Music Sync

iOS Abschlussprojekt

12.09.2017

Nils Axer

# Inhalt

[1 Inhalt 1](#_Toc493001816)

[2 Architektur 2](#_Toc493001817)

[3 App 3](#_Toc493001818)

[3.1 Features 3](#_Toc493001819)

[3.2 Importieren der App in Xcode 4](#_Toc493001820)

[3.3 View Controllers 5](#_Toc493001821)

[3.4 Benutzte Bibliotheken 6](#_Toc493001822)

[3.5 Datenbankschema der App 7](#_Toc493001823)

[3.6 Bedienung 8](#_Toc493001824)

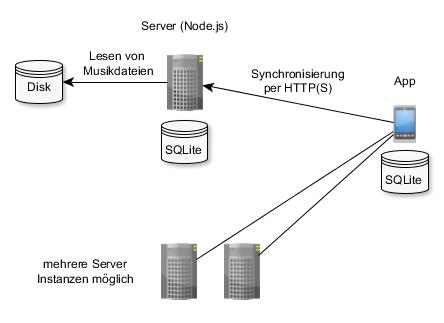
[4 Server 19](#_Toc493001825)

[4.1 Konfiguration 19](#_Toc493001826)

[4.2 Starten des Servers 20](#_Toc493001827)

[4.3 Datenbankschema Server 21](#_Toc493001828)

# Architektur



# App

## Features

Music Player, der sich mit einem Server verbindet und synchronisiert und die Musikdateien davon herunterladen kann.

**Implementierte Features:**

* Metadaten-Synchronisierung mit Server
* Bibliothek nach Schema Interpret -> Album -> Song-Liste browsen
* Downloaden und Abspielen von Songs
* Hinzufügen und Editieren von Servern
* Mehrere Bibliotheken pro Server

**Nicht implementierte Features**

Einige Features konnte ich zeitlich leider nicht mehr verwirklichen:

* Playlists
* Shuffle
* Streaming
* Passwort für Server
* Unit Tests, UI Tests

Außerdem funktioniert das Hinzufügen von mehreren Servern gleichzeitig noch nicht richtig.

## Importieren der App in Xcode

Der Source Code der App ist unter [*https://github.com/Nils1337/MusicSync-iOS*](https://github.com/Nils1337/MusicSync-iOS)zu finden. Da ein paar zusätzliche Bibliotheken benötigt werden, werden die Abhängigkeiten von CocoaPods (<https://cocoapods.org/>) verwaltet. Zum Importieren der App in Xcode werden folgende Schritte benötigt:

* Code von Github herunterladen, z.B. *git clone https://github.com/Nils1337/MusicSync-iOS.git*
* Gegebenenfalls CocoaPods installieren, z.B. *sudo gem install cocoapods*
* Im Verzeichnis des Source Codes *pod install* ausführen
* Die Datei *MusicSync.xcworkspace* mit Xcode öffnen (nicht *MusicSync.xcodeproj*)

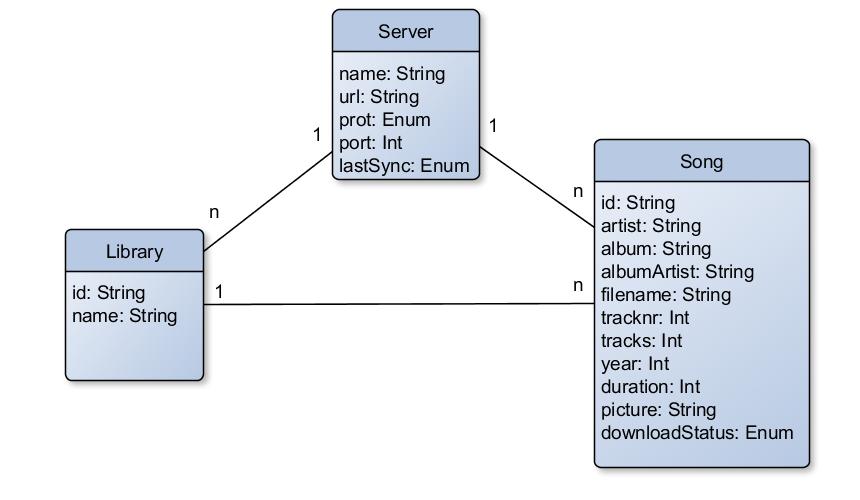
## View Controllers

|  |  |
| --- | --- |
| **Controller** | **Funktion** |
| SettingsTableViewController | Zeigt Tabelle mit Einstellungen |
| AlbumsTableViewController | Zeigt Alben eines Interpreten |
| ArtistsTableViewController | Zeigt alle Interpreten einer Bibliothek |
| LibraryViewController | Zeigt alle verfügbaren Server und ihre Bibliotheken. Bibliotheken lassen sich auswählen. Wird im Drawer verwendet |
| PlaylistsTableViewController | Sollte Playlists anzeigen, macht aber im Moment noch nichts |
| PlayingViewController | Zeigt aktuelle Wiedergabe an |
| ServersTableViewController | Zeigt alle hinzugefügten Server an |
| ServerDetailsViewController | Editieren und Hinzufügen von einem Server |
| AllSongsViewController | Zeigt alle Songs der Bibliothek an. Bettet einen **SongsTableViewController** ein. |
| SongsOfAlbumViewController | Zeigt alle Songs eines Albums an. Bettet einen **SongsTableViewController** ein. |
| SongsTableViewController | Zeigt eine Tabelle mit Songs an. |

## Benutzte Bibliotheken

|  |  |  |
| --- | --- | --- |
| **Bibliothek** | **Link** | **Beschreibung** |
| MMDrawerController | <https://github.com/mutualmobile/MMDrawerController> | Drawer-Implementierung (ähnlich wie in Android) |
| ReachabilitySwift | <https://github.com/ashleymills/Reachability.swift> | Swift Implementierung von der Reachability Klasse von Apple. Wird benutzt um Informationen über die Konnektivität zu erhalten. |
| Validator | <https://github.com/adamwaite/Validator> | Validierung von Eingabefeldern. Wird im **ServerDetailsViewController** benutzt. |
| CocoaLumberjack | <https://github.com/CocoaLumberjack/CocoaLumberjack> | Logging Framework |
| Sync | <https://github.com/3lvis/Sync>, <https://github.com/Nils1337/Sync> | Framework zur Synchronisierung von Core Data mit einem Server. Da ich ein paar Anpassungen vornehmen musste, habe ich ein Fork des Frameworks erstellt, welchen ich benutze. |

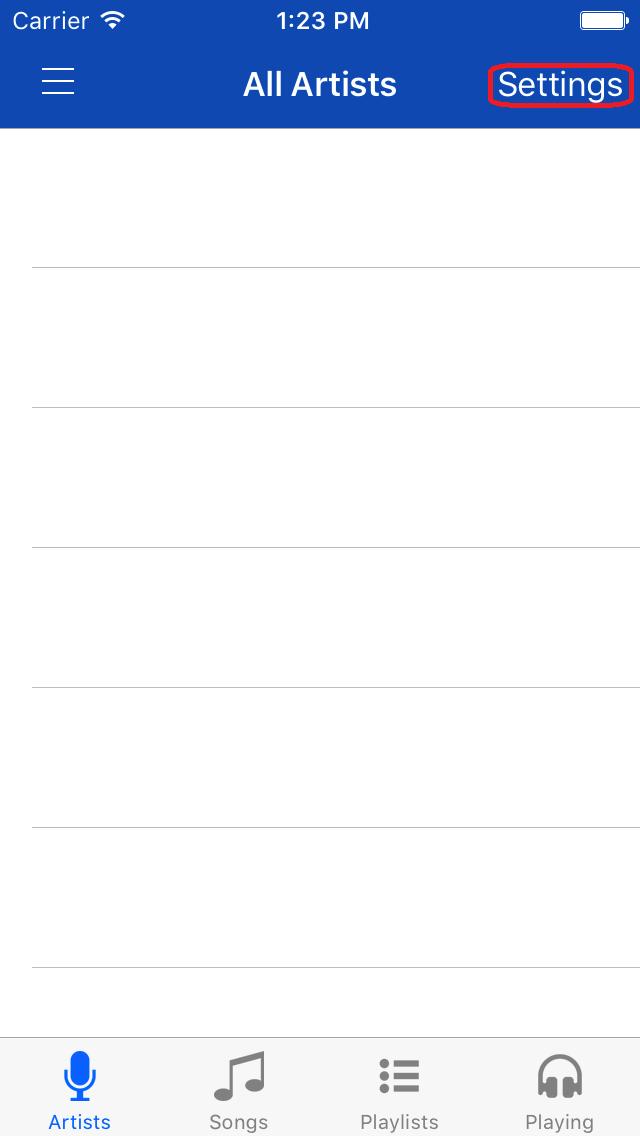
## Datenbankschema der App



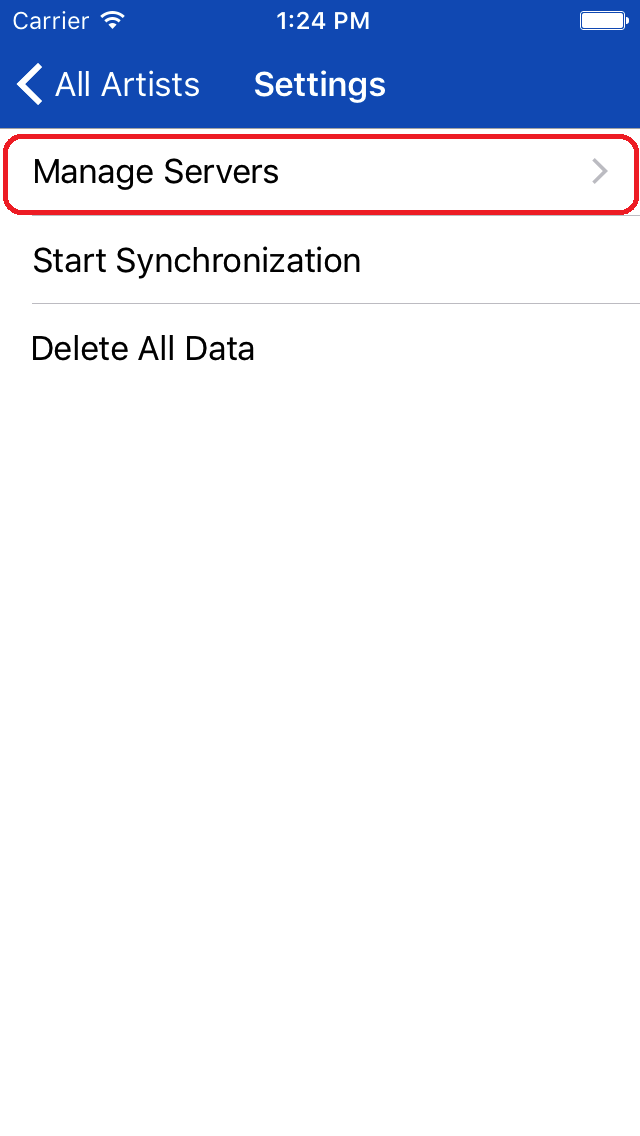
## Bedienung

Zunächst muss in der App ein Server hinzugefügt werden. Dazu müssen folgende Schritte ausgeführt werden:

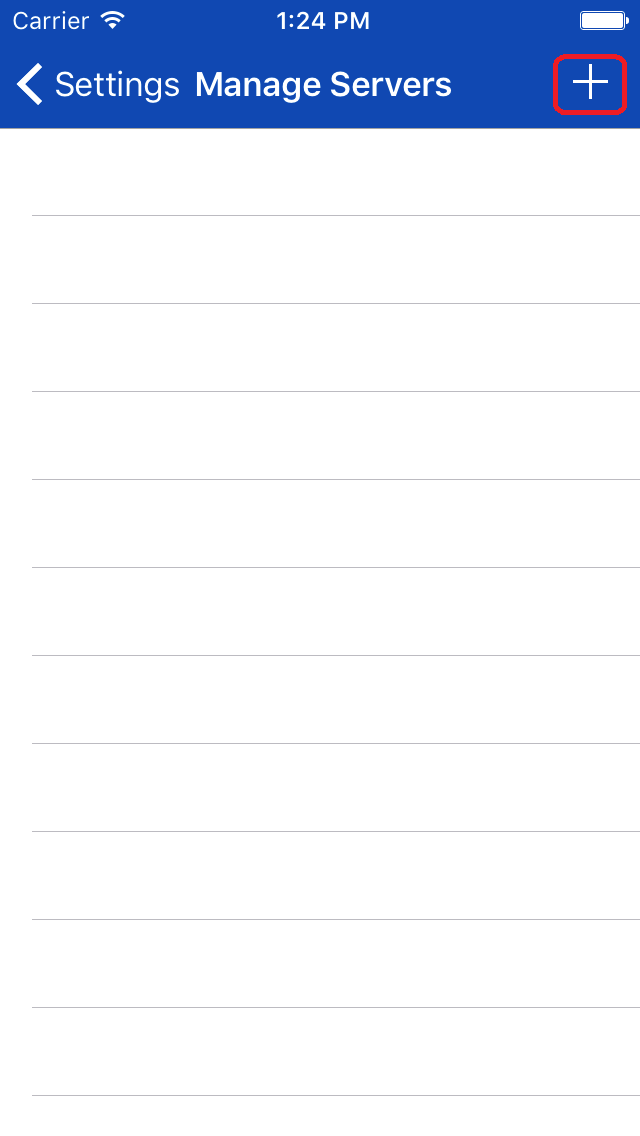
Klicke in der Navigation Bar auf Settings



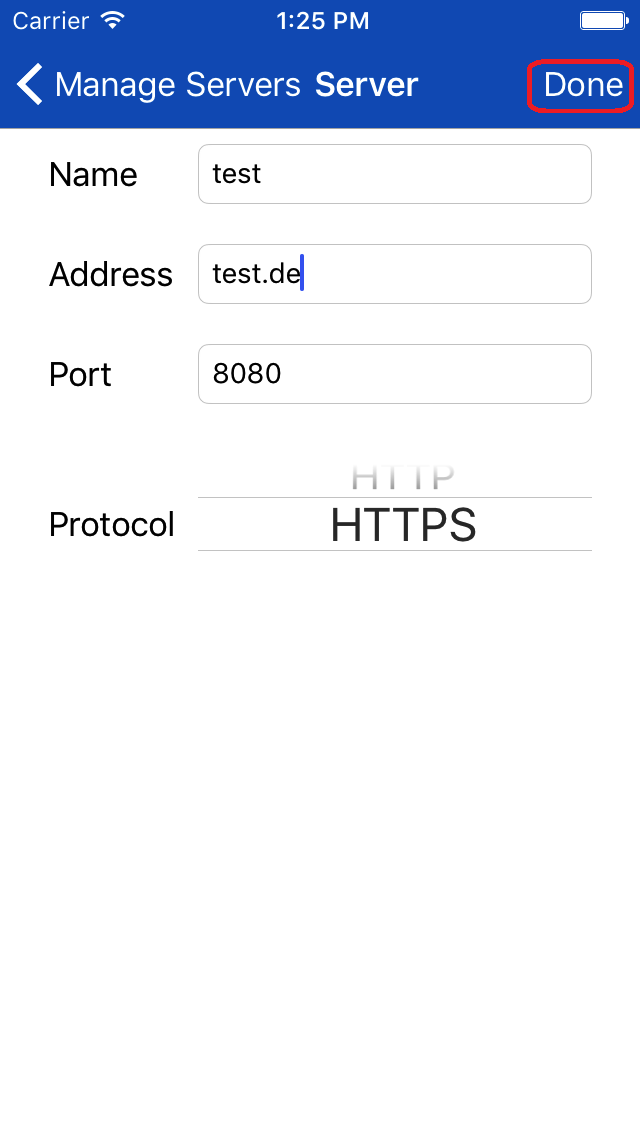
Wähle „Manage Servers“



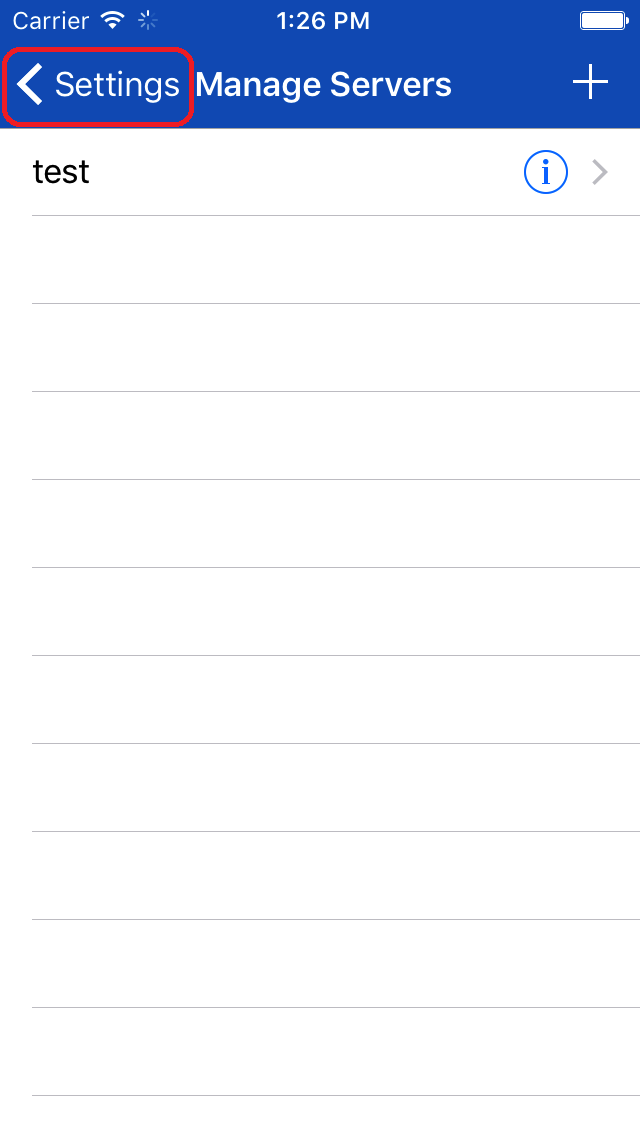
Klicke auf das Plus-Symbol in der Navigation Bar



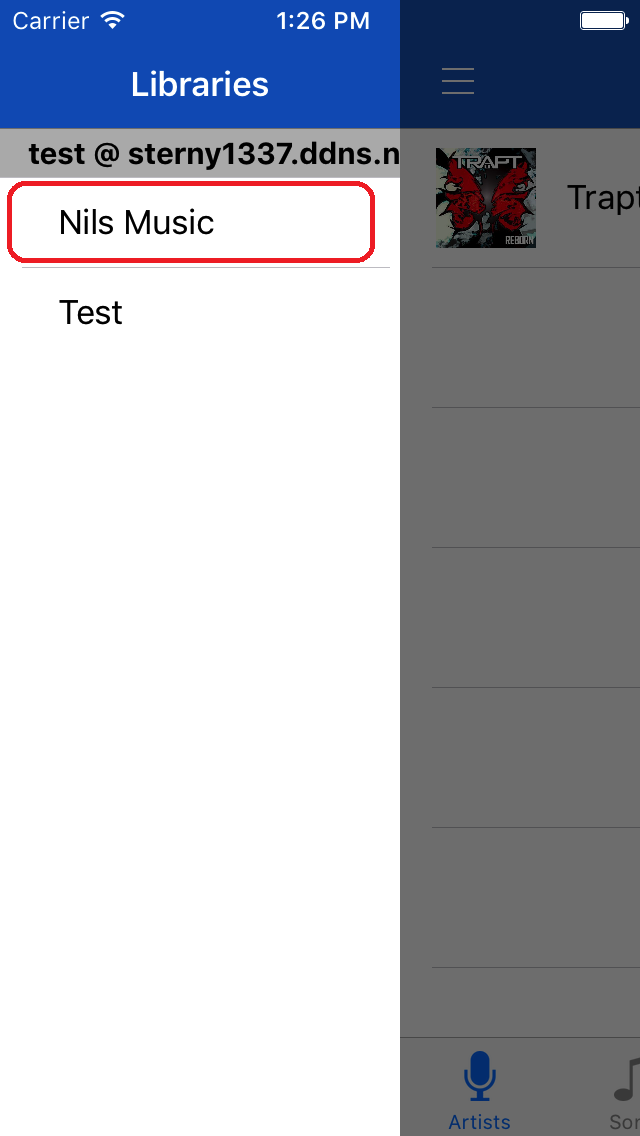
Gebe die Daten des Servers ein und klicke auf „Done“ in der Navigation Bar.



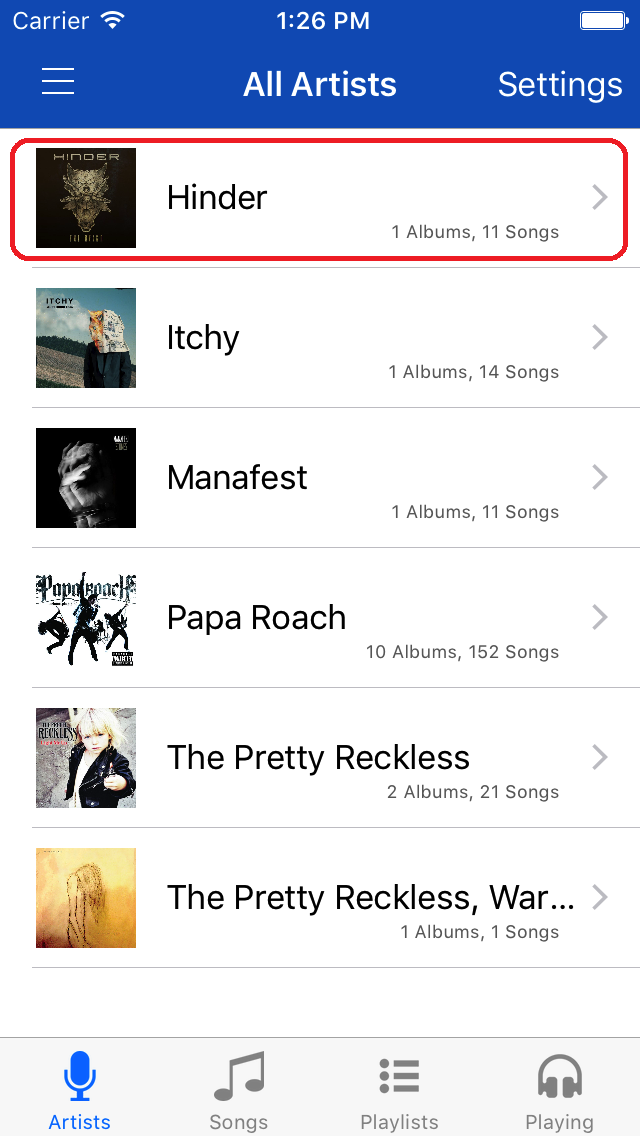
Navigiere zurück zum Hauptbildschirm.



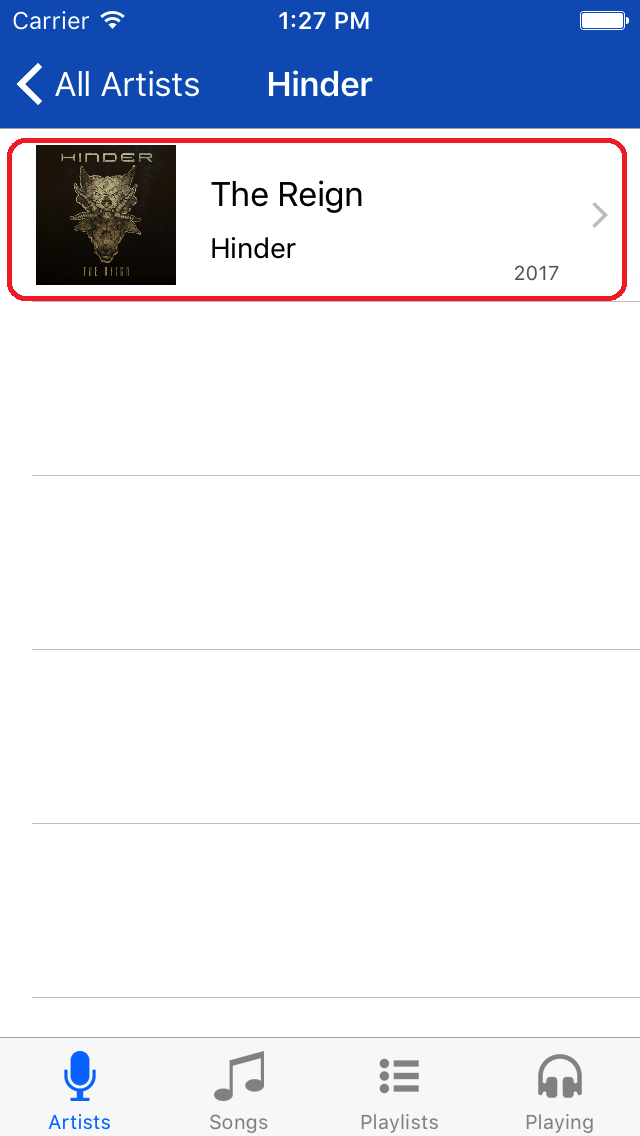
Nun kann im Drawer eine Bibliothek des Servers ausgewählt werden.



Danach werden alle Songs der Bibliothek angezeigt. Zum Navigieren durch die Songs, wähle einen Interpreten.



Hier werden nun alle Alben dieses Interpreten angezeigt. Wähle ein Album.

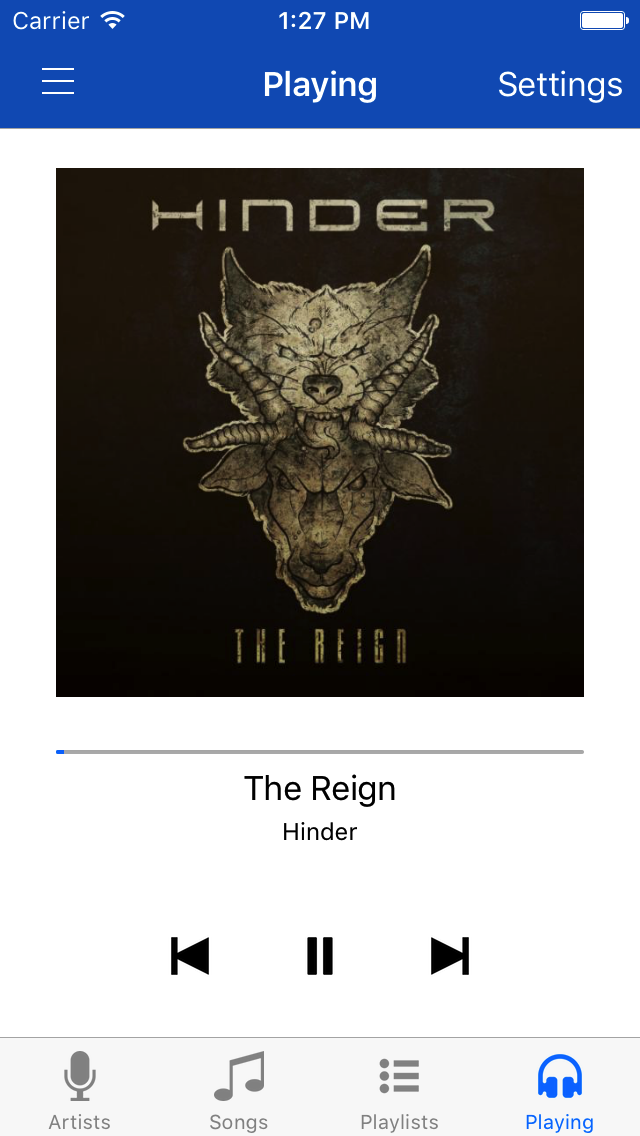


Hier werden nun alle Songs des Albums angezeigt. Das Wolken-Symbol zeigt an, dass sich die Datei noch auf dem Server befindet. Lade ein paar Songs vom Server mit einem Klick auf den Song herunter.



Nun kann mit einem Klick auf einen heruntergeladenen Song das Abspielen gestartet werden.





# Server

Der Source Code des Servers ist unter <https://github.com/Nils1337/MusicSync-Server> zu finden. Der Server wurde mit Node.js implementiert und startet einen HTTP-Server der eine REST-Schnittstelle bereitstellt. In *config.js* kann der Server konfiguriert werden.

## Konfiguration

|  |  |  |
| --- | --- | --- |
| **Variable** | **Typ/Werte** | **Beschreibung** |
| debug | Boolean | Sollen mehr Nachrichten auf die Konsole ausgegeben werden? |
| Port | Int | Port, unter dem der Server laufen soll |
| dbFileName | String | Dateiname der SQLite Datei, die angelegt wird |
| dropSongsOnStart | Boolean | Löscht beim Starten des Servers alle vorher vorhandenen Songs (zum Testen/ zur Fehlerbehebung) |
| dropLibrariesOnStart | Boolean | Löscht beim Starten des Servers alle vorher vorhandenen Bibliotheken (zum Testen/ zur Fehlerbehebung) |
| deleteDelay | Int | Zeitspanne, nach der nach dem Start nicht geupdatete Songs (nicht mehr im Dateisystem vorhanden) gelöscht werden (in ms) |
| protocol | „http“ oder „https“ | Soll HTTPS verwendet werden? |
| privateKeyPath | String | Pfad zum privaten SSL-Key (nur für HTTPS benötigt) |
| certificatePath | String | Pfad zum SSL-Zertifikat (nur für HTTPS benötigt) |
| libraries | Array mit Library Objekten. Library Objekte benötigen die Attribute   * „name“: Name der Library * „path“: Pfad unter dem die Musikdateien zu finden sind, die zu der Bibliothek gehören | Alle Bibliotheken, die vom Server verwaltet werden. |

## Starten des Servers

Der Server kann direkt mit Node.js oder in einem Docker Container ausgeführt werden.

**Node.js**:

* Gegebenenfalls Node.js und NPM installieren
* *npm install* im Ordner mit Source Code ausführen
* Server mit *npm start* starten

Der Server läuft dann unter dem in *config.js* angegebenen Port.

**Docker-Compose**:

* *docker-compose build musicsync*
* *docker-compose up*

Der Server läuft dann auf Port 80 des Docker-Containers

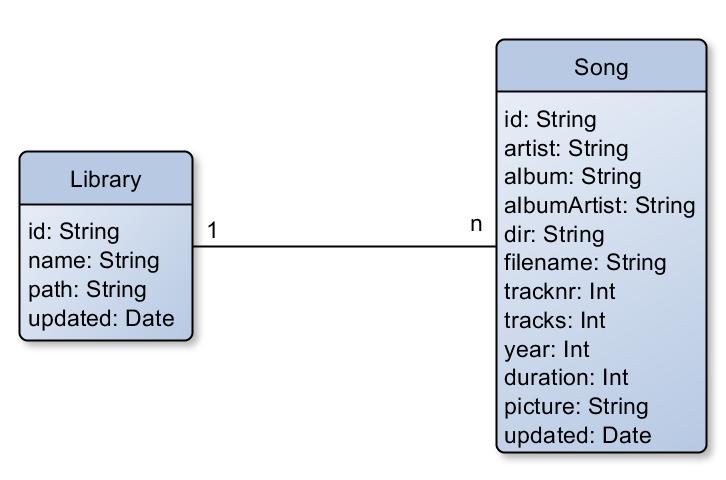
**Docker:**

* *docker build -t musicsync .*
* *docker run -p 80:80 musicsync*

Der Server läuft dann auf dem mit –p gemappten Port des Docker-Containers (hier 80).

Zum Testen habe ich unter */music* ein paar wenige Musikdateien committed. Man sollte da auch einfach neue Songs in den Ordner packen können bevor oder auch während man den Server ausführt und diese sollten dann korrekt in der App angezeigt werden, aber das habe ich noch nicht ausführlich getestet.

## Datenbankschema Server

****