UNIVERSITÄT LEIPZIG

SOFTWARETECHNIK-PRAKTIKUM

Entwurfsbeschreibung

SPE16: Gebäude-Navigator für Leipzig

15. August 2016

Inhaltsverzeichnis

1. Allgemeines

Im Rahmen des Softwaretechnikpraktikums wird ein Prototyp einer Web-Anwendung für den Behindertenverband Leipzig e.V. entwickelt. Es liegt ein Datenbestand über die Zugänglichkeitsbeschränkungen von mehreren hunderten Objekten von der Stadt Leipzig zu Grunde. Dieser Datenbestand soll mit Hilfe der Software gefiltert und dargestellt werden können, sodass eine Übersicht über die zugänglichen Objekte entsteht. Bei dieser Web-Anwendung steht Barrierefreiheit an vorderer Stelle, damit eine große Bandbreite an Personen erreicht werden können.

2. Produktübersicht

Da die Nutzung auf mobilen Endgeräten im Vordergrund steht, wird diese Web-Anwendung als Single-Page-Application entwickelt. Dies sorgt für eine flüssige Bedienung und ermöglicht den Einsatz auf den populärsten Smartphones, unabhängig vom Betriebssystem. Die Software führt ihre Programmlogik in JavaScript aus, da dies recht einheitlich auf den gängigsten Browsern ausgeführt werden kann. Die Darstellung wird über eine einzige HTML-Seite abgewickelt. Zur Formatierung wird CSS benutzt. Die Daten über die Objekte, sowie die HTML-Seite und das Skript zur Ausführung werden sich später auf einem Web-Server befinden und über das Internet zugänglich sein. Der Nutzer wird auf einer Oberfläche verschiedene Filter einstellen können und sich dann die gefilterten Objekte auf einer Karte, oder in einer Listenansicht anzeigen lassen. Die Position des Nutzers soll, wenn möglich über das Funknetz ermittelt werden und bei der Anzeige der Filterergebnisse beachtet werden, um die Bedienungsfreundlichkeit zu erhöhen.

3. Grundsätzliche Struktur- und Entwurfsprinzipien

3.1. Architekturmuster

Als Architekturmuster wird MVC realisiert. Die View-Komponente wird dabei über die React Bibliothek erstellt. Zur Veränderung und zum Halten von Daten wird das Flux Muster verwendet. Dieses Muster wurde aufgrund der Stabilität und leichten Skalierbarkeit verwendet.

3.2. Programmiersprache

Die gesamte Anwendung ist in JavaScript nach ECMA Script 6 Standard geschrieben, da dies die aktuellste Version von JavaScript zur Zeit der Projektdurchführung ist.
HTML 5 und CSS 3 werden zum Erstellen der Oberfläche verwendet.

3.3. Entwicklungsumgebung

Das node.js Framework dient als Entwicklungsumgebung. Darüber werden Bibliotheken, Transpiler und andere Entwicklungshilfen bereitgestellt.
Webpack wird als Modul-Bundler genutzt. Dieser Prozess sucht alle Abhängigkeiten zwischen den einzelnen Bibliotheken und unserem Code zusammen und erstellt daraus eine einzige Datei. Diese Datei wird dann in ECMA Script 5 transpiliert, damit auch weniger aktuelle Browser den Code ausführen können. Am Schluss wird diese Datei noch komprimiert. Dieses Kondensat bildet dann die fertige JavaScript Datei, welche auch beim Nutzer – eingebunden auf einer HTML Seite – zum Einsatz kommt.
Sobald das Git-Repository heruntergeladen und node.js installiert wurde, kann mittels des Konsolenbefehls (im Projektordner) “npm install“ der Paketmanager gestartet werden. Dieser liest die package.json und lädt dann alle benötigten Bibliotheken herunter und legt diese im node_modules Ordner ab. Wurde der Quellcode des Projektes geändert so muss der Befehl “webpack“ ausgeführt werden, damit der Bundler eine neue finale JS-Datei erstellt. Diese Datei liegt im src Ordner und heißt client.min.js.
Um die Arbeit mit webpack noch effizienter zu machen kann ein webpack-Server über node.js installiert werden. Mittels “npm run dev“ wird dieser dann gestartet und bietet die fertig transpilierte und gebündelte Web-Anwendung auf der Adresse “localhost:8080“ an. Dieser Server beobachtet nun den Quellcode auf Veränderungen und aktualisiert automatisch die Web-Anwendung sobald dies nötig ist.

4. Struktur- und Entwurfsprinzipien einzelner Pakete

4.1. React

Durch React ist es möglich einfacheren Code zu schreiben und Bestandteile weniger miteinander zu verschränken. Der zentralste und einzige Baustein von React sind Komponenten. Komponenten werden mit React.createClass() erzeugt. Die wichtigste Funktion einer Komponente ist die render() Funktion. Dort wird das Markup für die Dom-Präsentation definiert. Die Komponente List wird zur Darstellung von Listenobjekten verwendet. In der Komponente OSMMap wird hauptsächlich die OpenStreetMap eingebunden und die Darstellung der gesuchten Objekte mittels Marker realisiert. Die Komponente Filter wird dann alle Filtermöglichkeiten des RDF Datensatzes anbieten.

4.2. Flux

Flux ist ein Muster zur Lenkung des Informationsflusses in einer React-Anwendung. Dabei wird Wert daraufgelegt, dass der Fluss immer nur in eine Richtung geht, damit selbst bei komplexen Anwendungen die Übersichtlichkeit gewahrt bleibt.

Bestandteile von Flux:

Action:	JavaScript-Objekt, welches von einem ActionCreator erzeugt wird. Durch das „type“-Attribut können Stores relevante Actions erkennen.
Dispatcher:	Verteilt eine empfangene Action an alle Stores.
Store:	Verwaltet die Applikationslogik und dient als Datenquelle für React-Komponenten. Die Änderungen des Stores werden für den Benutzer über die React-Komponenten sichtbar.
View:	React-Komponenten werden durch den veränderten Store dazu aufgefordert sich selbst neu zu rendern. Es wird zwischen zwei Arten von Komponenten unterschieden. „Vereinfachte“ React-Komponenten erhalten ihre Daten mittels Properties. „Smarte“ React-Komponenten können Actions dispatchen.

MainPageAction → Dispatcher → MainPageStore → View → MainPageAction

4.3. DataHandler

Der DataHandler übernimmt das Parsen der CSV Datei. Über das Tool PapaParse werden die CSV-Daten in JSON-Objekte transformiert und in ein Array abgelegt. Das Resultat ist ein Array aus Objekten, deren Elemente die Attribute als keywords und ihren zugehörigen Wert enthalten. Die Daten werden in dieser Form im Store für die weitere Nutzung bereitgestellt.
Der DataHandler wird bei Programmstart vom MainPageStore initialisiert und stellt die Daten asynchron bereit.

4.4. MainPageStore

Durch die Verwendung der Flux Architektur übernimmt der MainPageStore die Aufgaben des Controllers wie bei einer klassischen MVC Architektur.
Bei jedem Start der Anwendung wird eine Instanz vom DataHandler erzeugt und füllt den MainPageStore. Der Dispatcher überführt einzelne Actions von der MainPageAction weiter und leitet somit eine Veränderung des Stores ein. Diese Veränderungen werden dann an die View, also an die einzelnen React-Komponenten, weitergeleitet.

4.4.1. Datenhaltung

Die Datenhaltung erfolgt in einem Array, welches alle übergebenen Objekte aus dem DataStore enthält. Durch einzelne Actions werden dann die relevanten Daten extrahiert, in separate Arrays gespeichert und an die Komponenten List, OSMMap und Filter geschickt. Das MainArray bleibt somit beständig und unverändert. Beim Schließen der Anwendung wird auch der Store geschlossen. Es findet also keine persistente Speicherung statt.

4.4.2. Programmlogik

Die Programmlogik wird wie die Datenhaltung im MainPageStore implementiert. Jedoch erfolgt eine strikte Trennung zwischen Datenhaltung und der Logik. Die Datenlogik beinhaltet hauptsächlich die Filter Funktionen, State Handling sowie das Initiieren des Stores.

5. Datenmodell

Das Datenmodell folgt dem von Facebook entwickelten Datenflussmodell flux. Die für die Anwendung wichtigen Daten werden über eine CSV Datei vom BVL zur Verfügung gestellt. Die enthaltenen Informationen werden über den DataHandler im MainPageStore abgelegt. Über den MainPageStore bekommen die Komponenten Filter, OSMMap und List die benötigten Daten. Diese Daten werden über den ActionHandler zugewiesen. Die Komponente List sowie OSMMap bekommen die Ergebnisse des Filters in angepasster Form übertragen. Diese Ergebnisse sind Gebäudeinformation um Marker in die OSM-Karte zu setzen oder in der Liste anzeigen zu lassen. Die Beschreibung des Datenmodells wird durch das Klassendiagramm in Anhang A1 vervollständigt.

6. Glossar

Prototyp ist ein lauffähiges Stück Software des Gesamtsystems. Es wird dazu verwendet dem Kunden und Entwicklern eine Basis für eine bessere Kommunikation anhand von konkreten Dingen, nicht nur abstrakten Modellen, zu gewährleisten.

Single-Page-Application (engl. Einzelseiten-Webanwendung) Webanwendung, die aus einem einzigen →**HTML**-Dokument besteht und deren Inhalte dynamisch nachgeladen werden

Webbrowser sind spezielle Computerprogramme die Website aus dem World Wide Web oder Dokumente und Daten darstellen.

HTML Hypertext Markup Language (engl. für Hypertext-Auszeichnungssprache) - textbasierte Auszeichnungssprache zur Strukturierung digitaler Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von →**Browsern** dargestellt.

CSS (Cascading Style Sheets) ist eine weitere Kerntechnologie für die Erstellung von Websites. Die Sprache beschreibt die Präsentation der Website. Dies beinhaltet die Darstellung der Farben, das Layout oder die Schriftart einer Website auf verschieden Geräten. Das Projekt SPE16 verwendet diese Technologie um die Webanwendung auf verschieden Geräten zu ermöglichen.

MVC (Model View Controller) ist ein Softwareentwicklungskonzept. Es dient als Muster um einen flexiblen Programmentwurf mit späteren Änderungen oder Erweiterungen zu erleichtern und die Wiederverwendbarkeit von einzelnen Komponenten zu ermöglichen. Die Anwendung wird dabei in drei Einheiten unterteilt: Datenmodell (engl. model), Ansicht (engl. view) und Programmsteuerung (eng. Controller).

Node.js ist ein auf dem JavaScript-Compiler Google-V8 basierendes Framework für die Entwicklung von serverseitigen Webanwendungen bzw. die Realisierung von Webservern. Es ermöglicht die Einbindung zahlreicher Module über den integrierten Paketmanager npm.

Transpiler sind Hilfsprogramme, die einen Quellcode von einer höheren Programmiersprache in eine andere höhere Programmiersprache übersetzen.

Bibliotheken sind Sammlungen von thematisch zusammenhängenden Unterprogrammen und -Routinen, die eigenständig nicht lauffähig sind, aber von anderen Programmen über Hilfsmodule angefordert werden können.

React ist eine von Facebook entwickelte JavaScript → **Bibliothek** zur Erstellung von Benutzeroberflächen. Ihr liegt eine modulare Komponentenarchitektur zugrunde, in der jede Komponente ihr eigenes → **virtuelles DOM** erzeugt. Wenn sich Komponenten – etwa durch Nutzerinteraktionen – ändern, versucht das virtuelle DOM nicht die ganze Anwendung, sondern nur das kleinste betroffene Element neu zu rendern.

(Virtual) DOM (Document Object Model) bezeichnet die Schnittstelle zwischen → HTML und dynamischem JavaScript. Es bildet die strukturelle Repräsentation des Dokumentes und ermöglicht die Änderung von Inhalten und der visuellen Darstellung. React-Komponenten arbeiten nicht direkt mit dem DOM des → **Browsers**, sondern erstellen bei einer Änderung einen virtuellen DOM als Abstraktion des eigentlichen DOM. Dieser wird mit dem vorherigen verglichen und minimale Änderungen durchgeführt.

Flux ist eine Architekur für clientseitige Webapplikationen, die View-Komponenten aus → **React** um einen unidirektionalen Datenfluss ergänzt. Sie umfasst die vier Hauptelemente Actions (i.e. Hilfsmethoden, um Daten an den Dispatcher zu übermitteln), Dispatcher (steuert den Datenfluss. Er enthält die Actions und sendet die Nutzdaten zu den Stores), Stores (enthalten Datenmodelle und Logik) und die Views (React-Komponenten, die sich Daten von den Stores holen und sich um das Rendering der Anwendung kümmern).

Unit Testing Framework (auch Modultest oder Komponententest Framework) ist ein Rahmen von Testprogrammen für funktionale Einzelteile einer Software. Diese Module oder Komponenten werden auf korrekte Funktionalität überprüft. Die Test werden anhand vorgegebener Soll-Bestimmungen bewertet.

Assertion System (auch Zusicherung oder Sicherstellung) ist ein Gebilde von Begriffen über die Beurteilung eines Programms. Dieses System wird verwendet um zu beschreiben welche Ergebnisse von Programmteilen zu erwarten sind.

Jest ist ein → **Unit Testing Framework** welches von Facebook im Zusammenhang mit → **React** entwickelt wurde. verbindet die Test Utilities von react und das Assertion System von → **Jasmine 2** um Tests auf den Komponenten durchzuführen.

Jasmine 2 ist ein verhalten orientiertes Softwareentwicklungsframework bei dem anhand von bestimmten Szenarios die Module getestet werden. Es ist das → **Assertion System** welches standartmäßig in → **Jest** verwendet wird.

PapaParse ist ein JavaScript-Tool zum Parsen von CSV-Daten zu JSON und umgekehrt.

Anhang 1: Klassendiagramm