

Apple App Store und Google Play Store

Heute schauen wir die einzelnen Schritte an, wie wir unsere App in die jeweiligen Stores hochladen können. Für Apple (99\$ / year) und Google (25\$ one time) wirst du einen Developer Account brauchen. Beginnen wir mit iOS, für Android ist der Prozess einiges einfacher und befindet sich gegen Schluss dieses Dokuments.

iOS: Release für App Store

iOS: App ID

Jede iOS App braucht eine ID, sprich eine App ID. Diese muss bei Apple zuerst registriert werden. Im Developer Center (<https://developer.apple.com/account/>), unter Certificates, Identifiers & Profiles > Identifiers > App IDs, kannst du eine neue App ID erstellen. Unter Explicit App ID musst du die ID angeben, welche du im config.xml deines Ionic Projektes angegeben hast. Sie wird üblicherweise in der **Reverse-Domain-Name** Schreibweise geschrieben, also z.B. ch.romanrast.rhyschwimme

```
app.module.ts      config.xml ✘      app.component.ts      overview.txt
1  <?xml version='1.0' encoding='utf-8'?>
2  <widget id="ch.romanrast.rhyschwimme" version="0.0.1" xmlns=
3    <name>Rhyschwimme</name>
```

The screenshot shows the Apple Developer Center interface for registering an App ID. On the left, there's a sidebar with navigation links for Apps, Certificates, Identifiers & Profiles, Devices, and Provisioning Profiles. Under Identifiers, 'App IDs' is selected. The main panel has a title 'Register iOS App IDs' and a sub-section 'ID Registering an App ID'. It contains fields for 'Name' (with a note about special characters) and 'App ID Prefix' (with a placeholder '(Team ID)'). Below this, there's a section for 'App ID Suffix' with a radio button for 'Explicit App ID' (selected) and a note about its use for Game Center, In-App Purchase, and Data Protection services. A note at the bottom states: 'If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must use an explicit app ID.'

The screenshot shows the 'App ID Suffix' section of the developer portal. It has two options: 'Explicit App ID' (selected) and 'Wildcard App ID'. Under 'Explicit App ID', it says: 'If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.' Below this, it says: 'To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.' A yellow box highlights the 'Bundle ID' field containing 'ch.romanrast.rhyschwimme'. Below the field, it says: 'We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).'

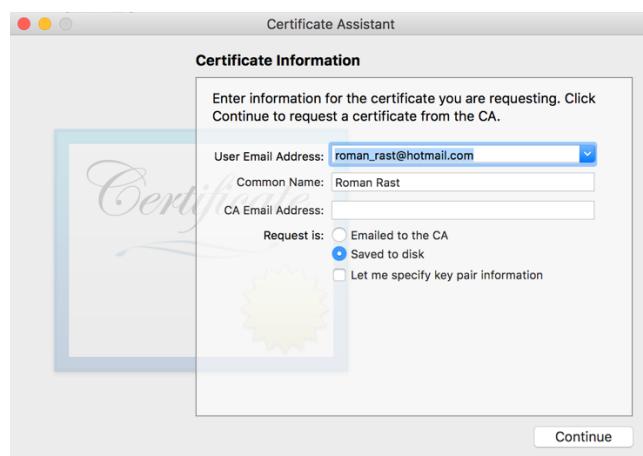
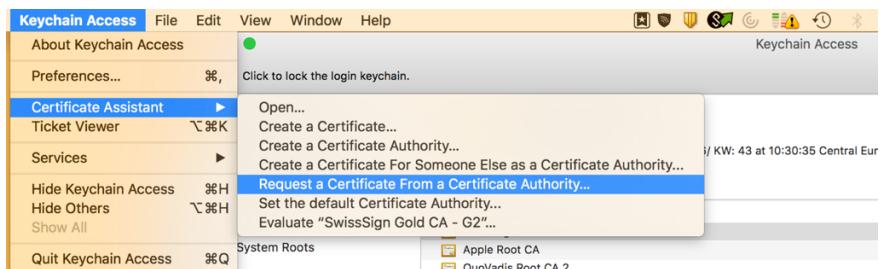
Danach weiter und registrieren.

Certificate

Als nächstes brauchen wir ein Certificate für unsere App. Dazu müssen wir zuerst ein

Certificate Signing Request File erstellen (CSR). Öffne dazu auf deinem Mac die Schlüsselbundverwaltung (Keychain Access).

Wähle zuerst *Certificates* in der Category Sektion aus und auf der rechten Seite *Apple Worldwide Developer Relations Certificate Authority* aus. Geh dann im Menu der Applikation (siehe Bild unten) unter Keychain Access > Certificate Assistant > Request a Certificate From a Certificate Authority.



Für Common Name, gib einen Namen für deinen Private Key (z.B. Max Muster Key) an.

Mobile Workshop FHNW 2018 - Ionic

Das Feld CA Email Address solltest du leer lassen und wähle **Save to disk aus**. Danach erhältst du das Signing Request File **xxx.certSigningRequest**.

Nun können wir das Certificate erstellen. Hierzu gibt es zwei Varianten, ein Development und ein Production Certificate. Mit der Development Variante haben wir die Möglichkeit zu debuggen und Infos in der Konsole zu bekommen, wir können aber nur auf max. 100 Devices die App installieren. Das Production Certificate kann jedoch für eine unlimitierte Anzahl Devices verwendet werden, jedoch fehlt dann der Debugmodus. Das Erstellen beider Zertifikate ist ziemlich ähnlich.

Gehe dazu im Apple Developer Center unter Certificates, Identifiers & Profiles > Certificates > Development oder Production. Hier kannst du zwischen **iOS App Development (dev)** und **App Store and Ad Hoc (prod)** wählen. Klick dich durch und du wirst nach dem CSR File gefragt, welches wir vorhin als xxx.certSigningRequest erstellt haben. Lade es hoch und danach kannst du das **xxx.cer** File runterladen.

The image consists of two vertically stacked screenshots of the Apple Developer Center interface, specifically the Certificates section.

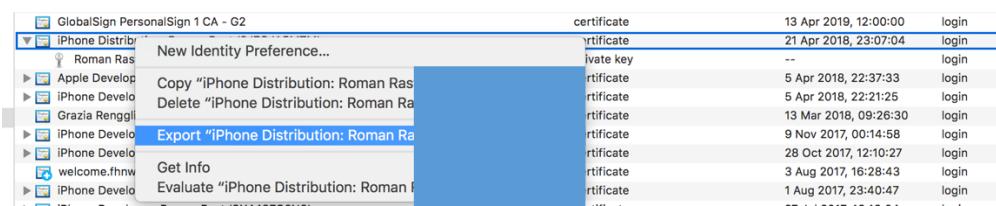
Screenshot 1 (Top): This screenshot shows the 'Request' step of the certificate generation process. On the left sidebar, 'Production' is selected under 'Certificates'. In the main area, there's a large button labeled 'Generate your certificate.' with a checkmark icon. Below it, a note states: 'When your CSR file is created, a public and private key pair is automatically generated. Your private key is stored on your computer. On a Mac, it is stored in the login Keychain by default and can be viewed in the Keychain Access app under the "Keys" category. Your requested certificate is the public half of your key pair.' A 'Choose File...' button is shown, with the file 'CertificateSigningRequest.certSigningRequest' selected.

Screenshot 2 (Bottom): This screenshot shows the 'Download' step of the process. The sidebar now includes 'Devices'. The main area displays a summary of the generated certificate: Name: iOS Distribution: Roman Rast, Type: iOS Distribution, Expires: Apr 15, 2019. A blue 'Download' button is visible. Below this, a 'Documentation' section links to the 'App Distribution Guide'.

So, als nächstes muss durch das **.cer** File in ein **.p12** File (verschlüsselt und enthält das Distribution Certificate) erstellt werden. Dazu kannst du auf das .cer File doppelklicken und es

Mobile Workshop FHNW 2018 - Ionic

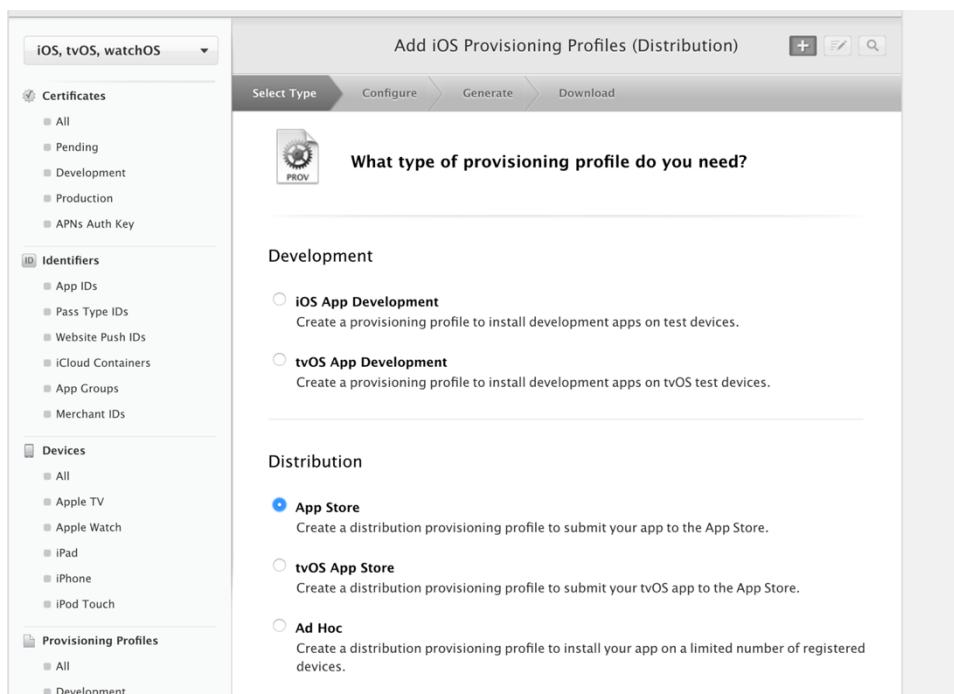
wird in die Schlüsselbundverwaltung kopiert, oder du kannst es einfach in die Schlüsselbundverwaltung unter **login** hereinziehen. Danach mit Rechtsklick > Export „iPhone Distribution:...“ das **xxx.p12** File herunterladen.



Gib ein Passwort an und speichere es.

Provisioning Profiles

Damit die App installiert werden kann, brauchen wir ein Provisioning Profile. Unter Certificates, Identifiers & Profiles > Provisioning Profiles > Development oder Distribution wählen. Danach auf Continue und die App ID auswählen, welche zur ID in deinem Ionic Projekt im config.xml steht. Klicke dich wie auf den folgenden Screens durch und lade das **xxx.mobileprovision** File herunter (siehe folgende Screens).



Mobile Workshop FHNW 2018 - Ionic

The screenshot shows the "Certificates, Identifiers & Profiles" section of the Apple Developer Portal. The left sidebar lists categories: Certificates (All, Pending, Development, Production, APNs Auth Key), Identifiers (App IDs, Pass Type IDs, Website Push IDs, iCloud Containers, App Groups, Merchant IDs), Devices (All, Apple TV, Apple Watch, iPad, iPhone, iPod Touch), and Provisioning Profiles (...). The main area is titled "Add iOS Provisioning Profiles (Distribution)" and shows a progress bar: Select Type → Configure → Generate → Download. A sub-section titled "Select App ID." displays a list of identifiers. One identifier, "Rhyschrimme (.ch.romanrast.rhyschrimme)", is highlighted with a blue selection bar. A note below explains the use of App IDs for Game Center, In-App Purchase, and Push Notifications.

If you plan to use services such as Game Center, In-App Purchase, and Push Notifications, or want a Bundle ID unique to a single app, use an explicit App ID. If you want to create one provisioning profile for multiple apps or don't need a specific Bundle ID, select a wildcard App ID. Wildcard App IDs use an asterisk (*) as the last digit in the Bundle ID field. Please note that iOS App IDs and Mac App IDs cannot be used interchangeably.

App ID: Rhyschrimme (.ch.romanrast.rhyschrimme)

Mobile Workshop FHNW 2018 - Ionic

The screenshots show the process of creating an iOS Provisioning Profile (Distribution type) using the Xcode developer portal.

Top Screenshot: Select certificates.

The sidebar on the left shows categories: Certificates, Identifiers, Devices, and Provisioning Profiles. Under Provisioning Profiles, the "Distribution" tab is selected. The main panel shows a list of certificates:

- Roman Rast (iOS Distribution) Jun 08, 2017 (unchecked)
- Roman Rast (iOS Distribution) Apr 21, 2018 (checked)

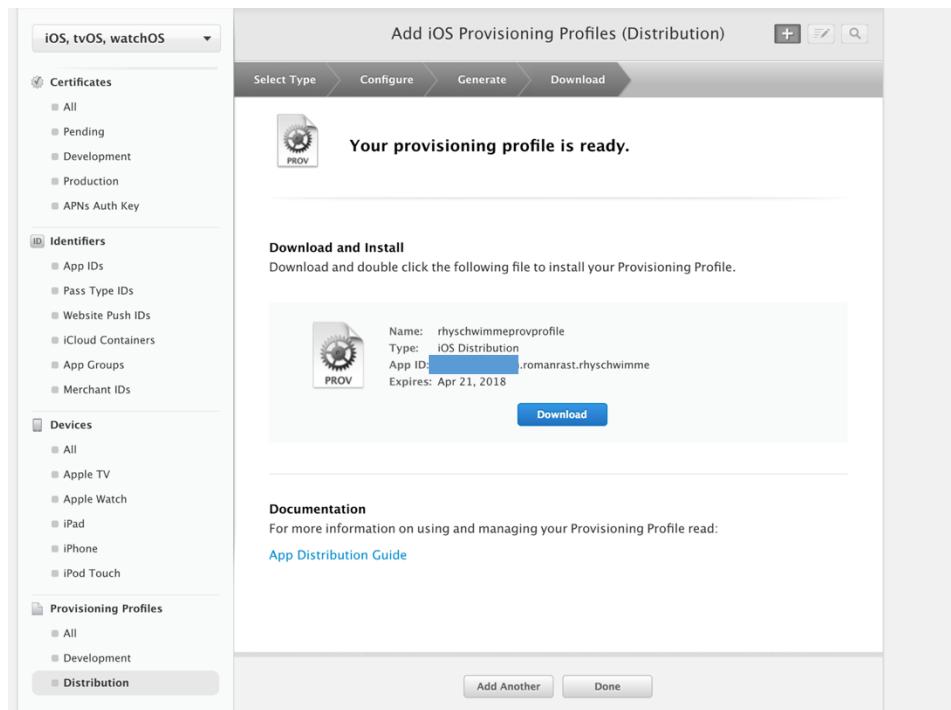
Bottom Screenshot: Name this profile and generate.

The sidebar on the left shows the same categories and selection for the "Distribution" tab. The main panel displays the configuration for the new profile:

- Profile Name: **rhyhschwimmeprofile**
- Type: **iOS Distribution**
- App ID: Rhyschwimme (ch.romanrast.rhyhschwimme)
- Certificates: **1 Included**

Buttons at the bottom include Cancel, Back, and Continue.

Mobile Workshop FHNW 2018 - Ionic

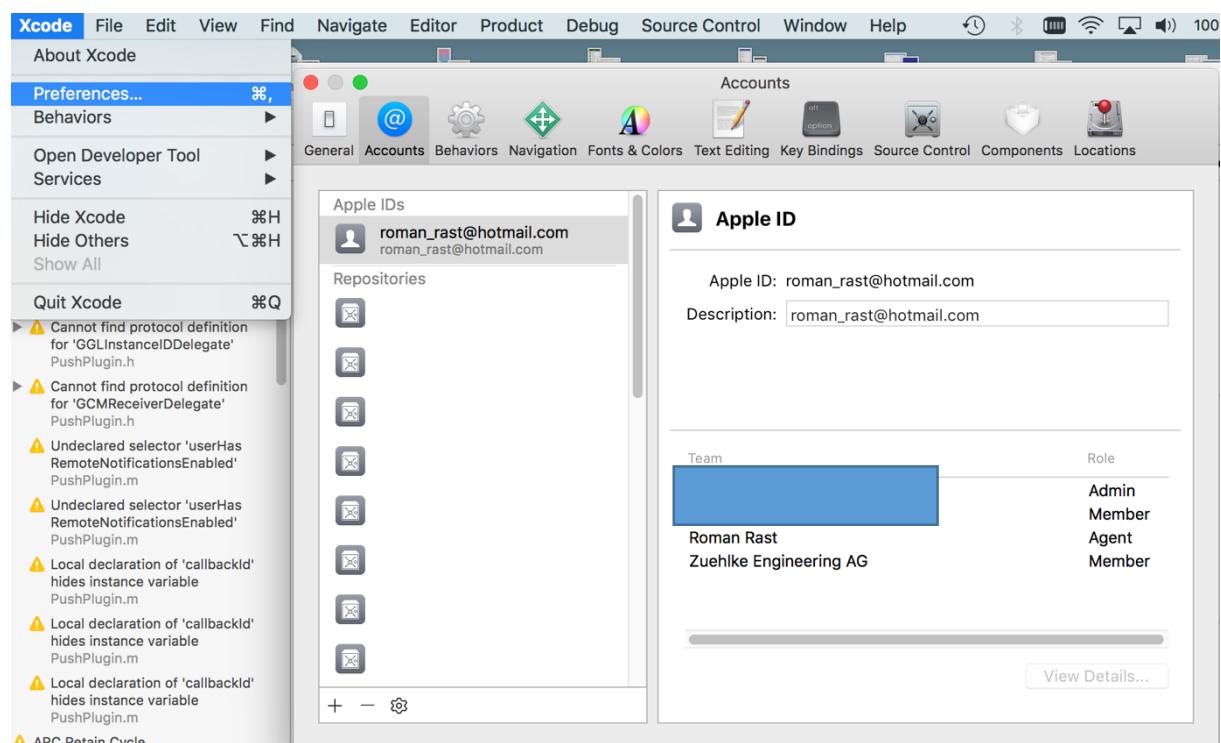


Jetzt haben wir alle Zertifikate, die wir benötigen (bewahre diese und die Passwörter gut auf!).

- **xxx.certSigningRequest**
- **xxx.cer**
- **xxx.p12**
- **xxx.mobileprovision**

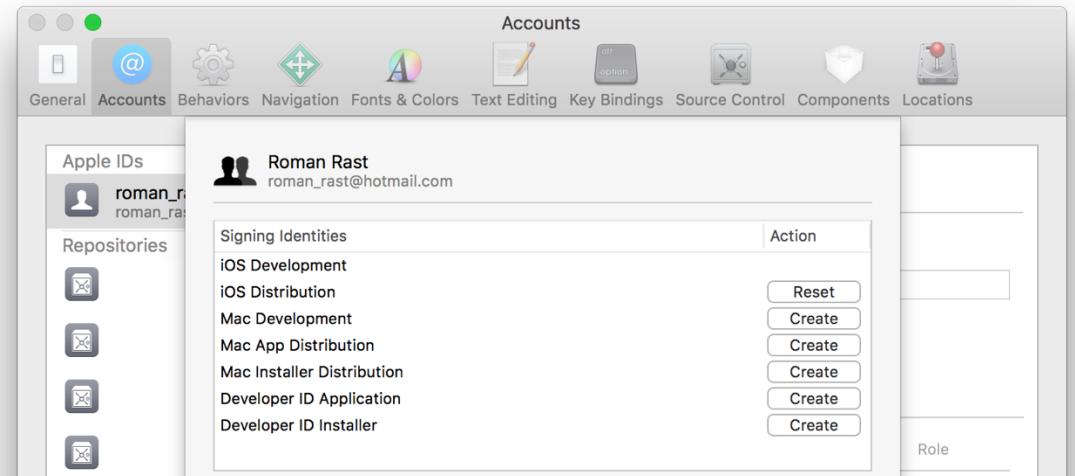
Release für App Store

Öffne Xcode und gehe unter Xcode > Preferences > Deine Apple ID > Dein Team > View Details



Mobile Workshop FHNW 2018 - Ionic

Danach unter Signing Identities klicke bei iOS Distribution auf Create (Falls nicht schon vorhanden)



Als nächster Schritt müssen wir in iTunes Connect einen Platz bzw. das Store Listing für unsere App anlegen. Hier werden wir auch den Build unserer App hochladen sowie die Screenshots und die Beschreibung der App.

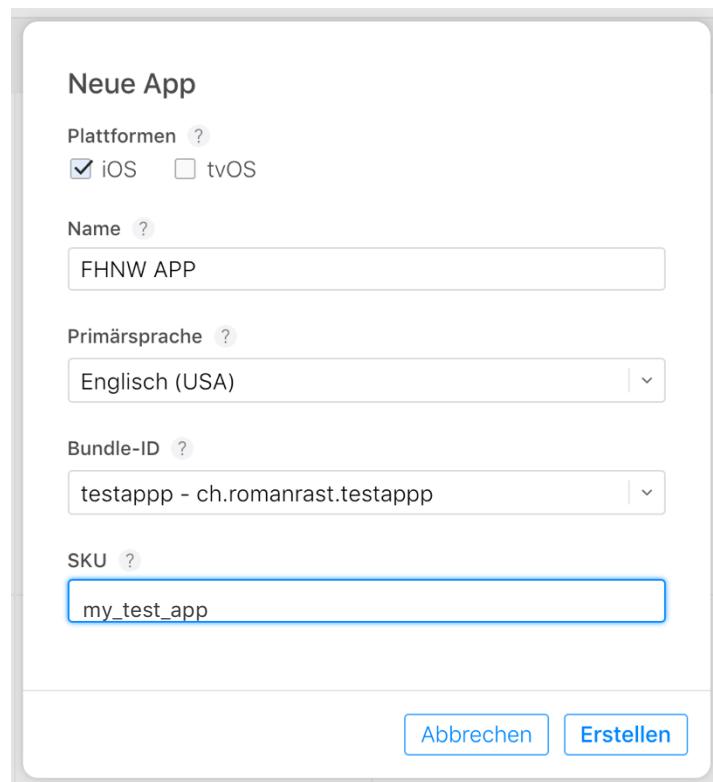
Gehe dazu auf <https://itunesconnect.apple.com/login>

> Meine Apps > Klicke auf das Plus > Neue App

A screenshot of the iTunes Connect 'Meine Apps' page. At the top, there is a search bar and a 'Meine Apps' dropdown. Below the search bar, there is a '+' button and three dots. A modal dialog box is open over the list of apps, with 'Neue App' highlighted. The list of apps includes: Rhyschwimme Basel (status: iOS 1.0 Warten auf Prüfung..), Sticky Sticker (status: iOS 1.0 Bereit zum Verkauf), and Hello Cat (status: iOS 1.2 Bereit zum Verkauf). Each app has a small thumbnail icon.

Wähle iOS bei den Plattformen an und fülle die Angaben aus. Name -> Wie der App Name im Store angezeigt werden soll. Bundle-ID ist dieselbe, welche du in deinem Xcode Projekt findest. SKU ist eine eindeutige ID, setze hier einen Namen, dieser wird nicht im App Store angezeigt.

Mobile Workshop FHNW 2018 - Ionic

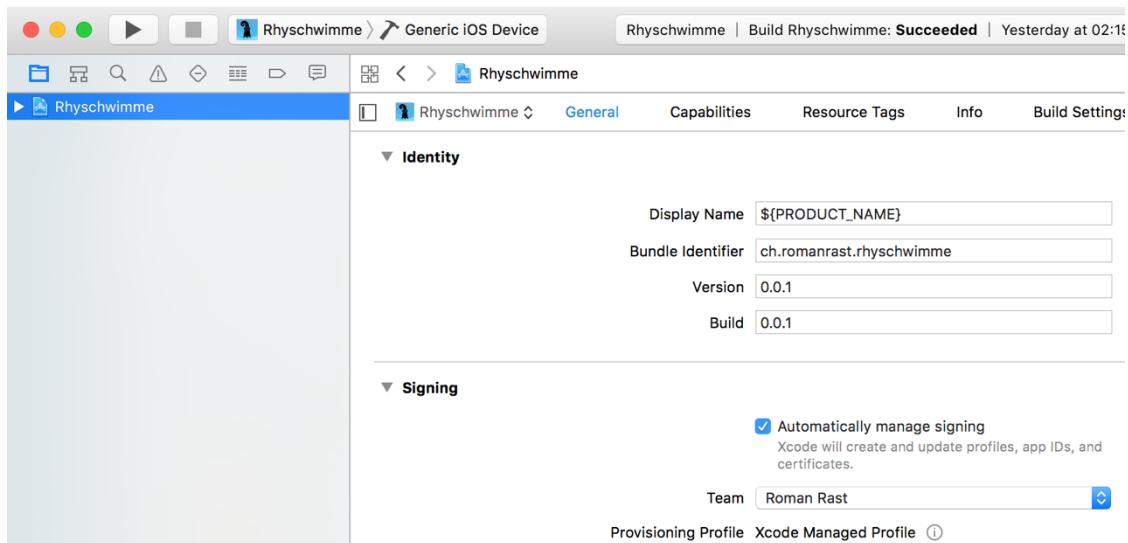


Nun solltest du auf den App-Informationen Screen kommen. Hier kannst du vorerst alle Informationen angeben.

Als nächstes können wir jetzt den Build deiner App erstellen und in iTunes Connect hochladen. Öffne dazu dein Terminal im Root deiner App und gib folgendes ein:

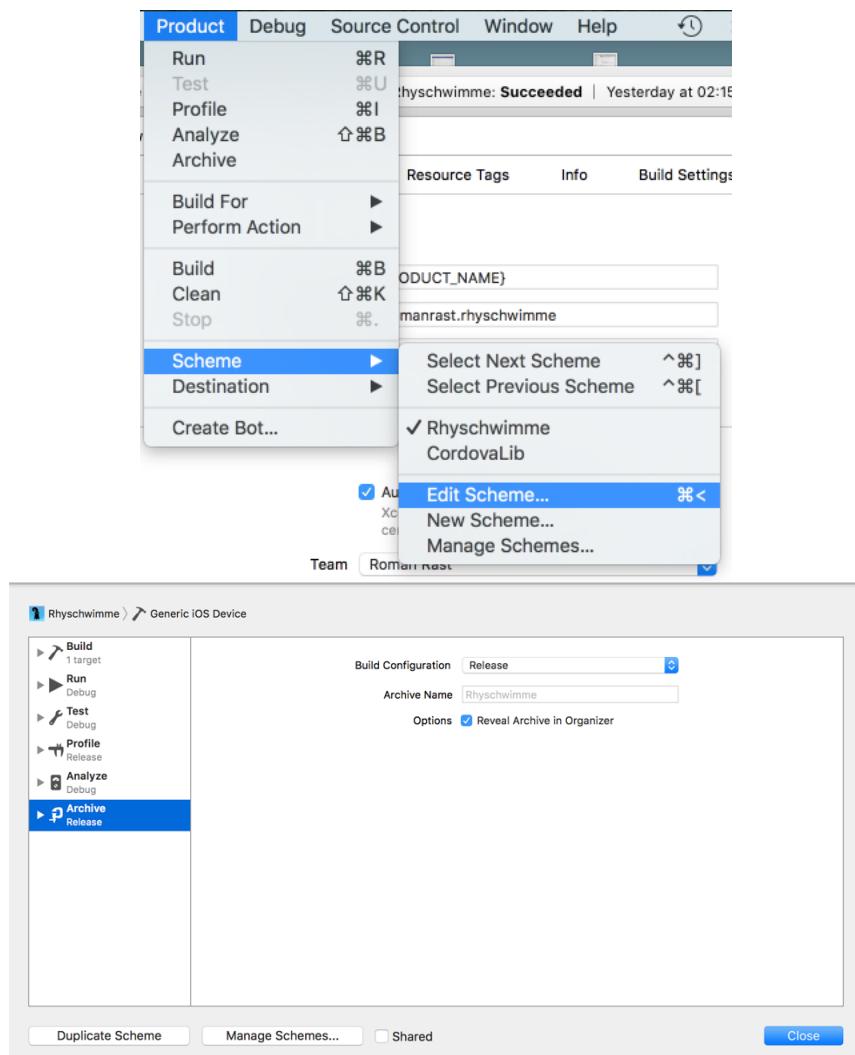
```
cordova build ios --release
```

Der Build sollte mit BUILD SUCCEEDED durchlaufen. Öffne nun dein xxx.xcodeproj deiner App unter Deine App > platforms > ios vergewissere dich, dass alle Angaben unter General korrekt sind und Bundle Identifier mit der Bundle-ID in iTunes Connect übereinstimmen. Falls nicht, kannst du diese im config.xml deines Ionic Projektes noch anpassen und nochmals bauen.

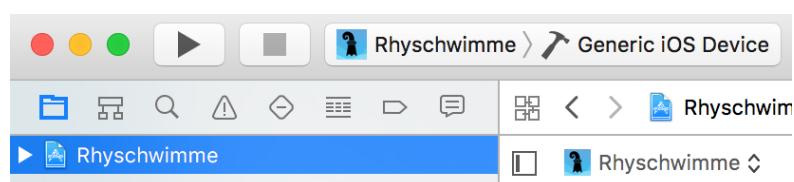


Mobile Workshop FHNW 2018 - Ionic

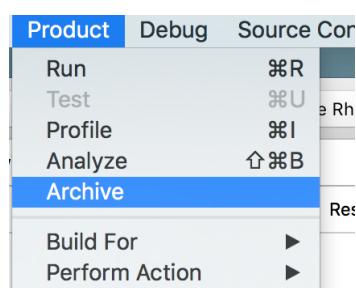
Unter Product > Scheme > Edit Scheme > Archive setze die Build Configuration auf Release



Jetzt können wir das Archive erstellen. Schliesse das Fenster und setze im Hauptfenster die Auswahl des Devices auf **Generic iOS Device**.



Danach auf Product > Archive > Wähle deine App aus > Upload to App Store



Mobile Workshop FHNW 2018 - Ionic

Name	Creation Date	Version	Archive Information
Rhyschwimme	24 Apr 2017, 11:35	0.0.1 (0.0.1)	Rhyschwimme 24 Apr 2017, 11:35
Rhyschwimme	23 Apr 2017, 00:20	0.0.1 (0.0.1)	

Upload to App Store...

Nun wieder zurück in iTunes Connect, kannst du die restlichen Infos angeben wie Screenshots, App Icon, Search Keywords etc. Ebenfalls wirst du jetzt den Build auswählen können.

Für Screenshots ist es am einfachsten den iOS Simulator als iPhone 8Plus (5.5") und als iPad Pro (12.9") zu starten. Mit cmd+s kannst du Screenshots erstellen. Diese werden auf deinem Desktop abgelegt. Die restlichen Screens für kleinere Devices werden runter gerechnet anhand der beiden Größen.

Nun gilt es nur noch abzuwarten, bis die App veröffentlicht ist. Üblicherweise geht die Prüfung folgende Schritte durch: Prepare For Review > Waiting For Review > In Review > Ready for Sale. Unter <http://appreviewtimes.com/> kannst du die durchschnittliche Dauer des Reviewprozesses anschauen (meistens im Bereich von zwei Tagen).

Android: Release für Google Play Store

Für Android ist der Prozess einiges einfacher.

Stelle zuerst sicher, dass du gradle installiert hast -> in der Konsole gradle -v sollte dir die Version zurückgeben, falls installiert, sonst mit:

```
brew install gradle
```

Homebrew kannst du via Konsole installieren, gib dazu folgendes ein:

```
/usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Nun können wir den release Build erstellen. In der Konsole im Root der App gib folgendes ein:

```
cordova build --release android
```

Dies generiert das unsigned .apk File unter platforms/android/build/outputs/apk/**android-release-unsigned.apk**.

Android: Android Keystore

Für Android brauchen wir einen Signing Key (den Android Keystore) um Apps zu signieren. Diesen kannst du in der Konsole im Root deiner App mit folgendem Befehl erstellen (ersetze dabei my-release-key und my_alias mit z.B. dem Namen deiner App -> rhyschwimme-release-key). Du musst noch zum Keystore und Key ein Passwort angeben. Merke die Passwörter und den Alias, wir werden diese später noch brauchen!

```
keytool -genkey -v -keystore my-release-key.keystore -alias my_alias -  
keyalg RSA -keysize 2048 -validity 10000
```

Im Root Ordner deines Projektes erscheint nun eine **my-release-key.keystore** Datei. Speichere dieses File an einem sicheren Ort, du wirst dieses File brauchen um weitere Updates der App in den Store stellen zu können!

Signieren des apk

Wir können nun das apk signieren mit:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-  
release-key.keystore android-release-unsigned.apk my_alias
```

Ersetze dabei **my-release-key.keystore** mit dem Keystore Namen, welchen du schon gesetzt hast und gib das Passwort dazu ein.

apk Optimieren - zipalign tool

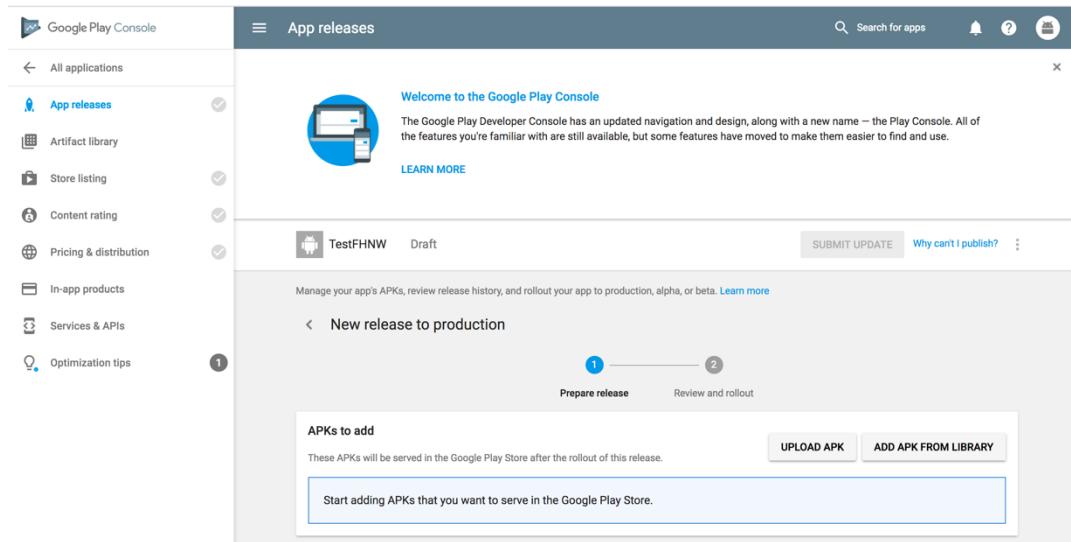
Als nächstes müssen wir das apk optimieren mit dem zipalign Tool (verringert den RAM verbrauch der App). Das zipalign Tool sollte sich auf Mac unter ~/Library/Android/sdk/build-tools/VERSION/zipalign befinden. Falls nicht, schau in Android Studio nach, ob die nötigen SDKs installiert sind > Tools > Android > SDK Manager > Launch Standalone SDK Manager > Android SDK Build-tools die nötigen Versionen installieren. Danach zipalign in den richtigen Pfad kopieren, z.B. /usr/local/Cellar/android-sdk/24.1.2/build-tools/21.1.2/zipalign.

Nun kannst du folgenden Befehl ausführen:

Mobile Workshop FHNW 2018 - Ionic

```
zipalign -v 4 android-release-unsigned.apk Rhyschwimme.apk
```

Jetzt können wir auf der Google Play Console <https://play.google.com/apps/publish/> unsere App hochladen > Create Application > App releases > Manage Production > Upload APK.



Die restlichen Infos ausfüllen und deine App ist in wenigen Stunden im Store!

