Dokumentation

1. Einleitung **2**

2. Set-Ups **3**

2.1 Set-Up von teleTunes 3

2.2 Set-Up von Docker 3

3. WebApp-Struktur **5**

3.1 Docker-Files 5

3.2 Server-Files 6

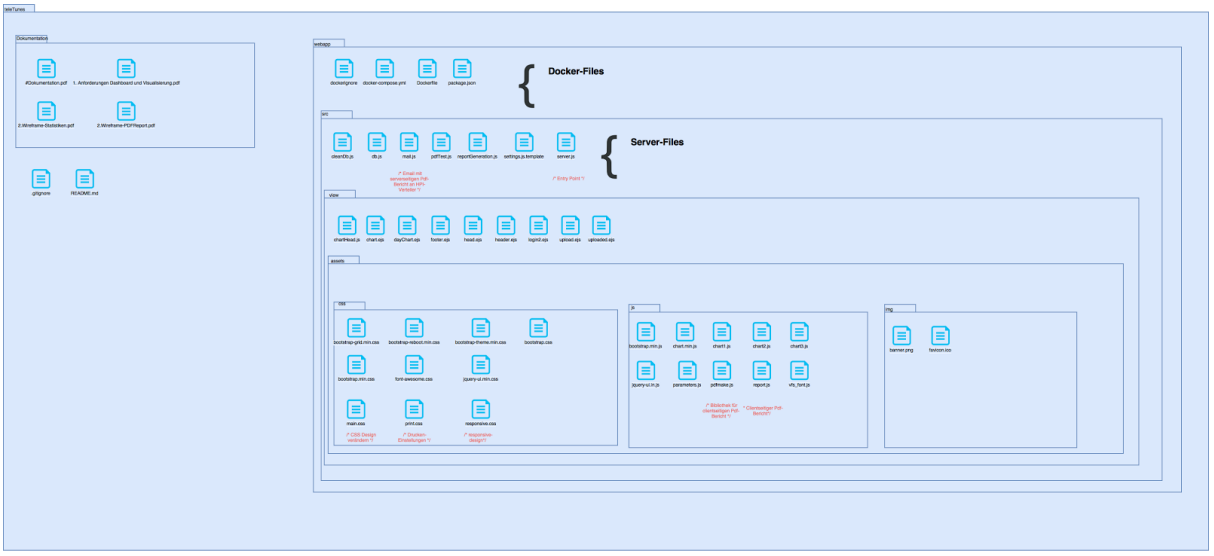
3.3 Front-End 7

3.3.1 assets Ordner 7

3.3.2 Templates (.ejs) 8

1. Einleitung

Dies ist die Dokumentation für die Web-Applikation von teleTunes. Die Applikation nimmt die Daten vom HPI-Podcast, welche ITunes als .tsv liefert und visualisiert diese. Anschließend kann ein Pdf-Bericht gedownloaded werden. Zusätzlich kann ein PDF-Report per E-Mail abonniert werden. Im Folgenden wird erklärt, wie die Applikation aufgebaut ist.



Im Ordner **Dokumentation** befindet sich die #Dokumentation.pdf der Applikation.
Außerdem befinden sich in dem Ordner der 0.Kick-Off-Vortrag.pdf, in dem die verwendeten Technologien näher erläutert werden. Die Anforderungen des Projektes werden bei 1.Anforderungen Dashboard und Visualisierungen.pdf erläutert.
Zudem befinden sich noch die Wireframe für die Visualisierung im Ordner 2. WireFrame-Statistiken.pdf und 3.WireFrame-PDFReport.pdf.

Im Ordner **webapp** befindet sich die geschriebene Applikation. Diese wird im späteren Verlauf erklärt.

Die Applikation kann vom folgenden Repository gedownloadet werden:
https://github.com/philipphoberg/teleTunes.git
Darüber hinaus findet sich im Folgenden eine Anleitung zur Installation von Docker, die gleichzeitig im Schritt 2.2. dieser Dokumentation erläutert wird.

2. Set-Ups

Im Nachfolgenden wird erklärt, wie die das Setup für teleTunes funktioniert.

2.1 Set-Up von teleTunes

Um teleTunes zu starten, wird das Set-Up von Docker benötigt (hier erklärt im Schritt 2.2)
https://github.com/philipphoberg/teleTunes/wiki/Setup-Dev-Env

kopiere `settings.js.template` zu `settings.js` und gebe die Werte ein

Run
`docker-compose up teletunes`

Datenbank leeren
`docker-compose up`
`docker-compose exec teletunes nodejs src/cleanDb.`
`docker-compose restart teletunes`

2.2 Set-Up von Docker

Für die Virtualisierung haben wir uns für die Technologie Docker entschieden. Hier werden zur Isolierung der Anwendung Container verwendet.

Hier finden Sie das Docker-Setup und weitere Installations- und Einrichtungshinweise:
https://github.com/philipphoberg/teleTunes/wiki/Setup-Dev-Env

Docker:
Auf der Webseite von Docker befindet sich eine Einleitung zur Installation.
https://www.docker.com/community-edition

Docker-Compose installieren
https://docs.docker.com/compose/install/#prerequisites

App installieren
Zuerst wird das git-repo geclont

```
git clone git@github.com:philipphoberg/teleTunes.git
cd teleTunes/webapp
```

Ab jetzt braucht ihr die ganze Zeit r**oot-Rechte** vergesst also das `sudo` nicht und achtet darauf, dass ihr nach wie vor im `webapp` Verzeichnis seid.

Dann erstellt ein Docker image (wie ein eigenes OS-Image): Dabei werden alle dependencies aus der `package.json` gelesen. Wenn ihr die verändert müsst ihr diesen Schritt also wiederholen.
`docker-compose build teletunes`

und anschließend könnt ihr die app starten mit:
`docker-compose up teletunes`

Wenn ihr wollt dass sie im Hintergrund startet könnt ihr auch die `-d` Option ergänzen:
`docker-compose up -d teletunes`

Die Webapp könnt ihr nun über http://localhost:8080 aufrufen.
Außerdem wurde automatisch eine mysql Datenbank mit erstellt.

Falls ihr eine Shell auf dem nodejs server wollt, nutzt:
`docker-compose exec teletunes /bin/bash`

Nachdem ihr eine shell habt könnt ihr eine mysql shell starten: (Passwort = myTeletunesPw)
`mysql --host=db -u root -p`

PhpMyAdmin
Damit uns die Arbeit mit der Datenbank leichter fällt habe ich außerdem eine GUI mit installiert. Ihr startet sie mit:
`docker-compose up -d teletunesdbadmin`

Dann könnt ihr euch auch http://localhost:8081 mit `root` und my `TeletunesPw` einloggen

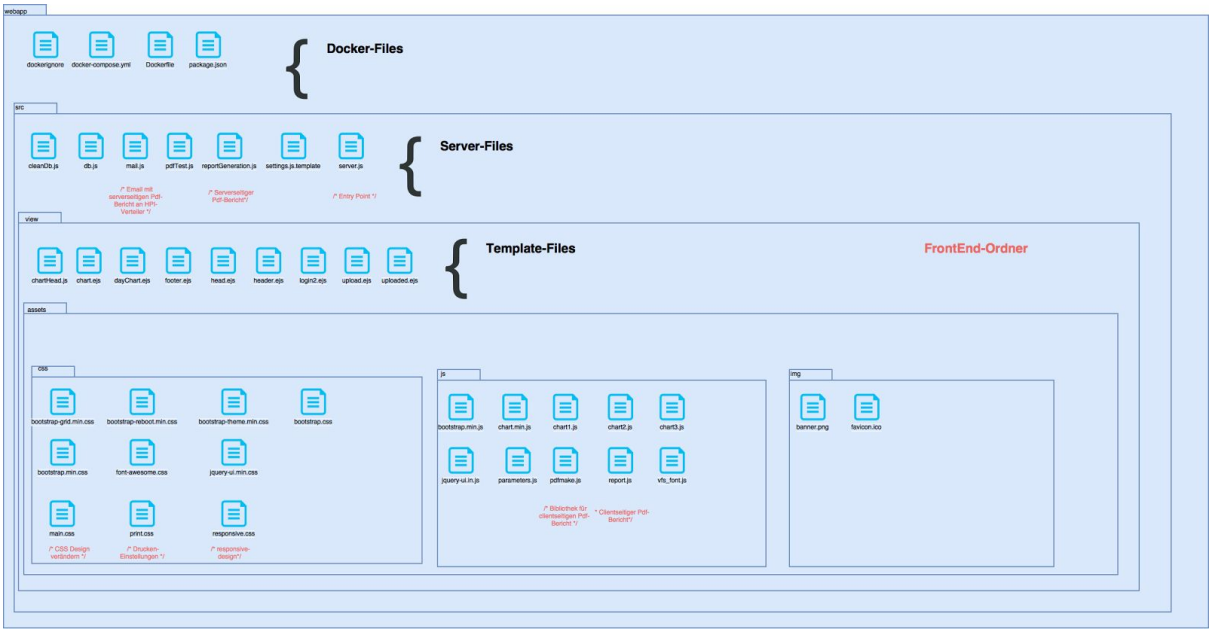
Arbeiten
um an der webapp zu arbeiten beginnt in der `src/server.js`

Bei nodejs muss man den Server nach jeder Änderung neu starten. Das macht ihr mit:
`docker-compose restart teletunes`

Herunterfahren
Ihr könnt das ganze mit `docker-compose stop` herunterfahren.

3. WebApp-Struktur

Im folgenden wird die WebApp-Struktur erklärt.



3.1 Docker-Files

dockerignore

- ignoriert bestimmte Docker-Files.

docker-compose.yml

- Hier werden die services definiert

package.json

- Einbinden von npm-Modulen
- Die Pakete werden beim Bauen des Containers (docker-compose build) installiert

DockerFile

- Hier wird beschrieben wie der Docker container gebaut werden soll
- verwendet von docker-compose build

Folgende npm-Packages sind in der **package.json** eingebunden:

csv-parse

- Parsen von .csv-Dateien

express.js

- Webframework für node.js

nodemailer

- E-Mail Versand mit node.js

request
- HTTP-Requests

mocha
- Javascript Test Framework

fs
- FileSystem

mysql
- MySql-Datenbank

http-auth
- HTTP-Authentifizierung

ejs
- Embedded Javascript Templates

express-session
- speichert Session Data

express-fileupload
- Datenuploads

pdfmake
- Clientseitige Bibliothek zur Pdf-Report-Erstellung

phantom-js-prebuilt
- Serverseitiges Erstellen des Pdf-reports

node-cron
- Aufgaben planen mit Cron-Jobs

3.2 Server-Files

server.js
- Entry Point

settings.js
- hier werden alle relevanten Einstellungen vorgenommen

db.js
- Allgemeiner Datenbankzugriff

cleanDB.js
- Datenbank bereinigen

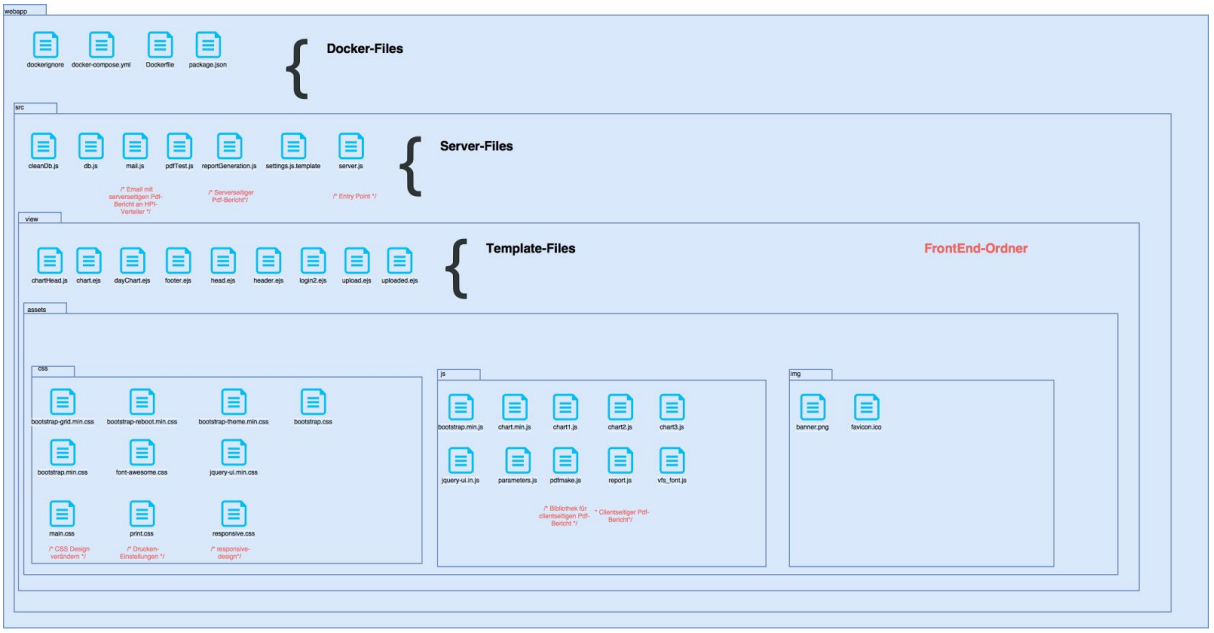
reportGeneration.js
- generiert Serverseitigen Pdf-Bericht mit Phantom.js

pdfGenerator.js
- script für phantomjs zum erstellen des PDFs

mail.js
- serverseitiger Bericht wird an den HPI-Mailer gesendet

3.3 Front-End

Im Ordner **view** befinden sich die ganzen Dateien für das Front-End.
Hier befinden sich die Template(.ejs)-Dateien, der assets-Ordner mit den eingebundenen CSS und Javascript Dateien sowie der **img-Ordner** mit den Bildern.



3.3.1 assets Ordner

Im assets Ordner befinden sich die einzubindenden CSS und Javascript-Dateien, jeweils untergliedert in die Ordner CSS und JS. Hier werden die Bibliotheken eingebunden.
Für das Front-End werden folgende Bibliotheken verwendet

Bootstrap als Front-End-CSS Bibliothek,
FontAwesome für Icons,
Jquery als Javascript-Bibliothek,
chart.js zur Visualisierung der Charts,
pdfmake.js Clientseitig einen PDF-Bericht zu erstellen,
HTML2Canvas.js um die die Charts in den Clientseitigen PDF-Bericht zu speichern

JS:

- chart1.js, chart2.js, chart3.js
- report.js
- parameter.js

Änderungen an den Charts 1-3 werden bei chart1.js, chart2.js und chart3.js vorgenommen. Parameteränderungen nimmt man in der parameter.js vor.

Um Änderungen an dem Clientseitigen Pdf-Bericht vorzunehmen, tut man diese in der Datei report.js

CSS:

- main.css
- print.css
- responsive.css

Um das CSS-Design zu ändern, verändert man dieses in der main.css .
Das Design responsive zu machen wird in der responsive.css gemacht.
Das verändern des Designs für das Ausdrucken einer seite wird in der print.css geändert.

3.3.2 Templates (.ejs)

Im folgenden wird erklärt, welche Dateien in den Template Dateien eingebunden werden:

ChartHead.ejs (Einbinden der Bibliotheken für die Charts)

- HTML2Canvas
- pdfmake.js
- report.js
- chart.js

chart.ejs (Visualisierung der Charts)

dayChart.ejs (Einbinden des chartHeads)

footer.ejs (Template für den Footer)

head.ejs (Bindet)

- main.css
- responsive.css
- JQuery
- bootstrap
- FontAwesome

header.ejs (HTML-Code, um den Header anzuzeigen)

login2.ejs (User kann sich hier einloggen**)**

upload.ejs (Upload der .csv Datei**)**

uploaded.ejs (.csv-Datei hochgeladen**)**