# App Name: GradeMate + Studify + SmartGrades

Inhaltsverzeichnis

[App Name: GradeMate + Studify 1](#_Toc181526261)

[Programmiersprache 2](#_Toc181526262)

[Für den App-Release zu beachten: 2](#_Toc181526263)

[Android App Release 3](#_Toc181526264)

[iOS App Release 5](#_Toc181526265)

[Finales Testing und Veröffentlichung 6](#_Toc181526266)

# Programmiersprache

**React Native mit TypeScript**: Diese Kombination bietet dir die Möglichkeit, den Großteil des Codes plattformübergreifend zu nutzen und die Vorteile von TypeScript wie Typensicherheit und besseres Code-Management einzubeziehen. React Native ist speziell für Cross-Plattform-Apps optimiert und unterstützt gängige mobile Funktionen wie Lokalspeicherung und UI-Komponenten, was ideal für eine App ohne Server-Anbindung ist.

# Für den App-Release zu beachten:

1. **Entwickler-Accounts**:
   * **Apple Developer Account** für iOS (kostet jährlich 99 USD).
   * **Google Play Developer Account** für Android (einmalig 25 USD).
2. **App-Signierung und -Zertifikate**:
   * **iOS**: App muss signiert und mit einem Zertifikat versehen sein, welches du über den Apple Developer Account erhältst.
   * **Android**: Benötigt ein „Keystore“ zur Signierung der App.
3. **Testen auf beiden Plattformen**:
   * Um sicherzustellen, dass die App auf verschiedenen Geräten und Betriebssystemversionen reibungslos funktioniert, empfiehlt sich gründliches Testen, idealerweise auf realen Geräten.
4. **Einhaltung von Richtlinien**:
   * Stelle sicher, dass die App den Richtlinien des **Apple App Store** und **Google Play Store** entspricht, z. B. in Bezug auf Datenschutz, Benutzererfahrung und Inhalte.
5. **Icon, Screenshots und Beschreibung**:
   * Diese sind für den App-Store-Eintrag erforderlich und sollten klar und ansprechend gestaltet sein.
6. **Lokale Speicherung und Datenschutz**:
   * Auch ohne Server-Verbindung ist es wichtig, sicherzustellen, dass die lokal gespeicherten Daten sicher sind und ggf. DSGVO-konform gehandhabt werden.
7. **Beta-Testing**:
   * Verwende Plattformen wie **TestFlight** für iOS und **Google Play Console** für Android, um die App vor dem Release mit Beta-Testern zu teilen.

# Android App Release

**Schritt 1: App-Signierung einrichten**

* Öffne das Projekt in **Android Studio**.
* Erstelle ein neues Keystore-File:
  + Build > Generate Signed Bundle / APK > APK
  + Erstelle einen neuen Keystore, setze ein Passwort und sichere es ab.
* Füge die Keystore-Informationen in die android/app/build.gradle Datei ein:

gradle

Copy code

android {

...

signingConfigs {

release {

storeFile file('path/to/your/keystore/file')

storePassword 'your-password'

keyAlias 'your-key-alias'

keyPassword 'your-key-password'

}

}

buildTypes {

release {

signingConfig signingConfigs.release

minifyEnabled true // optional, für Code-Minimierung

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

}

}

}

**Schritt 2: Release-Build erstellen**

* Führe den Befehl aus, um die APK zu erstellen:

Copy code

cd android

./gradlew assembleRelease

* Nach erfolgreichem Build findest du die signierte APK-Datei im Verzeichnis android/app/build/outputs/apk/release.

**Schritt 3: APK auf Google Play hochladen**

* Melde dich in der **Google Play Console** an und erstelle ein neues App-Projekt.
* Lade die APK-Datei hoch, gib die nötigen Informationen an (Beschreibung, Screenshots, Altersfreigabe) und reiche sie zur Überprüfung ein.

# iOS App Release

**Schritt 1: Projekt konfigurieren**

* Öffne das Projekt in **Xcode**.
* Navigiere zu **Project > Signing & Capabilities** und stelle sicher, dass das richtige Team und die Bundle-ID ausgewählt sind.
* Gehe zu **Build Settings** und setze den Build-Modus auf „Release“.

**Schritt 2: Release-Build erstellen**

* Wähle im Xcode-Menü „Product > Archive“.
* Warte, bis der Build-Prozess abgeschlossen ist. Xcode zeigt dann das Archiv an.
* Klicke auf „Distribute App“ und wähle „App Store Connect“.
* Folge den Anweisungen, um die App zu signieren und zu verifizieren.

**Schritt 3: App im App Store Connect hochladen**

* Nach erfolgreichem Upload zur **App Store Connect** Website navigieren.
* Erstelle einen neuen App-Eintrag, fülle die notwendigen Informationen aus und lade Screenshots hoch.
* Reiche die App zur Überprüfung ein.

# Finales Testing und Veröffentlichung

1. **TestFlight (iOS)**: Verwende TestFlight für iOS, um die App mit Beta-Testern zu teilen und letzte Fehler zu finden.
2. **Google Play Beta (Android)**: Google Play bietet ebenfalls die Möglichkeit, Beta-Tests vor dem offiziellen Release durchzuführen.

Sobald die Überprüfung abgeschlossen ist, wird die App im jeweiligen Store veröffentlicht.



**CI/CD**



**Database**

**DB-Zugriff**

**Frontend**

