Lehrstuhl für Pervasive Computing Systems

Teco

Tobias Röddiger
Dr. Paul Tremper

Anwenderorientierte Nutzerschnittstelle für Luftqualitätsdaten

Implementierung

Anna Csurkó
Jona Enzinger
Yannik Schmid
Jonas Zoll

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument dokumentiert die Änderungen an dem Softwareprojekt
Anwenderorientierte Nutzerschnittstelle für Luftqualitätsdaten
während der Implementierung. Diese ist über einen Zeitraum von 4 Wochen entstanden wobei der Implementierungsplan nach Wochen gegliedert war.

Der Plan dafür findet sich im gleichnamigen Kapitel, aufgrund von Terminkollisionen mit Klausuren sowie als Zeitpuffer wurde darauf geachtet die verfügbare Zeit nicht vollständig zu verplanen.
Im Kapitel *Änderungen* wird genauer auf die Entwurfsentscheidungen eingegangen die aus verschiedenen Gründen nicht in der Software umgesetzt wurden.
Im Kapitel *Herausforderungen* werden Verzögerungen gegenüber der Planung und ihre Ursachen erwähnt, beispielsweise Eigenheiten von React und TypeScript die beim Entwurf nicht bekannt waren.

2 Plan

2.1 Woche 1

2.1.1 Plan

- Anwendungsgerüst erstellen
- Model-Teil implementieren
- Unittests für Model
- Komponenten für View mit Mockdaten

2.1.2 Fazit

- Keine größeren Verzögerungen
- Einarbeitung mit verwendetem Toolset (Git, NodeJS, Jest)
- Komponenten sind größtenteils fertig

2.2 Woche 2

2.2.1 Plan

- FROST-Querys formulieren
- Controller implementieren (außer FROST)
- Unittests wo möglich für Controller
- Mock DataProvider mit Zufallsdaten erstellen

2.2.2 Fazit

- Verzögerungen insbesondere bei Datenspeichern und -laden
- Mockdaten erstellt, Kartenfunktionalitäten vollständig
- Erste Version der Querys, Optimierung erforderlich

- Unittests nur an wenigen Stellen sinnvoll einsetzbar
- MapConfigurations implementiert
- An einigen Stellen noch Platzhaltercode, muss ausgetauscht werden

2.3 Woche 3

2.3.1 Plan und Stand am Ende

2.3.2 Fazit

2.4 Woche 4

2.4.1 Plan und Stand am Ende

2.4.2 Fazit

3 Änderungen

3.1 FeatureProvider

FeatureProvider

Die Funktionalität dieser Klasse sollte ursprünglich in Controller.Frost.DataProvider enthalten sein und nach außen versteckt. Da die Features aus den Konfigurationsdateien allerdings logisch nicht vom Server abhängen wurden sie ausgelagert. Dies ermöglicht außerdem den Zugriff auf die Features auch außerhalb des FROST-Pakets.

Methoden

- `static getInstance()`
  Liefert die Singleton-Instanz zurück oder erstellt sie.
- `constructor()`
  Initialisiert den Feature-Speicher
- `getFeature(id : string) : Feature|undefined`
  Das gespeicherte Feature falls es bereits geladen wurde. Sonst wird zunächst das Feature aus der Konfigurationsdatei geladen. Schlägt dies fehl wird 'undefined' zurückgegeben.

3.2 MapController

MapController

Skala und Viewport werden von außen lesbar gemacht. Damit muss MapPage keine eigene Kopie bereithalten.

Hinzugefügt

- `getScale() : Scale`
  Die aktuelle Skala.
- `getViewport(): Viewport`
  Der aktuelle Viewport.

3.3 OnSearch(string) für Suche

OnSearch(string) für Suche

Die eigentliche Suche findet nun außerhalb der Komponente statt. Damit wird die Komponente leichter für verschiedene Anwendungen wiederverwendbar.

Hinzugefügt

- `Search.Props.onSearch(term: string) : void`
  Wird bei Klick auf den Such-Button oder Drücken von Enter aufgerufen. Enthält den aktuellen Inhalt der Suchbox.
- `MapView.onSearch(term: string) : void`
  Ruft die Suche im MapController auf und aktualisiert die Seite.

3.4 Legend

Legend

Der Ausschnitt der von der Legende angezeigt wird soll flexibel sein.

Hinzugefügt

- `Props.min : number`
  Das untere Ende der Legende
- `Props.max : number`
  Das obere Ende der Legende

3.5 Map

Map

Ursprünglich sollte der Viewport über die Mitte der Pins/Polygone bestimmt werden. Um die Karte von Anfang an auf die letzte Position zu zentrieren wird der Viewport direkt übergeben.

Hinzugefügt

- `Props.viewport : Viewport`
  Viewport mit dem die Karte initialisiert wird.

3.6 FeatureSelectInit

FeatureSelectInit

Die FeatureSelect Auswahl benötigt die aktuellen Werte um sie standardmäßig auswählen zu können.

Hinzugefügt

- `FeatureSelect.Props.startConf?: { conf: string; feature: string }`
  Die Startwerte der Auswahlboxen. Optional.
- `MapController.getFeatureSelectConf(): conf: string; feature: string`
  Gibt Werte für die Auswahlboxen basierend auf der Konfiguration aus.