My Songbook

Eine musikerfreundliche App zur Verwaltung von Lead Sheets

Alfons Becker, Carsten Dachner, Egon Fernandez

Inhaltsverzeichnis

1 Allgemeine Beschreibung

1.1 Sinn und Zweck der Anwendung

Egal ob auf der Bühne oder daheim: Für viele Musiker ist es oft nicht möglich und auch nicht praktikabel, sich die Texte und Akkorde ihres gesamten Repertoires auswendig zu merken. In der Vergangenheit war es daher üblich, sich sämtliche Texte auszudrucken und in sperrigen Leitz-Ordnern abzuheften. Doch wie findet man schnell einen Text, wenn man ihn sucht? Oder wie stellt man sich ein Programm zusammen, das beim nächsten Auftritt vorgetragen werden soll? Egal, wie man es anstellt. Es ist immer mit viel Arbeit und mit viel Papier verbunden. In neuerer Zeit werden daher immer öfters Tablet Computer und Smartphones genutzt, die aber ihre ganz eigenen Probleme mit sich bringen. Denn einerseits lassen sich PDF-Dokumente und andere Dateiformate auf diesen Geräten oft nur umständlich bedienen und andererseits ist auch die Erstellung und Organisation der Dokumente sowie der Übertrag auf das Endgerät immer sehr zeitaufwändig.

Zur Lösung des Problems soll daher eine Webanwendung geschrieben werden, die auf allen in Frage kommenden Geräten im Browser aufgerufen werden kann und die eine einfache Erstellung und Verwaltung von Songtexten bietet. Die App soll dabei eine übersichtliche Liste mit allen vorhandenen Texten bieten, in der ein gewünschter Text schnell gesucht und mit einem Klick aufgerufen werden kann. Zur Erstellung der Texte soll es dabei möglich sein, im Internet nach vorhandenen Songtexten zu suchen, um diese direkt zu übernehmen. Die App bietet sich somit für alle Musiker an, die anhand eines Lead Sheets[1] neue Lieder üben oder diese auf der Bühne vortragen wollen.

Abb. 1: Notenanzeige auf einem Tablet

Sweet Sixteen – Billy Idol

```
[Strophe 1]
am7
I'll do anything for my sweet sixteen

And I'll do anything for my little runaway child
                F                          am7
Gave my heart an engagement ring, she took everything
          G                          am7
Everything I gave her, oh my sweet sixteen
              F
Built a moon for a rocking chair
                    G
I never guessed it would rock her for from here
oh, oh, oh, oh, oh

F                G                    am2
Someone's build a candy castle for my sweet sixteen
F                G               am2
Someone's built a candy brain and filled it in
```

Abb. 2: Anzeige eines Songtexts als Lead Sheet

1.2 In Frage kommende Benutzergruppen

„My Songbook“ ist als Stand-Alone-Anwendung für Einzelanwender konzipiert. Es gibt daher nur eine Anwendgruppe, bei der es sich um Musiker mit geringem bis mittleren Technikwissen handelt. Die Anwendung muss somit auch für technikunerfahrene Benutzer leicht zu bedienen sein. Es kann jedoch davon ausgegangen werden, dass die in Frage kommenden Anwender genügend musikalisches Grundwissen besitzen, um den Aufbau von Lead Sheets zu verstehen und daher wissen, wie ein solcher Songtext zu lesen ist.

1 Ein Lead Sheet besteht nur aus dem Songtext mit Akkorden ohne wirkliche Noten.

1.3 Problemstellung aus Sicht der Benutzer

Folgende Anforderungen muss die Anwendung aus Sicht der Benutzer erfüllen:

- Beim Start der Anwendung soll eine Liste mit allen vorhandenen Songtexten erscheinen.
- Die Liste soll entweder nach Songtitel oder nach Interpret alphabetisch sortiert werden können.
- Innerhalb der Liste soll anhand Songtitel oder Interpret gesucht werden können.
- Ein vorhandener Songtext soll mit einem Klick einfach aufgerufen werden können.
- Neue Songtexte sollen möglichst einfach hinzugefügt werden können.
- Hierfür soll es möglich sein, Songtexte im Internet zu suchen und direkt zu übernehmen.
- Die übernommenen Songtexte sollen nachträglich bearbeitet werden können.
- Wenn ein Text nicht im Internet gefunden wird, soll es möglich sein, ihn selbst einzugeben.
- Ein bereits vorhandener Text soll gelöscht werden können, wenn er nicht mehr benötigt wird.
- Die Anwendung soll auch offline funktionieren, wenn keine Internetverbindung besteht.

In der ersten Version müssen folgende Funktionen noch nicht möglich sein:

- Es muss nicht möglich sein, mehrere Listen mit Songtexten zu verwalten.
- Es muss nicht möglich sein, die vorhandenen Texte herunterzuladen oder zu exportieren.
- Es muss nicht möglich sein, die Songtexte zwischen mehreren Endgeräten zu teilen.
- Es muss nicht möglich sein, Audiodateien mit den Songtexten zu verknüpfen.
- Es muss nicht möglich sein, Seiten wie Youtube oder Soundcloud zu integrieren.
- Es muss keine Benutzerverwaltung oder serverseitige Authentifikation vorhanden sein.
- Es muss nicht möglich sein, Songtexte öffentlich zugänglich zu machen oder zu teilen.
- Es muss nicht möglich sein, Songtexte zu kommentieren oder zu bewerten.
- Es muss nicht möglich sein, die Anwendung außerhalb des Browsers als Native App zu nutzen.

Diese Funktionen können in späteren Ausbaustufen zur Verfügung gestellt werden, wofür dann aber eigene Fachkonzepte benötigt werden.

2 Benutzeroberfläche und UI-Skizzen

2.1 Allgemeine Gestaltung der Benutzeroberfläche

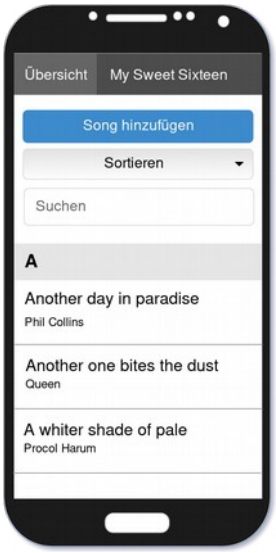

Abb. 3: Mobile Darstellung

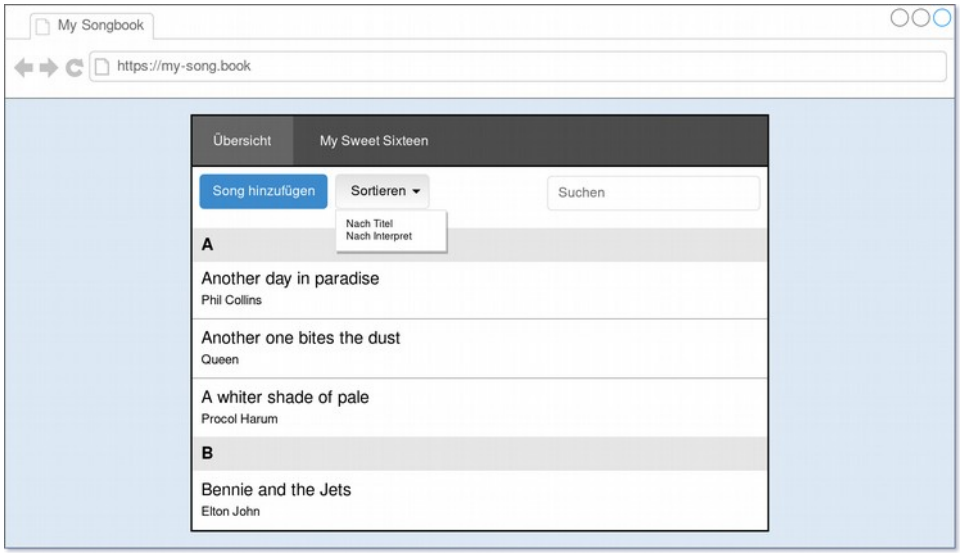

Abb. 4: Darstellung am Computer

Darstellung auf unterschiedlichen Endgeräten

- Die Anwendung soll nach dem „Mobile First“-Prinzip entwickelt werden, so dass sie vom Smartphone bis zum großen Bildschirm auf allen Endgeräten genutzt werden kann.
- Auf kleinen und mittleren Bildschirmen soll die Anwendung den gesamten Bildschirm füllen.
- Auf großen Bildschirmen soll die Anwendung vor einem Hintergrundbild mittig dargestellt werden.
- Für jede Geräteklasse soll eine eigene Schriftgröße definiert werden, so dass die Schriftgröße mit der Größe des Bildschirms zunimmt.
- Die Anwendung soll druckfreundlich gestaltet werden, so dass jede Seite ausgedruckt werden kann. Dabei sind alle für den Ausdruck überflüssigen Sachen wie Hintergrundbilder, Seitenrahmen oder Bedienelemente auszublenden. Der Ausdruck darf nur die reinen Daten enthalten, so als hätte man mit Word oder LibreOffice ein entsprechendes Textdokument aufgesetzt.

Allgemeine Gestaltungsrichtlinien

- Am oberen Bildschirmrand soll es möglich sein, zwischen der Übersicht und dem zuletzt geöffneten Songtext umzuschalten.
- Generell nutzt die Anwendung ein flaches Design mit wenig Farben.
- Der Hintergrund des Hauptbereichs ist weiß, die Schriftfarbe ist schwarz.
- Hellgrau dient als Akzentfarbe für Hervorhebungen und Trennlinien.
- Anklickbare Buttons werden durch eine matt-leuchtende Hintergrundfarbe mit weißer Schrift betont.

2.2 Liste der vorhandenen Songtexte

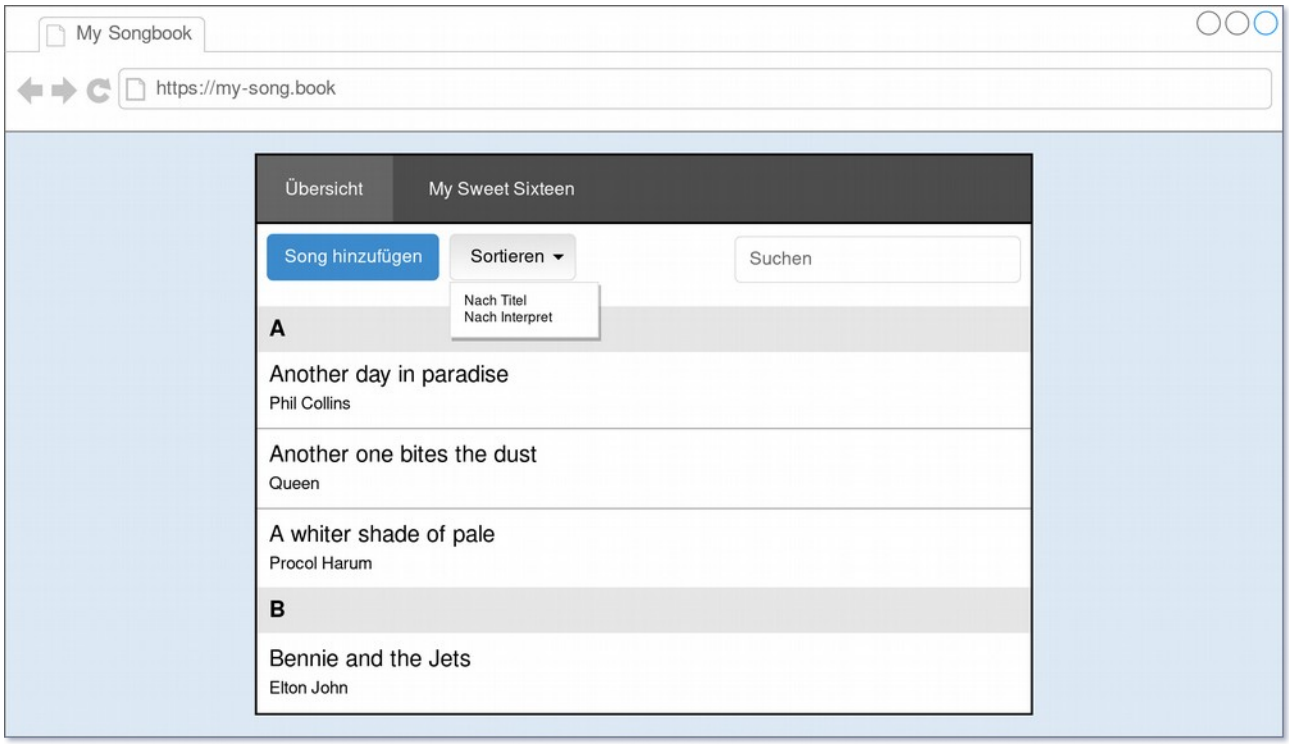

Abb. 5: Übersichtsseite mit den vorhandenen Songtexten

2.3 Anzeige eines vorhandenen Songtexts

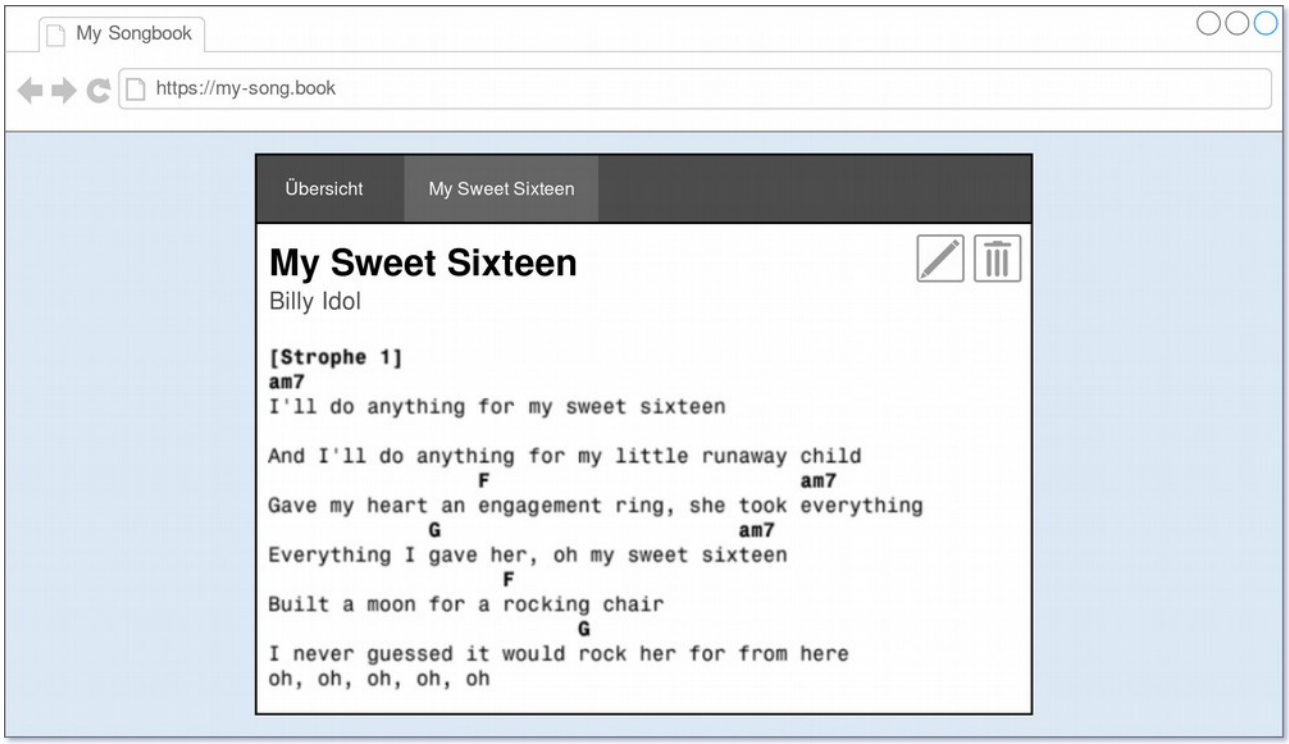

Abb. 6: Detailseite eines Songtexts im Anzeigemodus

2.4 Nachbearbeiten eines Songtexts

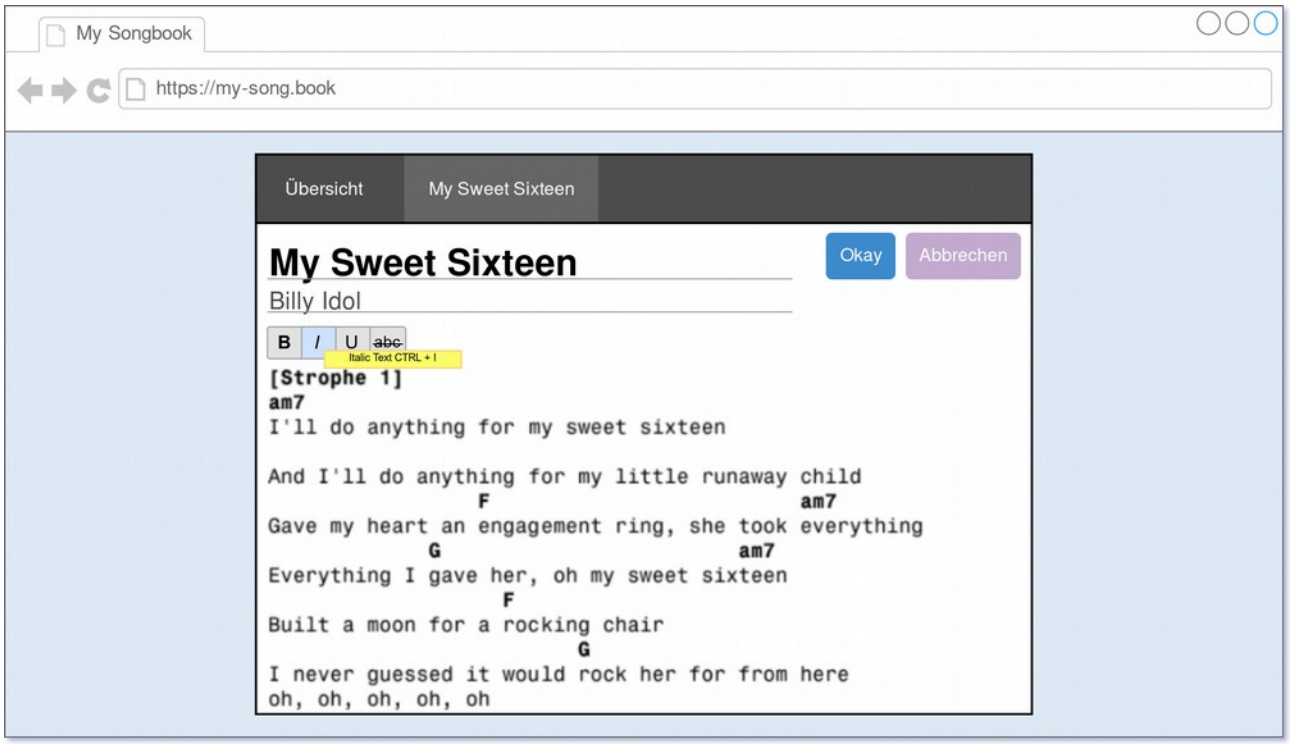

Abb. 7: Detailseite eines Songtexts im Änderungsmodus

2.5 Anlage eines neuen Songtexts

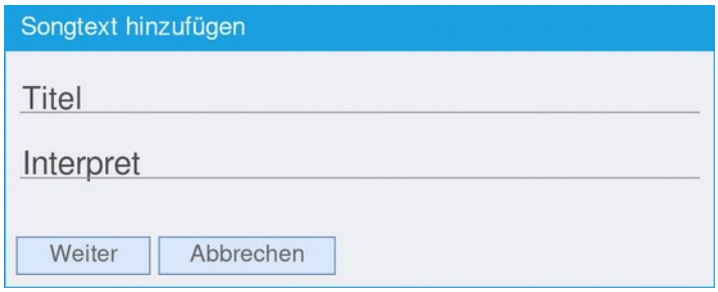

Abb. 8: Popup-Fenster zur Anlage eines neuen Songtexts

2.6 Löschen eines Songtexts

Abb. 9: Popup-Fenster zum Löschen eines Songtexts

3 Funktionale Anforderungen

Die nachfolgenden Teilkapitel enthalten eine vollständige Liste mit den funktionalen Anforderungen an die Anwendung. Sie konkretisieren die Problembeschreibung und die UI-Mockups um weitere Details zum Laufzeitverhalten der Anwendung.

3.1 Liste der vorhandenen Songtexte

- Beim Start der Anwendung soll eine Liste mit allen vorhandenen Songtexten erscheinen.
- Gibt es noch keinen Text, soll stattdessen die Meldung „Noch keine Texte vorhanden“ erscheinen.
- Die Liste soll zu jedem Text den Titel und den Interpreten anzeigen.
- Standardmäßig soll die Liste nach Titel alphabetisch sortiert sein.
- Es soll jedoch auch eine Sortierung nach Interpret möglich sein.
- Bei einer Sortierung nach Titel soll je Anfangsbuchstabe eine Zwischenüberschrift eingefügt werden.
- Bei einer Sortierung nach Interpret soll je Interpret eine Zwischenüberschrift eingefügt werden.
- Oberhalb der Liste soll sich ein Suchfeld befinden, mit dem nach vorhandenen Text gesucht werden kann. Die Liste soll dabei in Echtzeit aktualisiert werden, so dass sie nur Songtexte zeigt, deren Titel oder Interpret den gesuchten Begriff enthalten.
- Löscht der Anwender die Eingabe aus dem Suchfeld, sollen wieder alle Titel angezeigt werden.
- Ein Songtext soll durch Anklicken des Titels oder des Interpreten angezeigt werden können.
- Zusätzlich soll es oberhalb der Liste einen Button geben, um einen neuen Text anzulegen.
- Innerhalb der Liste soll zu jedem Songtext ein Button angezeigt werden, um den Text zu löschen.
- Das Löschen muss jedoch durch eine Sicherheitsabfrage bestätigt werden.

3.2 Anzeige eines vorhandenen Songtexts

- Auf der Detailseite eines Textes werden sein Name, der Interpret sowie der Text selbst angezeigt.
- Namen und Interpret sollen deutlich zu erkennen sein, aber nicht zu viel Platz beanspruchen.
- In derselben Zeile wie der Titel soll ein Button sichtbar sein, um den Text zu bearbeiten.
- Außerdem soll der Anwender von hier wieder zurück auf die Übersichtsseite wechseln können.

3.3 Nachbearbeiten eines Songtexts

- Im Änderungsmodus soll der Anwender den Titel und den Interpreten des Textes ändern können.
- Zusätzlich soll der Text selbst in einem WYSIWYG-Editor erscheinen, indem der Anwender beliebige Änderungen vornehmen kann.
- Der Editor soll folgende Formatierungsmöglichkeiten bieten:
  - Auswahl der Schriftart
  - Änderung der Schriftgröße
  - Fett
  - Kursiv
  - Unterstrichen
  - Schriftfarbe
  - Hintergrundfarbe
- Weitere Formatierungsoptionen können unterstützt werden, sind aber kein Muss.

- Über zwei Buttons soll der Anwender entscheiden können, ob er die Änderungen sichern oder verwerfen will.
- In beiden Fällen soll der Anwender wieder in den Anzeigemodus zurückgeworfen werden.

3.4 Anlage eines neuen Songtexts

- Der Button zur Anlage eines neuen Textes soll nur in der Übersichtsseite erscheinen.
- Klickt der Anwender auf den Button, soll sich ein modaler Dialog mit folgenden Elementen öffnen:
  - Eingabefeld für den Songtitel
  - Eingabefeld für den Interpreten
  - Button „Weiter“
  - Button „Abbrechen“
- Klickt der Anwender auf „Abbrechen“ soll der Dialog geschlossen werden und wieder die Übersichtsseite sichtbar sein.
- Klickt der Anwender auf „Weiter“ soll über eine entsprechende API online nach dem Text gesucht werden.
- Egal, ob der Text gefunden wird oder nicht, der Anwender landet in jedem Fall auf der Detailseite des neuen Textes im Änderungsmodus.
- Songtitel und Interpret des neuen Textes sollen anhand der bei der Suche eingegebenen Werte immer vorbelegt sein.
- Im WYSIWIG-Editor soll der gesuchte Text erscheinen, sofern er gefunden wurde.
- Wurde der Text nicht gefunden, soll der Editor leer bleiben und stattdessen eine Meldung mit folgendem Text ausgegeben werden: „Der gesuchte Text wurde nicht gefunden.“
- Über zwei Buttons soll der Anwender entscheiden können, ob er den Text speichern oder verwerfen will (vgl. dieselbe Anforderung beim Nacharbeiten eines Textes).
- Entscheidet sich der Anwender jedoch, den Text zu verwerfen, soll er wieder in der Übersichtsseite landen, da der Text ja nicht gespeichert wurde.

4 Technologieauswahl

4.1 Art der Webanwendung

Da in der Anwendung keine Benutzerverwaltung vorgesehen ist, sie ähnlich wie eine Native App nur von einem Anwender lokal genutzt werden soll und sie auch offline lauffähig sein muss, soll die Anwendung als reine Browser App realisiert werden. Die Aufgabe des Servers besteht lediglich darin, die für die Ausführung der Anwendung notwendigen Dateien bereitzustellen.

Dadurch entfällt die Komplexität, neben dem Frontend noch einen Serverteil entwickeln zu müssen. Umgekehrt müssen dadurch aber sämtliche Funktionen, die sonst üblicherweise durch den Server erbracht werden, clientseitig nachgebildet werden:

- Bereitstellung einer einheitlichen Seitenstruktur mit gleichem Aussehen für alle Seiten
- Unterstützung des Zurück-Buttons im Browser
- Speichern der Songliste und der darin enthaltenen Songtexte

Dennoch kann die Anwendung in einer späteren Ausbaustufe um einen Serverteil zu einer hybriden Anwendung erweitert werden. Der Server würde in diesem Fall einfach verschiedene API-Methoden zur Verfügung stellen, die aus der Browser App heraus aufgerufen werden können. Dies soll in der ersten Version der App jedoch noch nicht berücksichtigt werden. Im ersten Wurf genügt es, wenn die App ausschließlich lokal im Browser läuft.

Ein weiterer Vorteil ist, dass die Anwendung später mit relativ wenig Aufwand zu einer Native App für unterschiedliche Betriebssysteme verpackt werden kann. Frameworks wie Apache Cordova (bzw. Phonegap) oder Electron können hierfür den notwendigen Rahmen zur Verfügung stellen. Dies soll in der ersten Version der Anwendung jedoch ebenfalls noch nicht berücksichtigt werden.

4.2 Serverseitige Technologien

Der einzige Serverdienst, der innerhalb der Anwendung zum Einsatz kommt, ist die Suche nach Songtexten im Internet. Hierzu wird die unter der Adresse https://lyric-api.herokuapp.com/api/ aufrufbare „Lyric API“ verwendet. Sie ist zwar nicht für die kommerzielle Nutzung freigegeben, da sie lediglich die unter https://lyrics.wikia.com abrufbaren Songtexte zur Verfügung stellt, sie hat jedoch den Vorteil, mit einer simplen HTTP-GET-Anfrage aufgerufen werden zu können. Außerdem liefert sie das Ergebnis im JSON-Format zurück, so dass es sehr einfach mit JavaScript ausgewertet werden kann.

Eine typischer Aufruf besteht aus einer einfachen GET-Anfrage an folgende URL:

https://lyric-api.herokuapp.com/api/find/artist/song

„Artist“ ist dabei der Interpret und „Song“ der Name des gesuchten Songs. Zum Beispiel:

http://lyric-api.herokuapp.com/api/find/John%20Lennon/Imagine

Das Ergebnis ist immer ein JSON-Objekt:

```
{
    "err": "none",
    "lyric": "Textzeilen mit \n getrennt"
}
```

Die API kann daher ohne zusätzliche Bibliotheken direkt aus JavaScript heraus aufgerufen werden. Durch das im npm-Repository enthaltene Paket „lyric-get“ wird die Nutzung jedoch weiter vereinfacht. Dadurch wird es möglich, einen Songtext wie folgt mit JavaScript abzurufen:

```
import lyric from "lyric-get";

lyric.get("John Lennon", "Imagine", (err, res) => {
    if(err) {
        // Fehler
    } else {
        // Kein Fehler
    }
});
```

4.3 Clientseitige Technologien

Für die Frontendentwicklung sollen folgende Eintwicklungswerkzeuge zum Einsatz kommen:

- Atom: Spezieller Texteditor für Webentwickler und Programmierer
- git: Versionsverwaltung zur gemeinsamen Arbeit am Quellcode
- npm: Paketverwaltung zum automatischen Download abhängiger Bibliotheken
- Parcel: Web Application Bundler und Entwicklungsserver

Zusätzlich sollen folgende Bibliotheken genutzt werden:

- Navigo: Single Page Router zur Vereinfachung der Navigation innerhalb der App
- PouchDB: Clientseitige NoSQL-Datenbank zum Speichern der Songtexte
- lyric-get: Kleine Bibliothek zur Suche von Songtexten im Internet
- Quill: WYSIWYG-Editor zum Nachbearbeiten der Songtexte

Weitere Frameworks und Bibliotheken sind zu diesem Zeitpunkt nicht vorhergesehen. Insbesondere das Layout der Webseite soll mit einfachem HTML und CSS ohne weitere Frameworks erstellt werden. Zwar kann Parcel externe Frameworks wie Bootstrap nur auf die wirklich benötigten Funktionen reduzieren, aufgrund der einfachen Layoutanforderungen wird ein Framework jedoch nicht benötigt.

Parcel benötigt an dieser Stelle eine Erklärung: Im Grunde genommen könnte die Webanwendung auch ohne dieses Werkzeug entwickelt werden. Parcel vereint jedoch mehrere allgemein nützliche Funktionen, die die Entwicklung an einzelnen Stellen etwas vereinfachen:

- Der JavaScript-Code kann in sauber gekapselte Module zerlegt werden. Jede Quellcodedatei entspricht dabei einem Modul, das ganz gezielt einzelne Elemente exportiert, damit diese in anderen Modulen importiert werden können. Inhalte, die nicht exportiert werden, stehen für andere Module nicht zur Verfügung. Reines JavaScript im Browser (ohne ein Bundler-Werkzeug) unterstützt dies nicht, da hier aus Kompatibilitätsgründen alle Dateien alle Inhalte teilen.
- Sämtliche Codedateien und sämtliche Bilder werden komprimiert und verdichtet, um wertvolle Bandbreite beim Laden der Anwendung zu sparen.
- Aus sämtlichen externen Bibliotheken wird (sofern möglich) nur der notwendige Code übernommen, der innerhalb der Anwendung auch tatsächlich verwendet wird.[2] Dadurch kann gerade bei großen Bibliotheken die Größe der Anwendung nochmals sehr reduziert werden.

2 Dieses Feature nennt sich „Tree Shaking“ oder auch „Pruning“.

- CSS-Prozessoren wie LESS oder SASS werden out of the box unterstüzt, wodurch die Syntax für Stylesheets deutlich erweitert wird (im Projekt jedoch nicht genutzt).
- Es ist bereits ein Entwicklungsserver enthalten, mit dem die Anwendung unter Echtbedingungen getestet werden kann, ohne einen eigenen Webserver hierfür aufsetzen zu müssen.
- Der Entwicklungsserver unterstützt „Hot Reloading“, wodurch die Seite im Browser nach einer Quellcodeänderung automatisch neu geladen wird.

In anderen Projekten wird für all diese Aufgaben häufig Webpack genutzt, welches den Vorteil besitzt umfangreich konfiguriert werden zu können. Webpack kann daher ganz genau an die spezifischen Anforderungen eines Projekts angepasst werden, wohingegen Parcel durchaus subjektive Voreinstellungen bietet, die nicht änderbar sind. Dadurch kommt Parcel jedoch komplett ohne Konfiguration aus und kann im Vergleich zu Webpack sehr einfach eingesetzt werden.

PouchDB ermöglicht es, JSON-Dokumente im lokalen IndexDB-Speicher des Browsers abzuspeichern, ohne hierfür die überaus komplizierte Schnittstelle von IndexDB verwenden zu müssen. Der Funktionsumfang und die API orientieren sich dabei an dem CouchDB-Datenbankserver, so dass PouchDB (mit P statt C) im Browser nahezu die gleichen Features zur Verfügung stellt. Außerdem kann die Anwendung dadurch später sehr einfach um eine serverseitige Datenhaltung erweitert werden, da PouchDB eine automatische Synchronisierung mit einem CouchDB-Server unterstützt. Diese Möglichkeit wird in der ersten Version der Anwendung zwar noch nicht genutzt, jedoch ist es für spätere Ausbaustufen sehr nützlich, dass hierfür nur geringfügige Anpassungen am Code vorgenommen werden müssen.