
# Plan ya Concerts

EINE LISTENBASIERTE APPLIKATION ZUR KONZERTORGANISATION

Jonas Hirsekorn, Dominik Klippert | Mobile Anwendungen| Abgabe: 28.06.2019

# Inhaltsverzeichnis

# 1. Aufgabenstellung

Die Aufgabenstellung im Modul: Mobile Anwendungen ist, eine Web-App mit der JavaScript Bibliothek: React zu erstellen. Die Anwendung soll Daten in Form einer Liste anzeigen, speichern, löschen und bearbeiten können. Darüber hinaus soll die Anwendung ein übergeordnetes Thema haben.

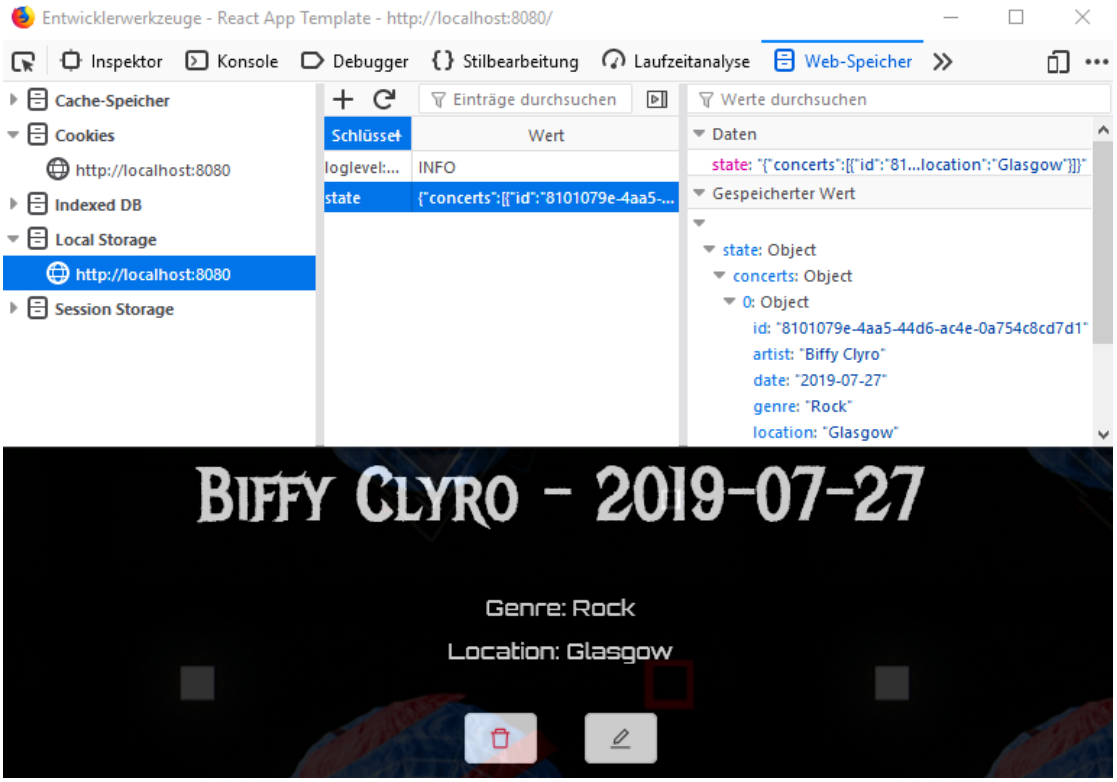
# 2. Vorgehen

Wie geht man bei der Erstellung einer Web-App vor? Zunächst muss ein Thema, das mit den Vorgaben des Projekts vereinbar ist, gefunden werden. Hierbei wurde sich für das Thema Konzertorganisation entschieden. Als nächstes gilt es herauszufinden, welche Funktionen bei der Auflistung von Konzerten wünschenswert sind. Dazu gehören das Eintragen des Künstlers, des Datums, des Veranstaltungsorts und eventuell noch einer Schnittstelle, in die man den persönlichen Lieblingstitel des Künstlers einfügen kann. Im nächsten Schritt wurde eingeschätzt, welche dieser Funktionen in der vorgegebenen Zeit umsetzbar sind. Dabei wurde klar, dass die Schnittstelle, den zeitlichen Rahmen des Projekts überschreitet und sich somit auf das Wesentliche konzentriert wurde.

Im nächsten Schritt wurde eine lokale Entwicklungsumgebung eingerichtet, um die Web-App zu erstellen. Hierbei wurde die IDE[1]: Webstorm[2] benutzt. Nachdem die serverseitige Plattform node.js[3] installiert wurde, konnte ein React-Projekt angelegt werden. Dazu wurde ein Grundgerüst vom Dozenten bereitgestellt, welches man von dessen github-repository[4] herunterladen konnte. Abgesehen davon, wurden jsx-Dateien bereitgestellt, die bereits Grundfunktionen der Anwendung enthielten. Dazu zählen das Hinzufügen und Löschen von Listeneinträgen. Im nächsten Schritt wurden mehrere Eigenschaften der Listeneinträge hinzugefügt. Dabei wird jeder Eintrag als Objekt betrachtet. Ein solches Objekt kann mehrere Eigenschaften oder auch „properties“ haben. Diese können verschiedene Werte haben, wie z. B. Strings, Zahlen oder Wahrheitswerte. In diesem Fall hat die Liste „concerts“ die Eigenschaften: „id“, „artist“, „date“, „genre“ und „location“.


[1] Itegrated development environment
[2] https://www.jetbrains.com/webstorm/
[3] https://nodejs.org/en/
[4] https://github.com/mjleehh/web-project-skeleton

Nachdem diese Eigenschaften erstellt und wieder gelöscht werden konnten, wurde an der Speichern-Funktion gearbeitet. Da der Anwendung kein Backend zugrunde liegt, nutzt die Anwendung den lokalen Speicher des Browsers, um Daten zu speichern. Hierbei werden im Browser verschiedene Zustände von Objekten gespeichert. Dies geschieht immer dann, wenn ein neues Konzert – ein neues Objekt – hinzugefügt wird. Das heißt, dass die erstellten Konzerte auch nach einem Seiten-Refresh verfügbar sind. Gibt der User Daten in die jeweiligen „Input"-Felder ein und klickt auf den Hinzufügen-Button, erstellt die „createHandler"-Methode, Referenzen zu den Input-Feldern. Über die „dispatch"-Funktion innerhalb dieser Komponente, werden die Eingaben mit der „reducer"-Funktion verbunden. Dort tritt der Fall „CREATE_ELEM" ein und ein neuer Array wird erstellt. Dieser Array enthält ein neues Objekt. Dem Objekt liegen die gleichen Eigenschaften zugrunde. Jedoch haben diese Eigenschaften andere Zustände, nämlich die, die vorher in die „Input"-Felder eingetragen wurden. Die veränderten Zustände, aus der Reducer-Funktion, werden als JSON-String in einer Konstante „serializedState" gespeichert und an die saveState-Funktion übergeben. Anschließend werden diese in dem lokalen Speicher des Browsers eingefügt. Danach

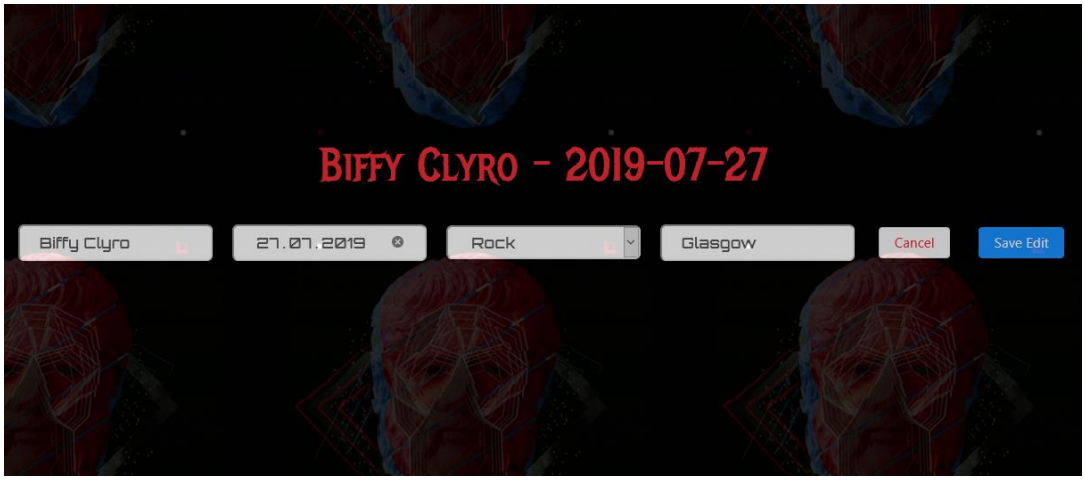



wird in der „loadState"-Funktion überprüft, ob Werte übermittelt wurden. Falls dies der Fall ist, wird der jeweilige String durch „JSON.parse" in ein Array formatiert. Über „store.subscribe" werden alle aktuellen Zustände an die „saveState"-Funktion übermittelt, um die Zustandsveränderung zu überwachen.

Nachdem der User ein Konzert hinzugefügt hat, wird der Name des Künstlers inklusive des Veranstaltungsdatums auf der rechten Seite des Browserfensters angezeigt. Dieser Listeneintrag ist anklickbar und erweitert sich nach unten, wo weitere Informationen angezeigt werden. Dazu gehören die Eigenschaften „Genre" und „Location", sowie ein Löschen- und ein Bearbeiten-Button. Wenn auf den Löschen-

Button geklickt wird, wird die jeweilige ID des angeklickten Listeneintrags an die „reducer“-Funktion weitergeleitet. Dort tritt der Fall „DELETE_ELEM“ ein. Ein neuer Array wird erstellt, in dem alle Objekte hinzugefügt werden, deren ID ungleich der übertragenen ID des angeklickten Listeneintrags sind. Dies hat zur Folge, dass der Eintrag in der Liste nicht mehr angezeigt wird. Durch die Nutzung der „dispatch“-Funktion in der „deleteHandler“-Methode, wird auch der lokale Speicher aktualisiert.

Wenn der User auf den Bearbeiten-Button klickt, werden „Input“-Felder bereitgestellt, in denen die bereits vorhandenen Daten verändert werden können. Außerdem wird ein Abbruch-Button angezeigt. Wird dieser geklickt, wird der Bearbeitungsmodus beendet und die vorherigen Konzertdaten bleiben bestehen. Beim Klick auf den Bearbeitung-Speichern-Button, wird der angeklickte Listeneintrag gelöscht. Die „dispatch“-Funktion wird aufgerufen, wobei in der „reducer“-Funktion der Fall „DELETE_ELEM“ eintritt. Als nächstes werden die neuen Daten, über Referenzen, innerhalb der „updateComponentValue“-Methode als neue Zustände gespeichert. Daraufhin wird ein neuer Listeneintrag mit den eingegebenen Daten erstellt. Somit handelt es sich eigentlich nicht um eine Bearbeitungs-Funktion, da der alte Array gelöscht und ein neuer hinzugefügt wird. Der Zustand der Ausgangsdaten ändert sich und die Liste wird automatisch neu geladen. Die geänderten Daten werden nun angezeigt obwohl sich die Seite nicht neu geladen hat.




# 3. Problemstellungen und Besonderheiten

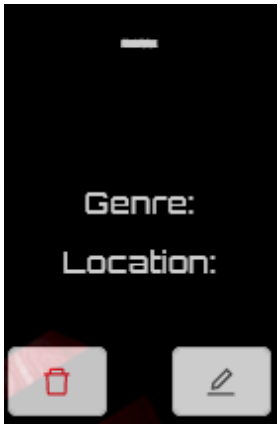
## 3.1 PROBLEME

Wie in der Aufgabenstellung bereits erwähnt, soll die Anwendung Daten in Form einer Liste anzeigen, speichern, löschen und bearbeiten können. Hierzu sei gesagt, dass Probleme bei der Bearbeitungsfunktion in Verbindung mit der Speicherungsfunktion auftraten. Der Grund dafür war, dass zuerst eine veraltete Syntax, nämlich die „refs“-Syntax, verwendet wurde. Dadurch konnte die Bearbeitungsfunktion nicht mit der „reducer“-Funktion verbunden werden. Der lokale Speicher konnte deshalb nicht aktualisiert werden, weshalb bei einem Seiten-Refresh nach dem Bearbeiten wieder die vorherigen Werte angezeigt wurden. Das nächste Problem war, dass davon

ausgegangen wurde, dass ein neuer Fall der „reducer“-Funktion hinzugefügt werden musste.

Ein weiteres Problem tritt ein, wenn die Anwendung zum ersten Mal im Browser geladen wird. Hierbei wird ein leerer Listeneintrag ausgegeben. Dies geschieht aufgrund des „initialStates“, der zu Beginn gesetzt werden muss.

```
const initialState = {
    concerts: [
        {
            id: uuid(),
            artist: '',
            date:'',
            genre:'',
            location:''
        }
    ]
}
```




Um dieses Problem zu lösen, müsste man vermutlich eine Fallunterscheidung vereinbaren, die diesen Listeneintrag nicht rendert. Dieser Fall müsste eintreten, wenn z. B. der Zustand des „artists“ leer ist.

Ein weiteres Problem stellte die Datenspeicherung über den lokalen Speicher dar. Zunächst wurde versucht, einzelne „properties“ in einzelnen Arrays zu speichern. Nach einer Recherche[5] stellte sich heraus, dass ein Array mit mehreren Objekten erstellt und gespeichert werden muss. Eine weitere Schwierigkeit war, den lokalen Speicher je nach Programmablauf (Löschen-, Bearbeiten- und Hinzufügen-Funktion) auf dem aktuellen Stand zu halten.

[5] https://egghead.io/lessons/javascript-redux-persisting-the-state-to-the-local-storage

### 3.2 BESONDERHEITEN

Die Anwendung wurde durch verschiedene Erweiterungen wie „FlipMove“, „Antd“, „CSS“, „Fonts“ und Bilder optisch aufbereitet. Hierbei ist zu erwähnen, dass

```
module: {
    rules: [
        {
            test: /\.jsx?$/,
            use: 'babel-loader',
            exclude: /node_modules/
        },
        {
            test: /\.scss$/,
            use: [ 'style-loader', 'css-loader', 'sass-loader']
        },
        {
            test: /.css$/,
            use: [ 'style-loader', 'css-loader']
        },
        {
            test: /\.(png|jpe?g|gif)$/,
            use: [
                {
                    loader: 'file-loader',
                    options: {},
                },
            ],
        },
        {
            test: /\.(eot|svg|ttf|woff|woff2)$/,
            loader: 'file-loader',
```

verschiedene „module-rules“ in die webpack.config.js eingefügt wurden. Zudem wurde die User-Interface-Library „antd“ und eine Sammlung an JS-Dateien (FlipMove) mittels des Paketmanagers (npm) installiert. „Antd“ stellt vorgefertigte Seitenobjekte wie beispielsweise „Buttons“, „Input-Felder“ oder „Icons“ zur Verfügung. FlipMove animiert Veränderungen im Browserfenster, sofern sich das verändernde Objekt innerhalb der „FlipMove“-Komponente befindet. Das bedeutet, dass der Lösch- und Hinzufügevorgang eines Listeneintrags animiert wird.

Außerdem wurde ein Akkordeon, welches die Listeneinträge öffnet und schließt, programmiert.

## 4. Wie funktionieren React und die Modularisierung?

Bei dem Framework „React“ handelt es sich um eine flexible JavaScript-Bibliothek, die dem effizienten Zusammensetzen von Userinterfaces dient.

Für diese Anwendung wurde zusätzlich die quelloffene JavaScript-Bibliothek „Redux[6]“, verwendet. Diese dient zur Verwaltung und Veranschaulichung von Zustandsinformationen in einer Webanwendung.

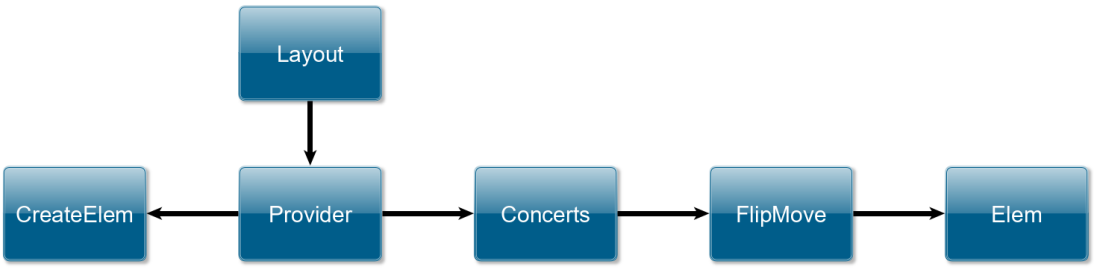
### 4.1 REACT-KOMPONENTEN

In der Anwendung „Plan ya Concerts“ wurden verschiedene Komponenten in ES6 Klassen definiert. Komponenten sind ein fundamentaler Bestandteil von React-Apps. Sie gelten als einzelne unabhängige Bausteine und vermitteln dem Framework, was dem User auf dem Bildschirm angezeigt wird. Der Vorteil von React-Komponenten ist, dass der Code in unabhängige Teile zerlegt werden kann, sodass bei einer Änderung von Daten im Browser, nicht immer die gesamte Seite neu geladen werden muss, wenn etwas Neues angezeigt werden soll. Dadurch lässt sich eine bessere Performance erreichen, da insgesamt weniger Daten neu geladen werden müssen. Komponenten

[6] https://redux.js.org/

beginnen mit einem Großbuchstaben, da in JSX, kleingeschriebene Namen-tags als HTML-tags betrachtet werden. JSX ist eine Erweiterung der üblichen Javascript-Grammatik für React. Es verringert den Schreibaufwand, da HTML-tags in Java-Script eingebunden werden. Dadurch entsteht eine übersichtlichere, HTML-artige Syntax.

Die Struktur einer Komponente kann in einem Baumdiagramm dargestellt werden. Dadurch können Eltern-Kind-Beziehungen zwischen den Komponenten entstehen. Verändert man das Eltern-Element, hat das gleichzeitig Auswirkung, auf das Kind-Element.




Die Anwendung „Plan ya Concerts“ liegt der oben zu sehenden Grafik zugrunde. Der Einstiegspunkt dieser React-App ist die index.html. In dieser Datei befindet sich ein „div“-Container mit der ID „main“. Auf diese ID wird von der index.jsx-Datei aus mittels der ReactDom.render-Methode zugegriffen. Die Methode nimmt zwei Parameter an. Der erste Parameter gibt an, was gerendert werden soll, der zweite wo das Gerenderte dargestellt werden soll. Dabei ist zu beachten, dass die jeweiligen Komponenten ineinander geschachtelt werden müssen, sodass nur eine Elternkomponente existiert. Im Fall von der Anwendung „Plan ya Concerts“ ist das die „Layout“-Komponente. Diese umschließt alle anderen Komponenten. Die dazugehörige Kinderkomponente heißt „Provider“. Diese ist eine von Redux bereitgestellte Komponente, die einen „store“ beinhaltet. Dort werden alle Zustände gespeichert, wodurch es möglich wird, über die Redux-Browser-Erweiterung die verschiedenen Zustände einzusehen. Die Provider Komponente verbindet demnach die Komponenten „CreateElem“ und “Concerts“. In der „CreateElem“-Komponente wird das Eingabe-Formular auf der linken Seite des Browserfensters gerendert. Die „Concerts“-Komponente ist für die Darstellung der Konzertliste verantwortlich. Die Konzertliste besteht wiederum aus der „FlipMove“-Komponente, die allerdings nur für die Animation der in dieser befindlichen „Elem“-Komponente zuständig ist. In der „Elem“-Komponente werden, je nach Fall, die einzelnen Listeneinträge gerendert.

Jede Komponente die nicht nur eine render Methode beinhaltet, sollte eine Konstruktor Methode besitzen, um einen eigenen „state“ definieren zu können. Hierbei ist es außerdem wichtig, die super(props) Methode aufzurufen. Dadurch wird umgehend auf die Elternklasse referenziert und ein Zugriff auf die jeweilige property (this.props) im Konstruktor ermöglicht.