

# Abschlussprojekt Nils Axer – Music Sync

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> 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

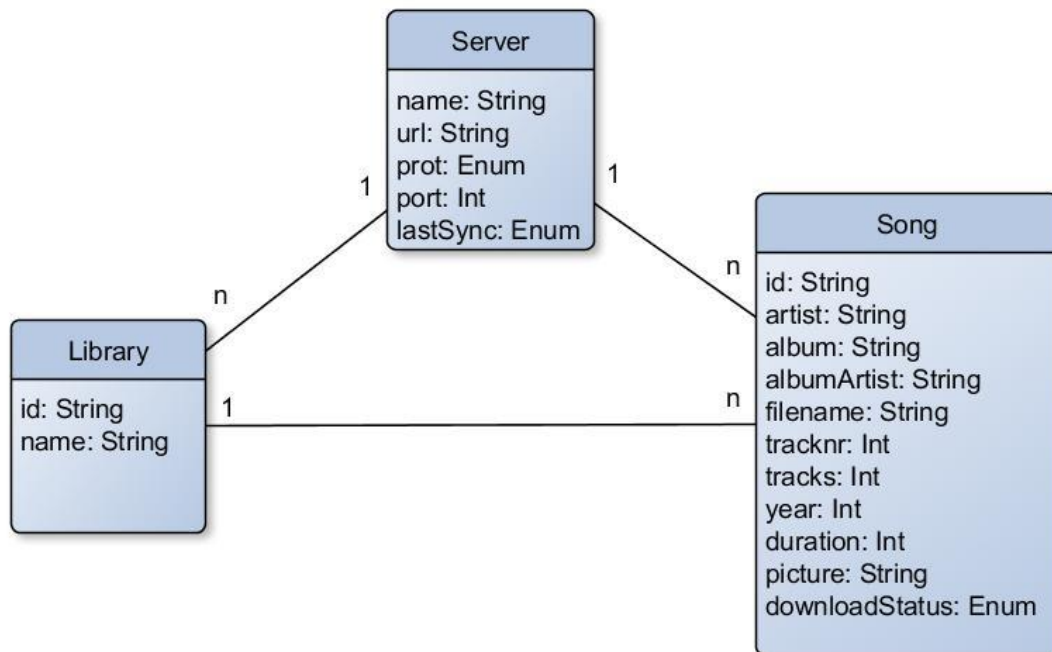
SettingsTableViewController	Zeigt Tabelle mit Einstellungen
AlbumsTableViewController	Zeigt Alben eines Interpreten
ArtistsTableViewController	Zeigt alle Interpreten einer Bibliothek
LibraryViewController	Zeigt alle verfügbaren Server und ihre Bibliotheken. Bibliotheken lassen sich

	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 <b>SongsTableViewController</b> ein.
SongsOfAlbumViewController	Zeigt alle Songs eines Albums an. Bettet einen <b>SongsTableViewController</b> ein.
SongsTableViewController	Zeigt eine Tabelle mit Songs an.

## Benutzte Bibliotheken

Bibliothek	Link	Beschreibung
MMDrawerController	<a href="https://github.com/mutualmobile/MMDrawerController">https://github.com/mutualmobile/MMDrawerController</a>	Drawer-Implementierung (ähnlich wie in Android)
ReachabilitySwift	<a href="https://github.com/ashleymills/Reachability.swift">https://github.com/ashleymills/Reachability.swift</a>	Swift Implementierung von der Reachability Klasse von Apple. Wird benutzt um Informationen über die Konnektivität zu erhalten.
Validator	<a href="https://github.com/adamwaite/Validator">https://github.com/adamwaite/Validator</a>	Validierung von Eingabefeldern. Wird im <b>ServerDetailsViewController</b> benutzt.
CocoaLumberjack	<a href="https://github.com/CocoaLumberjack/CocoaLumberjack">https://github.com/CocoaLumberjack/CocoaLumberjack</a>	Logging Framework
Sync	<a href="https://github.com/3lvis/Sync">https://github.com/3lvis/Sync</a> , <a href="https://github.com/Nils1337/Sync">https://github.com/Nils1337/Sync</a>	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



## 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 <ul style="list-style-type: none"><li>- „name“: Name der Library</li><li>- „path“: Pfad unter dem die Musikdateien zu finden sind, die zu der Bibliothek gehören</li></ul>	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*

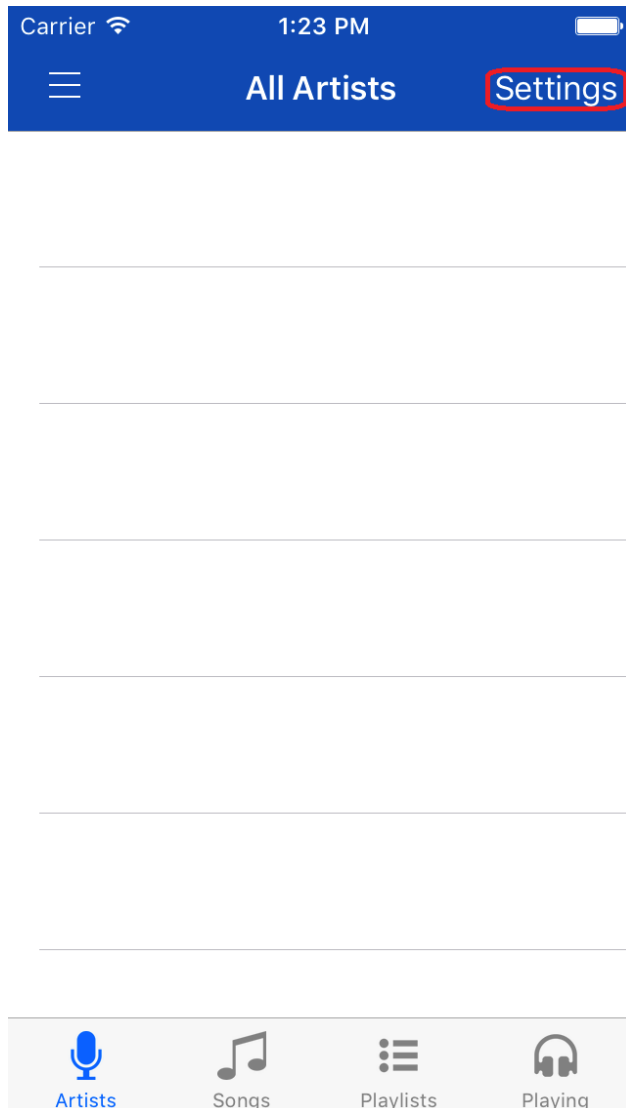
Der Server läuft dann auf dem mit *-p* gemappten Port des Docker-Containers (hier 80).

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.

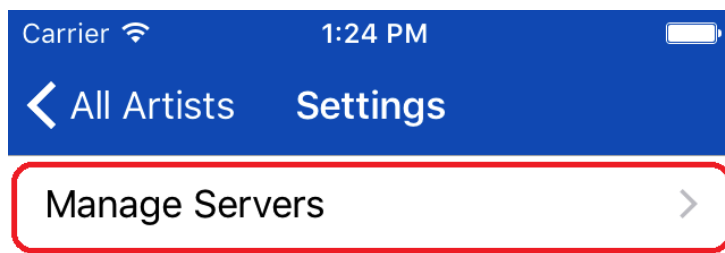
## Bedienung der App

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

Klicke in der Navigation Bar auf Settings



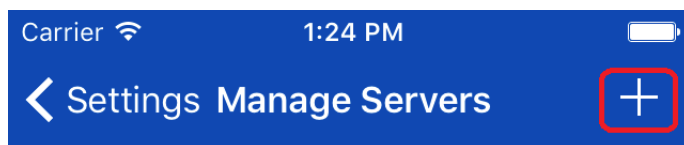
Wähle „Manage Servers“



Start Synchronization

Delete All Data

Klicke auf das Plus-Symbol in der Navigation Bar



---

---

---

---

---

---

---

---

---

---

---

---



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

Carrier 1:25 PM

< Manage Servers Server Done

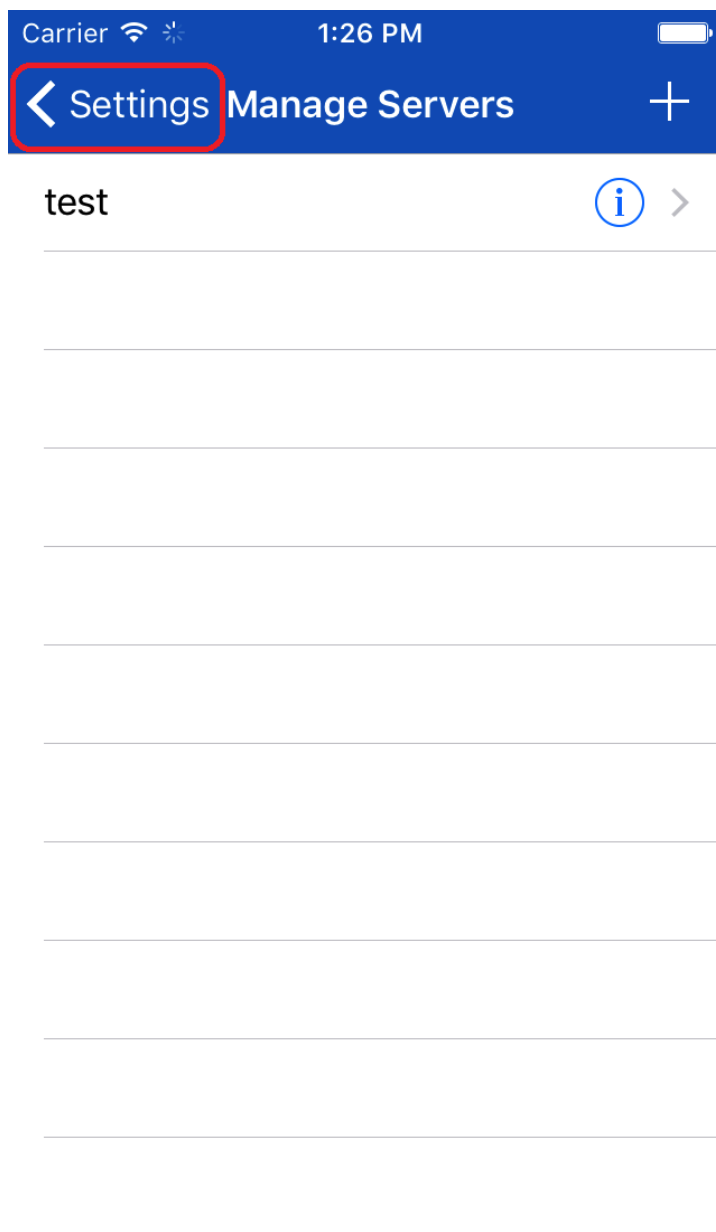
Name test

Address test.de

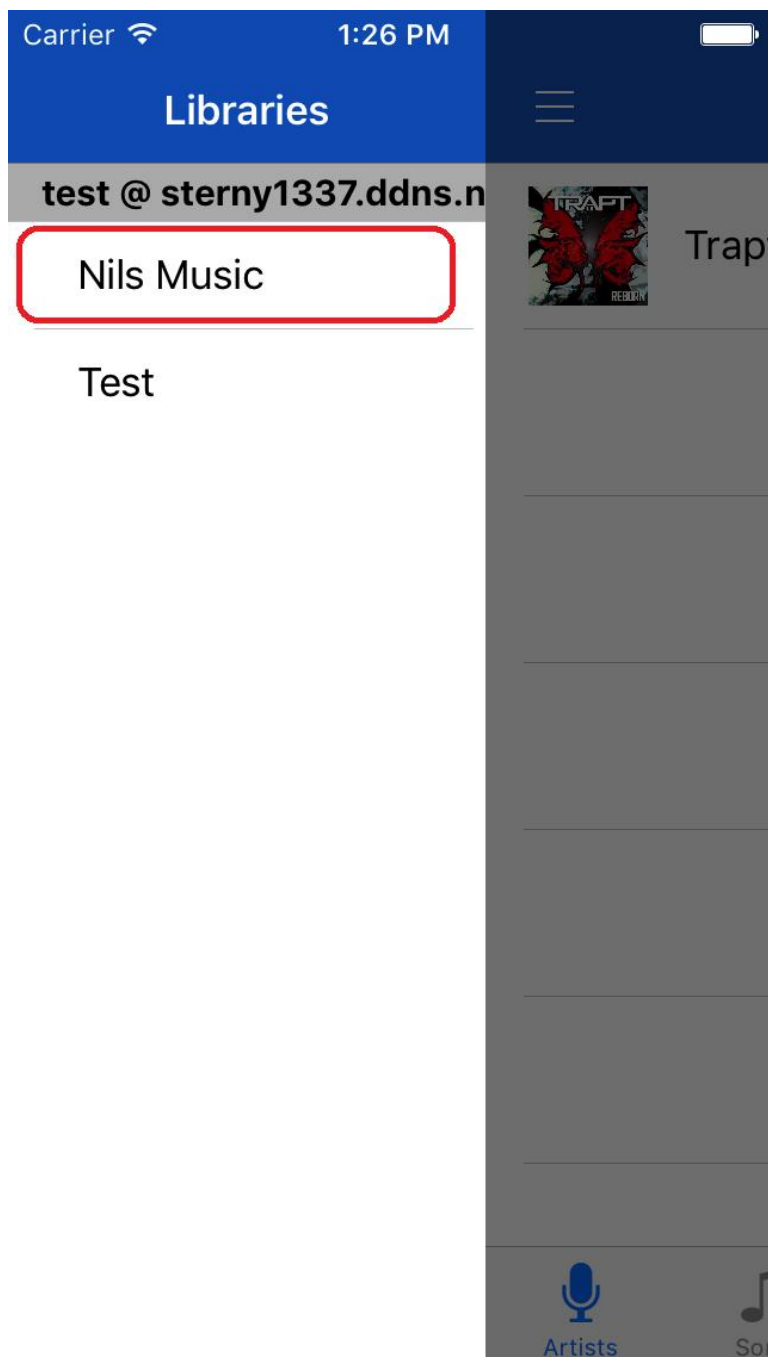
Port 8080

Protocol HTTP  
HTTPS

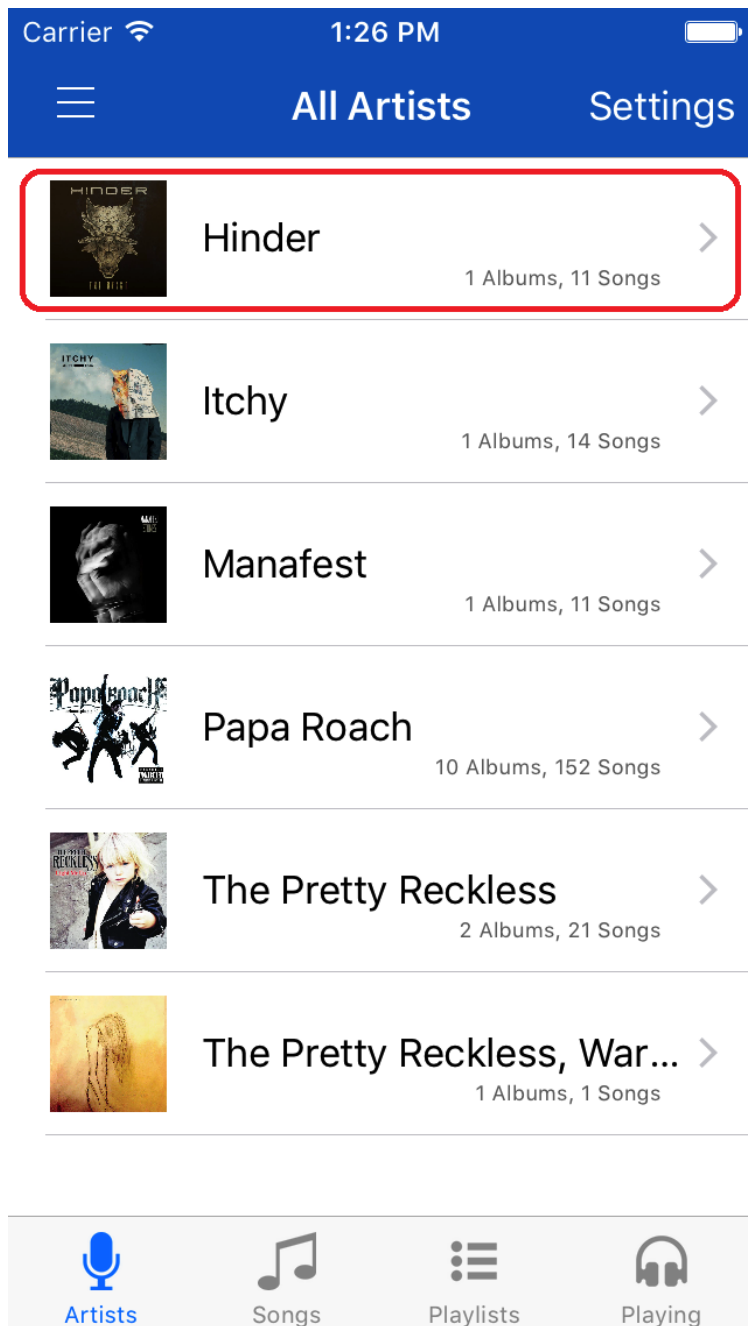
Navigiere zurück zum Hauptbildschirm.



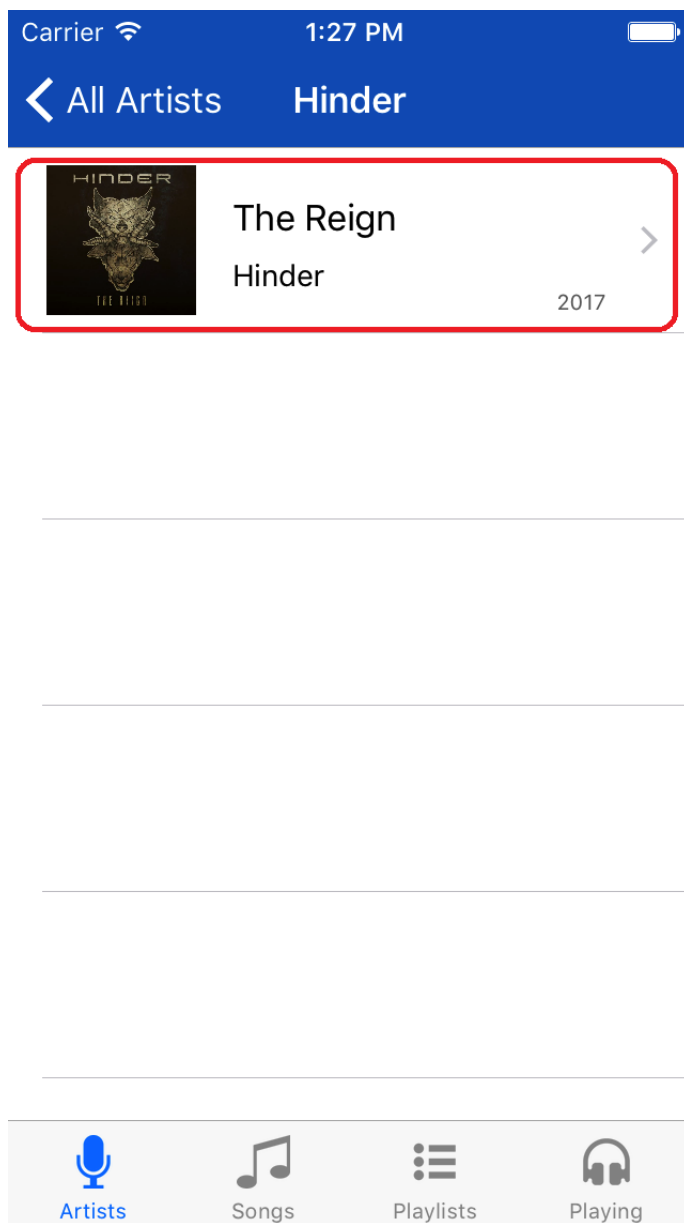
Nun kann im Drawer eine Bibliothek des Servers ausgewählt werden.



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.

Carrier

1:27 PM

<


Hinder

1

The Reign

Hinder

03:12




2

Burn It Down

Hinder

03:38




3

King Of The Letdown

Hinder

03:16




4

Remember Me

Hinder

03:15




5

Too Late

Hinder

04:27




6

Another Way Out

Hinder

03:37




7

Making It Hard

Hinder

03:34




8

Drink You Away

Hinder

03:33




9

Play To Win

Hinder

03:46




10

Long Gone


Hinder

03:49




11


Loser's Salute




Artists



Songs

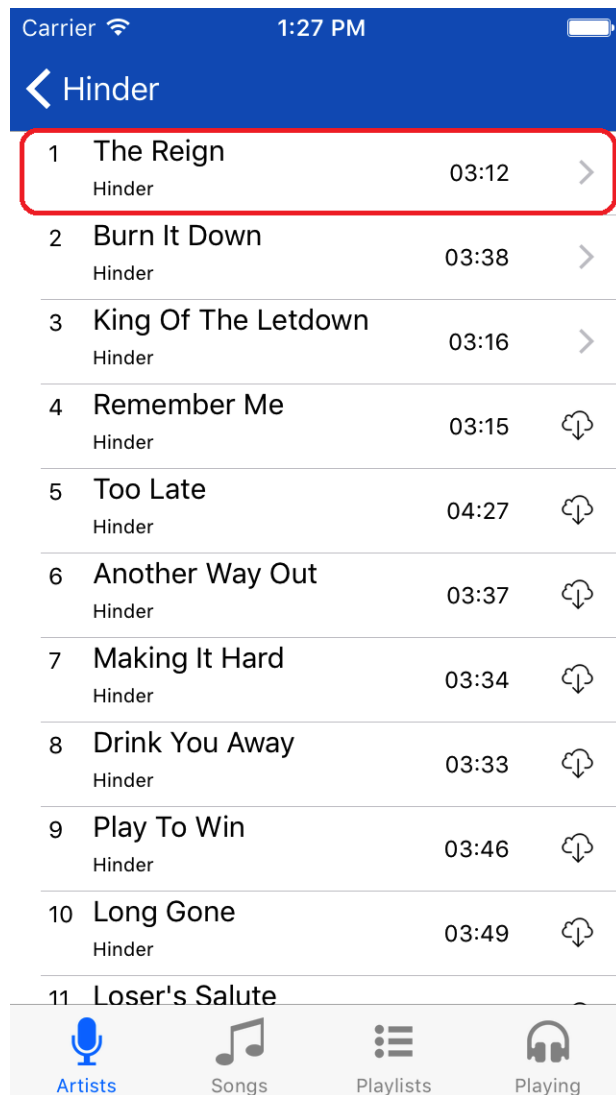


Playlists



Playing

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





Playing

Settings



The Reign

Hinder



Artists



Songs



Playlists



Playing