Änderungspunkte

Es könnte sich nach einer Testphase herausstellen, dass es sinnvoll ist, an einigen Stellen in der App Änderungen vorzunehmen. Um diese Änderungen zu vereinfachen, werden in diesem Kapitel die wahrscheinlichsten Änderungspunkte aufgelistet und kurz erklärt, wo man im Code einsetzen müsste, um diese Änderungen anzuwenden.

Backend

Auslaufzeit des Tokens

Das Token, das zur Authentifizierung der User verwendet wird, hat eine voreingestellte Lebensdauer von 5 Stunden, nach deren Ablauf ein erneutes Login erforderlich ist. Falls die Lebensdauer geändert werden soll, kann diese in `backend/src/app/config.py` angepasst werden.
Es ist zu beachten, dass zur Zeit keine Möglichkeit besteht ein Token als ungültig zu markieren, außer die Lebensdauer zu überschreiten oder den im Token gespeicherten User aus der Datenbank zu entfernen.

Voreinstellungen der Queue-Settings

Die Settings, welche Admins während der Laufzeit der App über das Frontend verändern können, beziehen ihre Voreinstellungen beim Start der App aus `backend/src/app/queue/config.py`. Dort können sie nach Bedarf angepasst werden.

Einschränkungen für Nutzer*innen beim Hinzufügen eines Songs

Sollte gewünscht sein die Einschränkungen für Nutzer*innen beim Hinzufügen eines Songs zu ändern, können in der Datei `backend/src/app/queue/routes.py` in der Funktion `add_song_to_queue` nach Bedarf Einschränkungen entfernt oder hinzugefügt werden.

Einschränkungen für Admins beim Hinzufügen eines Songs

Ebenso können die Einschränkungen für Admins beim Hinzufügen eines Songs in der Datei `backend/src/app/admin/routes.py` in der Funktion `add_song_to_queue_as_admin` abgeändert werden.

AutoDocs

FastAPI erstellt automatische Dokumentationen für das Backend über Swagger UI und ReDoc. Sollte dies nicht erwünscht sein, können `SWAGGER_UI_URL` und `REDOC_URL` aus `.env` entfernt werden. Alternativ können die URLs auch nach Bedarf angepasst werden.

Datenbank

Die Datenbank wird - sofern sie noch nicht vorhanden ist - automatisch von Alembic erstellt. Sollte dies nicht erwünscht sein, kann das Verhalten in `backend/Dockerfile` geändert werden.
Beim Starten der App wird die Datenbank automatisch mit Daten aus Ultrastar-Dateien befüllt und es wird ein User mit den entsprechenden Daten aus der `.env`-Datei als User mit Admin-Rechten zugefügt. Dies passiert in `backend/src/app/main.py`.

Falls die SQLite-Datenbank durch eine andere ersetzt werden oder der Pfad geändert werden soll, kann dies in `.env` in der Variablen `DATABASE_URL` angepasst werden.
Sofern weiterhin mit Alembic gearbeitet wird, muss der Pfad ebenso in `backend/src/alembic/env.py` bei `config.set_main_option` geändert werden.
Es ist zu beachten, dass die Anwendung eine Datenbank erwartet, die Asynchronität unterstützt.

Sollte eine nicht SQL-basierte Datenbank aufgesetzt werden, muss die Datei `backend/src/app/database.py` angepasst werden.
Außerdem müssen unter Umständen die `crud.py`-Dateien in `backend/src/app/auth` und `backend/src/app/songs`, sowie die Datenbank-Modelle in `backend/src/app/auth/models.py` und `backend/src/app/songs/models.py` angepasst werden.

Anm.: Es ist zu beachten, dass die SQLite-Datenbank bei der Zerstörung des Docker-Containers ebenso zerstört wird. Daher ist es empfehlenswert die SQLite-Datenbank durch eine externe Datenbank, welche über Docker mit der Anwendung verbunden ist, zu ersetzen.

FastAPI

Falls das FastAPI-Framework ausgetauscht werden soll, müssen einige Dateien verändert bzw. gelöscht werden.

In `backend/src/app/main.py` wird die FastAPI-App erstellt, was von dem neuen Framework ersetzt werden sollte.

In `backend/src/app/dependencies.py` und `backend/src/app/auth/dependencies.py` befinden sich FastAPI spezifische Dependencies.

Folgende Dateien enthalten die API-Endpoints und sollten demnach vollständig von dem neuen Framework ersetzt werden:
`backend/src/app/admin/routes.py`
`backend/src/app/auth/routes.py`
`backend/src/app/queue/routes.py`
`backend/src/app/songs/routes.py`

Die Exceptions in `backend/src/app/auth/exceptions.py` und `backend/src/app/songs/exceptions.py` sind alle von FastAPI-Modulen abhängig und sollten ersetzt werden, falls das neue Framework damit nicht kompatibel ist.
In `backend/src/app/queue/exceptions.py` befinden sich ebenfalls einige Exceptions, die von diesen FastAPI-Modulen abhängen.

Da die Tests in `backend/tests/unit/api` die FastAPI-API testen und einige aus FastAPI importierte Module benutzen, müssen diese unter Umständen ebenfalls angepasst oder ersetzt werden.

Sveltekit

Sollte der Bedarf bestehen Sveltekit als Frontend-Framework zu ersetzen, kann der Inhalt des Ordners `frontend/` komplett gelöscht und gegen das entsprechende Framework ausgetauscht werden.

Frontend

Startseite

Sollte es sich als unpraktisch erweisen auf der Seite mit der vollständigen Songübersicht zu starten, da z. B. ältere Geräte die Seite nicht performant laden können, kann die Startseite verändert werden, indem man die in `frontend/routes/` gespeicherten `+page.svelte`- und `+page.server.js`-Dateien in einen neuen Unterordner verschiebt und die `+page.svelte` und - sollte diese existieren - `+page.server.js` aus dem gewünschten Ordner (der Ordnername entspricht dem URL-Pfad) direkt nach `/routes/` verschiebt.
Die Buttons und deren Verlinkung im HTML-Teil von `frontend/lib/Navbar.svelte` sollten entsprechend angepasst werden.

Beispiel *Queueübersicht zur Startseite machen*:

```
└──routes
    ├──...
    ├──queue
    │   └──+page.svelte  # Queueübersicht
    ├──+layout.svelte
    ├──+page.server.js  # Songübersicht
    └──+page.svelte  # Songübersicht
```

(Beispiel *Queueübersicht zur Startseite machen*: Alte Ordnerstruktur)

```
└──routes
    ├──...
    ├──songs
```

```
│   ├──+page.server.js  # Songübersicht
│   └──+page.svelte  # Songübersicht
├──+layout.svelte
└──+page.svelte  # Queueübersicht
```

(Beispiel *Queueübersicht zur Startseite machen*: Neue Ordnerstruktur)

Login-Button

Sollte es als sinnvoll erachtet werden, kann der Login-Button entfernt werden. Dies könnte von Vorteil sein, da sich normale Nutzer*innen durch das Vorhandensein eines Login-Buttons irritiert fühlen könnten, da es für sie keine Möglichkeit gibt einen Account zu erstellen. Der Login-Button dient lediglich der Authentifizierung der Admins, was allerdings auch über den direkten Zugriff über den URL-Pfad `/login` erfolgen kann.
Dazu muss in `frontend/lib/Navbar.svelte`

```
{:else}
    <a class="btn btn-outline-success me-2" href="/login">Login</a>
```

entfernt werden.