**Technische Dokumentation**

Autor: Luis Hutterli

Datum: 10.01.2025

Version: 0.1

# Analyse

## Projektbeschrieb

Es soll eine Web-App entwickelt werden, mit welcher man eine elektronische JukeBox mit NFT-Songs (Non-Fungible Tokens) benutzen und verwalten kann.

## Anforderungen

Ein jukeStack Benutzer (z.B. Steven Hatch) soll sich mit seiner E-Mail Adresse (Steven.Hatch@gmx.net) und einem Passwort (z.B. djfhU78&wR) in der jukeStack-App registrieren können.

Sobald er sich registriert hat, kann er sich mittels E-Mail und Passwort in der jukeStack-App anmelden. Gelingt ihm dies, so sieht er einerseits eine Liste mit NFT-Songs (z.B. Ian Kelly, Speak your mind, 4:08, 2008), die er ausleihen kann und eine zweite Liste mit bereits ausgeliehenen Songs, die aber im Moment noch leer ist.

Nun kann er aus der ersten Liste einen Titel anwählen (z.B. Speak your mind) und ihn ausleihen. Sind die unten angeführten Bedingungen erfüllt, so wird die Ausleihe gewährt und der Song auf die Liste der ausgeliehenen Songs gesetzt. Die Liste der ausgeliehenen Songs enthält nun diesen Song, welcher um das Ausleihdatum ergänzt wurde (z.B. Ian Kelly, Speak your mind, 4:08, 2008, 11.12.2024 14:03) und kann nun abgespielt werden.

Die Bedingungen für eine Ausleihe lauten wie folgt:

- Ein Benutzer darf maximal nur 5 Songs gleichzeitig ausleihen.

- Ein Song kann, da jeder Song in dieser App ein NFT ist, gleichzeitig maximal nur einmal ausgeliehen sein.

- Ein Song kann für maximal n Tage ausgeliehen werden.

Sind die ersten zwei Bedingungen nicht erfüllt, so soll eine entsprechende Warnung ausgegeben werden.

Ist ein Song ausgeliehen, kann er so oft angehört werden, wie der Benutzer es möchte.

Ein ausgeliehener Song, kann, wenn er nicht mehr gewünscht wird, wieder zurückgegeben werden.

Zudem gibt es noch einen Administrator:in (z.B. Liza Suller, Liza.Suller@tokentec.ch, kljht564xc). Logt die sich ein, sieht sie statt der Songs, eine Liste mit allen Benutzern. Wählt sie einen Benutzer an, sieht sie wiederum dessen Ausleihen und kann diese auch rückgängig machen. Zudem kann der Admin die maximale Ausleihdauer n (siehe Bedingung 3) festlegen (default = 1).

Die Passwörter müssen unbedingt gehasht und gesalzen (kein MD5!) in der DB abgelegt werden.

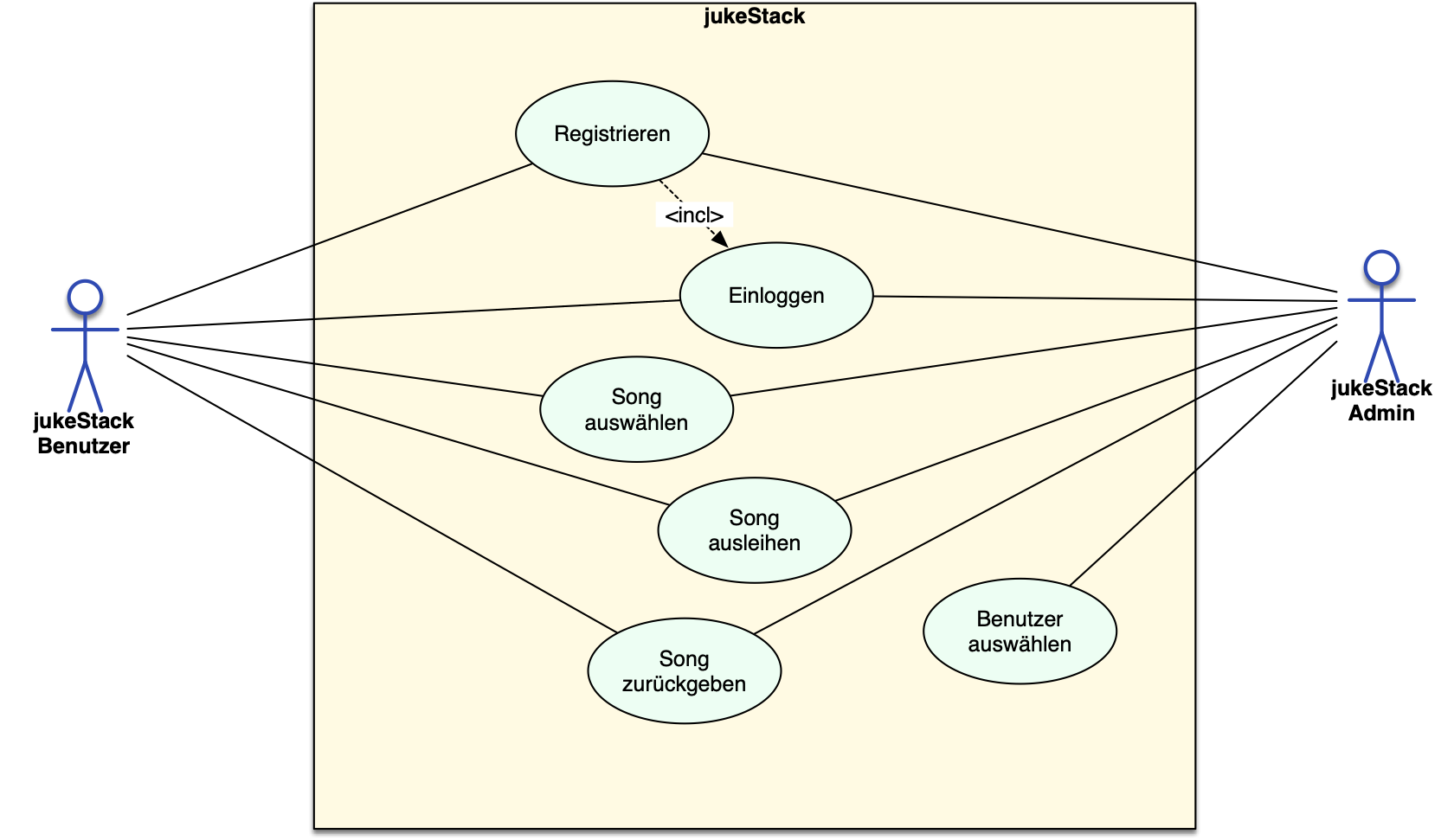
## Realisierungstechnologie

Die Web-App wird folgende Technologien:

Front-End: React, (UI und weitere Util Library)

Back-End: MySQL, Java Vert.X Server laufend auf Google Cloud Run, Cloudflare R2 Object Storage für die Songdateien

## Use-Cases

Aus dem folgenden Diagramm sind die gewünschten Anwendungsfälle erkennbar.

## ERM

A diagram of a diagram

Description automatically generated

# Design

## Layout

Login/Signup:  
A screenshot of a login screen

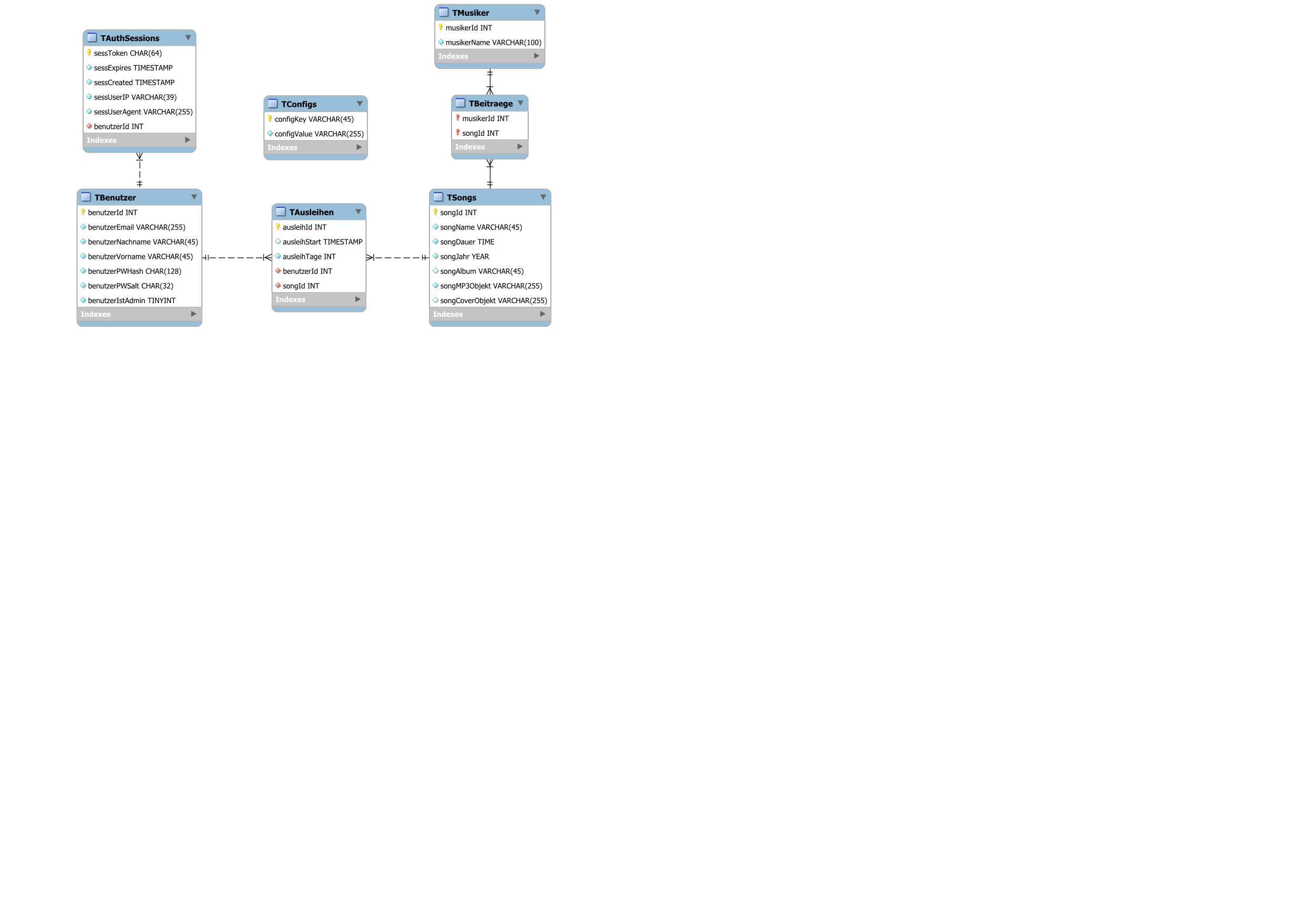
Description automatically generated

Main:

A screenshot of a video

Description automatically generated

## Datenbankschema



## API Endpunkte

Basis: jukestack.ch/api

|  |  |  |  |
| --- | --- | --- | --- |
| **Route** | **Method** | **Description** | **Status** |
| /user | POST | Create user | Fully Implemented |
| /user | GET | Get user info | Fully Implemented |
| /user | PUT | Update user info | Not Implemented |
| /user | DELETE | Delete user | Not Implemented |
| /auth/login | POST | Login | Fully Implemented |
| /auth/logout | POST | Logout | Not Implemented |
| /auth/verify | POST | Check Token validility | POC / MVP |
| /auth/refresh | POST | Refresh Token | Fully Implemented |
| /songs | GET | List all songs | Fully Implemented |
| /lend | GET | List all borrowed songs | Fully Implemented |
| /lend/{id} | POST | Lend a song | Fully Implemented |
| /lend/{id} | DELETE | Return a song | Fully Implemented |
| /lend/{id}/listen | GET | Generate a link to song file | Fully Implemented |
| /admin/users | GET | List all users | Fully Implemented |
| /admin/users/{id}/lend | GET | List a users borrowed songs | Fully Implemented |
| /admin/lend/{id} | PUT | Update a users lend by lendId | Fully Implemented |
| /admin/lend/{id} | DELETE | Return a users lend by lendId | Fully Implemented |

|  |  |  |  |
| --- | --- | --- | --- |
| **Funktionale Testfälle:** | | | |
| **Test** | **Erwartetes Ergebnis** | **Bestanden** | **Bemerkung** |
| Benutzer erstellen | Der Benutzer wird erfolgreich in der Datenbank gespeichert. |  |  |
| Anmelden | Ein Session-Cookie wird erstellt, und der Benutzer wird eingeloggt. |  |  |
| Songs anzeigen | Alle verfügbaren Songs werden korrekt angezeigt. |  |  |
| Ausgeliehene Songs anzeigen | Alle ausgeliehenen Songs werden korrekt angezeigt. |  |  |
| Song ausleihen | Der Song wird in die persönliche Bibliothek hinzugefügt. |  |  |
| Song zurückgeben | Der Song wird aus der persönlichen Bibliothek entfernt. |  |  |
| Song abspielen | Der Song wird abgespielt, und Ton ist hörbar. |  |  |
| Maximal 5 Songs ausleihen | Eine Fehlermeldung wird angezeigt, und der Song wird nicht hinzugefügt. |  |  |
| Admin - Alle Benutzer anzeigen | Alle Benutzer (außer andere Admins) werden angezeigt. |  |  |
| Admin - Ausleihen eines Benutzers anzeigen | Alle ausgeliehenen Songs des Benutzers werden korrekt angezeigt. |  |  |
| Admin - Ausleihe eines Benutzers zurückgeben | Der Song wird aus der Bibliothek des Benutzers entfernt. |  |  |
| Admin - Ausleihe eines Benutzers ändern | Die neuen Ausleihtage werden korrekt gespeichert. |  |  |

## Testliste

|  |  |  |  |
| --- | --- | --- | --- |
| **Destruktive Testfälle** | | | |
| **Test** | **Erwartetes Ergebnis** | **Bestanden** | **Bemerkung** |
| Benutzer erstellen - Unvollständige Eingaben | Eine Fehlermeldung wird angezeigt, und der Benutzer wird nicht erstellt. |  |  |
| Anmelden - Falsche Zugangsdaten | Eine Fehlermeldung wird angezeigt, und der Benutzer wird nicht eingeloggt. |  |  |
| Admin - Ungültige Änderungen an Ausleihen | Es wird ein Standard Wert verwendet. |  |  |

# Realisation

## Backend

Ich habe mich für ein Java-Backend entschieden, da ich auf der suche nach einer guten Serverless Cloud Platform gesucht habe. Hierzu habe ich diese Webseite verwendet: <https://www.srvrlss.io/>

Ursprünglich wollte ich das Backend in Rust schreiben und auf Cloudflare Workers deployen, jedoch war die Integration mit dritten SQL-Datenbank zu aufwändig und ich musste eine andere Plattform suchen.

Ich habe dann schliesslich Google Cloud Run gefunden und ausgewählt, da es sehr flexibel ist, ich ein wenig Erfahrung mit der Google Cloud Plattform habe und es für meinen Anwendungsfall gratis ist. GCR (Google Cloud Run) kann direkt mit Java verwendet werden, jedoch mit SpringBoot, zum glück kann man jedoch sein eigenes Deployment mit Docker erstellen und jede Sprache und Framework deployen. Ich habe mich also für einen Vert.X Server entschieden, da er nach meiner Recherche als einer der schnellsten und flexibelsten herausging.

## Frontend

Das Frontend ist eine React Webapplikation, statt JavaScript habe ich TypeScrypt verwendet.

Als Packager habe ich die Vite Framework verwendet, da sie das Entwickeln und Deployen einfach etwas schneller macht.

Um das Frontend zu Deployen habe ich Google Firebase Hosting verwendet, da ich Firebase schon sehr oft verwendet habe und gute Erfahrungen damit gemacht habe.

# Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Funktionale Testfälle:** | | | |
| **Test** | **Erwartetes Ergebnis** | **Bestanden** | **Bemerkung** |
| Benutzer erstellen | Der Benutzer wird erfolgreich in der Datenbank gespeichert. | Ja | - |
| Anmelden | Ein Session-Cookie wird erstellt, und der Benutzer wird eingeloggt. | Ja | - |
| Songs anzeigen | Alle verfügbaren Songs werden korrekt angezeigt. | Ja | - |
| Ausgeliehene Songs anzeigen | Alle ausgeliehenen Songs werden korrekt angezeigt. | Ja | - |
| Song ausleihen | Der Song wird in die persönliche Bibliothek hinzugefügt. | Ja | - |
| Song zurückgeben | Der Song wird aus der persönlichen Bibliothek entfernt. | Ja | - |
| Song abspielen | Der Song wird abgespielt, und Ton ist hörbar. | Ja | - |
| Maximal 5 Songs ausleihen | Eine Fehlermeldung wird angezeigt, und der Song wird nicht hinzugefügt. | Ja | - |
| Admin - Alle Benutzer anzeigen | Alle Benutzer (außer andere Admins) werden angezeigt. | Ja | - |
| Admin - Ausleihen eines Benutzers anzeigen | Alle ausgeliehenen Songs des Benutzers werden korrekt angezeigt. | Ja | - |
| Admin - Ausleihe eines Benutzers zurückgeben | Der Song wird aus der Bibliothek des Benutzers entfernt. | Ja | - |
| Admin - Ausleihe eines Benutzers ändern | Die neuen Ausleihtage werden korrekt gespeichert. | ja | - |

|  |  |  |  |
| --- | --- | --- | --- |
| **Destruktive Testfälle** | | | |
| **Test** | **Erwartetes Ergebnis** | **Bestanden** | **Bemerkung** |
| Benutzer erstellen - Unvollständige Eingaben | Eine Fehlermeldung wird angezeigt, und der Benutzer wird nicht erstellt. | Ja | - |
| Anmelden - Falsche Zugangsdaten | Eine Fehlermeldung wird angezeigt, und der Benutzer wird nicht eingeloggt. | Ja | - |
| Admin - Ungültige Änderungen an Ausleihen | Es wird ein Standard Wert verwendet. | ja | - |

# Deployment

## Benutzeranleitung

Mehr über dieses Thema findet man im README.

### Backend

Das Backend kann mit dem deploy.sh Script in ./backend/ direkt zu Google Cloud Run Deployed werden. Hierzu braucht es lediglich ein Google Cloud Projekt mit den richtigen Berechtigungen und die entsprechenden Umgebungsvariablen in Google Secrets Manager.

Lokal kann das Backend mit den zwei Scipts build-docker.sh und run-docker.sh in ./backend/ ausgeführt werden. Hierzu braucht man auch keine jdk, lediglich nur Docker.

### Frontend

Das Frontend wird mit dem Befehl «npm run build» gebundelt und dann mit dem Befehlt «firebase deploy» auf eine Google Firebase Seite deployed.

Lokal kann man die webseite wie folgt ausführen: «npm run dev»

(für beides müssen zuerst die Dependencys installiert werden: «npm i»)

## Live Deployment

Auf <https://jukestack.ch/>

Login auf <https://jukestack.ch/app/login>

Beispiel Benutzer: [user1@example.com](mailto:user1@example.com) mit Password: user1@example.com