersten Spalte befindet sich ein Button, über den später der Vordruck zur genaueren Ansicht bzw. Bearbeitung ausgewählt werden können soll. Die Formularmaske bildet den Vierfachvordruck realitätsnah nach. Die Leitrichtung (eingehend oder ausgehend) kann über einen Switch am oberen Rand ausgewählt werden und hat Einfluss auf die verfügbaren Felder. Am unteren Rand befindet sich ein provisorischer 'Abschicken'-Knopf, welcher die eingegebenen Daten zunächst nur temporär abspeichert. Es findet derzeit auch noch keine Überprüfung der Daten hinsichtlich Konsistenz und Vollständigkeit statt. Die Rollenauswahl beschränkt sich im Vorprojekt auf ein simples Dropdown-Menü und dient nur zur technischen Demonstration.

3 Grundsätzliche Struktur

Der Quelltext der Anwendung ist serviceorientiert aufgebaut. Hierzu werden Container verwendet welche die jeweiligen Services kapseln. Zwingende Voraussetzung in der Entwicklungsumgebung ist:

- Docker
  - Bevor die Anwendung ausgeführt werden kann müssen auf dem ausführenden Computer (Host) die folgenden Programme installiert sein:
    - docker
    - docker-compose
  - Die Anwendung ist auf zwei Docker-Container verteilt:
    - Server- Container der das Frontend und Controller-Logik implementiert
    - Daten-Container der den SPARQL-Endpoint implementiert

4 Struktur- und Entwurfsprinzipien einzelner Pakete

Frontend

Das Frontend ist in JavaScript geschrieben und verwendet das Vue.js-Framework. Als Erweiterungen kommen Vuex, sowie Vue-Router zum Einsatz. Die Verwaltung des Quelltextes sowie das kompilieren geschieht via webpack. Zur Vereinfachung wird auf die Bibliothek von Element-UI zurückgegriffen. Der Quelltext des Frontends befindet sich im Ordner `src/frontend`. In diesem Ordner befinden sich neben einem weiteren Ordner `src` die von webpack benötigten Dateien und Ordner; diese werden im Folgenden nicht weiter beschrieben und sind weitgehend Standard. Der Ordner `src` unterteilt sich weiter in folgende Struktur:

```
./src/frontend/src/
├── api/
├── assets/
├── components/
├── router/
├── sparql_help/
├── store/
│   ├── actions.js
│   ├── state.js
│   ├── getters.js
│   ├── mutations.js
│   └── index.js
├── test-examples/
├── App.vue
└── main.js
```

- `api` Quistore-Adapter
- `assets` Mediendateien (aktuell nur das THW-Logo)
- `components` Vue-Komponenten
- `router` Konfiguration von vue-router
- `sparql_help` SPARQL Umwandler
- `store` Konfiguration von Vuex
  - `actions.js` formuliert einen POST Request an Quitstore
  - `state.js` deklariert globale Zustandsvariablen
  - `getters` stellt Funktionen zum Abrufen der Variablen bereit
  - `mutations` enthält Funktionen zum Bearbeiten des Zustandes
  - `index.js` fügt die obigen Dateien zusammen zu einem globalen Store
- `test-examples` Beispieltests
- `App.vue` Wrapper-Komponente für das Frontend
- `main.js` Einstiegspunkt für das Programm, enthält alle Importe

```
Frontend - Arbeitsverzeichnis von Webpack
├── dist - Kompilierte und minimierte Webapp
└── restliche - Konfiguration des Webpack oder node-module, wird in der Regel
    nicht angefasst
```

5 Datenmodell

Als Datenmodell wird RDF verwendet. Die Eingaben aus der UI werden über eine SPARQL-Schnittstelle im Quitstore in Tripel-Form gespeichert und ausgelesen. Der Quitstore wird über `Projekt/quitstore.sh` lokal gestartet. Jeder Datei wird eine ID zur Identifizierung zugeordnet. Der Quitstore stellt die Daten auf Anfrage als JSON-response zur Verfügung.