

BEZAHLTE APPS NACHBAUEN MIT CLAUDE CODE UND REA

15 PROMPTS

TOOL: CLAUDE CODE

STAND 2026-10-16

Du lädst die App runter, gibst sie REA und das Ding zerlegt sie dir komplett. Danach weiß Claude, wie sie funktioniert, und baut dir daraus deine eigene, mit deinen eigenen Wünschen, einfach auf Deutsch. Das Herzstück ist die Tabelle mit den App-Arten: Sie zeigt dir, was du für Desktop-Apps, Web-Apps, Android-Apps und native Programme zusätzlich brauchst, und zu jeder Art bekommst du einen fertigen Prompt. Am Ende bringst du deine eigene App ins Web oder in den App Store.

0 €

FÜR REA

5

APP-ARTEN MIT EIGENEM PROMPT

1 Prompt

PRO ANALYSE

WAS DU BRAUCHST

Was du brauchst

Claude Pro oder höher: rund 21 € im Monat bei monatlicher Zahlung (Max kostet rund 107 € oder 214 €). Claude Code steckt in allen bezahlten Plänen, im kostenlosen Plan fehlt es.

Die Claude Desktop-App von claude.com/download, für Mac oder Windows. Darin steckt Claude Code: ein Claude, das direkt auf deinem Rechner arbeitet, Programme installiert und Dateien bearbeitet, ohne dass du selbst ein Terminal öffnen musst. Für den App Store am Ende brauchst du einen Mac.

Node.js 22.19 oder neuer, kostenlos. Das ist die Laufzeit, auf der REA läuft. Claude prüft und installiert es in Schritt 2 für dich.

Die App, die du nachbauen willst: installiert auf deinem Rechner, als Datei oder als Website.

Je nach App-Art ein Zusatzprogramm, siehe Tabelle. Alles kostenlos. Optional: eine Hopper-Lizenz für 99 € einmalig, nur für native Mac-Apps.

Nur für den App Store: das Apple Developer Program für 99 € im Jahr. Den genauen Betrag zeigt Apple dir bei der Anmeldung.

WAS DU PRO APP-ART BRAUCHST

	WAS DU REA GIBST	ZUSÄTZLICH INSTALLIEREN	KOSTEN
Desktop-App mit Electron (z. B. Notion, Slack, Superhuman, Linear, Obsidian)	die installierte App	nichts	0 €
Web-App im Browser	die Seite in Google Chrome	Google Chrome	0 €
Android-App	die APK-Datei	Java (JDK 21) und die JADX-Datei	0 €
Native Mac-App	die installierte App	Hopper	0 € mit der Demo, Lizenz optional 99 €
Native Windows-Programm (.exe)	die Programmdatei	Ghidra 12.1 und Java (JDK 21)	0 €

WELCHE APP-ART DU HAST

Bei Apps, die es nur im iPhone App Store gibt, geht das aktuell nicht. Bei Android oder im Web schon. Für Android installiert Claude dir in Schritt 6 zwei kostenlose Zusatzprogramme. Desktop-Apps auf Mac und Windows kannst du ebenfalls analysieren.

Electron heißt: Die Desktop-App ist innen eine Website, die in ein eigenes Fenster verpackt ist. Das ist der einfachste Fall. Welche Art deine App ist, findet Claude in Schritt 3 für dich heraus.

WENN DAS NICHT KLAPPT

Gibt es die App, die du nachbauen willst, auch als Web-Version oder als Desktop-App für Mac oder Windows, nimm diese: Weg A für die Desktop-App, Weg B für die Web-Version. Damit kommst du an dieselben Funktionen.

01

CLAUDE CODE ÖFFNEN

Leg auf deinem Schreibtisch einen leeren Ordner **App-Nachbau** an. Darin landen alle Analysen und später deine eigene App.

1. Öffne die Claude Desktop-App und klick oben auf den Tab **Code**.
2. Stell als Umgebung **Local** ein. Claude arbeitet dann auf deinem Rechner.
3. Wähl als Projektordner deinen Ordner **App-Nachbau**.

Wenn Claude im Lauf des Guides um Erlaubnis fragt, etwas zu installieren oder eine Datei zu ändern, lies kurz, worum es geht, und bestätige.

REA INSTALLIEREN

REA ist kostenlos und Open Source. Es ist ein MCP-Server, also ein Stecker zwischen Claude und deinen Apps: Darüber kann Claude Apps untersuchen, ohne ihren Quellcode zu haben. Kopier den Prompt in den Claude Code Chat. Claude prüft Node.js, installiert es bei Bedarf und richtet REA ein.

Danach musst du Claude einmal komplett neu starten, sonst sieht Claude die neuen Werkzeuge nicht. Auf dem Mac mit **Cmd+Q** beenden und neu öffnen, unter Windows das Fenster schließen und die App neu starten. Öffne danach wieder den Code-Tab im Ordner App-Nachbau.

TIPP

Nach dem Neustart schreibst du: „Welche REA-Werkzeuge hast du gerade?“ Claude zählt dir dann Werkzeuge wie `analyze_javascript_application` oder `open_binary` auf. Dann ist REA verbunden.

REA EINRICHTEN

Richte mir REA ein, das Tool zum Analysieren von Apps. Nimm nur das offizielle Paket `rea-agents` vom Repo github.com/morluto/rea.

1. Prüf, ob Node.js 22.19 oder neuer installiert ist. Wenn nicht, installier die aktuelle LTS-Version von nodejs.org in meinen Benutzerordner, ohne Admin-Rechte.
2. Führe dann aus: `npx -y rea-agents@6.5.0 setup --client claude_code --yes`
3. Führe danach aus: `npx -y rea-agents@6.5.0 doctor --client claude_code --json` und erkläre mir das Ergebnis in einfachen Worten.
4. Sag mir am Ende, ob alles geklappt hat und dass ich Claude jetzt neu starten muss.

WENN DAS NICHT KLAPPT

Falls Claude nach dem Neustart keine REA-Werkzeuge findet, schreibe: „Führe `npx -y rea-agents@6.5.0 doctor --client claude_code --json` aus, repariere die Registrierung für Claude Code und sag mir, ob ich neu starten muss.“ Hilft das nicht, beende die Claude Desktop-App komplett, öffne sie neu und starte im Code-Tab eine neue Session.

HERAUSFINDEN, WELCHE ART APP DU HAST

Bevor du etwas zusätzlich installierst, lässt du Claude nachsehen, welcher Weg aus diesem Guide zu deiner App passt. Ersetz im Prompt die eckigen Klammern durch den Namen deiner App.

APP-ART BESTIMMEN

Ich will die App [Name der App] mit REA analysieren. Find heraus, ob sie auf meinem Rechner installiert ist und welche Art App das ist:

- Electron (prüf, ob im App-Paket eine Datei `app.asar` liegt)
- native Mac-App oder natives Windows-Programm
- oder ob es sie nur im Web gibt.

Sag mir, welchen Weg ich nehmen soll: A (Electron), B (Web), C (Android), D (native Mac-App) oder E (Windows-Programm), und was ich dafür noch installieren muss.

WEG A: DESKTOP-APP MIT ELECTRON ANALYSIEREN

Für Apps wie Notion, Slack, Superhuman, Linear oder Obsidian.

Das ist der einfachste Weg, du brauchst nichts zusätzlich. Der Code der Oberfläche steckt in einer Datei namens **app.asar**, das ist das Paket, in das Electron die App verpackt. Auf dem Mac liegt sie im App-Paket unter **Contents/Resources/app.asar**, unter Windows im Installationsordner unter **resources\app.asar**. Claude findet sie selbst.

REA führt dabei nichts von der App aus, es liest die Datei nur. Je nach Größe der App dauert die Analyse eine bis ein paar Minuten.

TIPP

Manche Desktop-Apps laden einen großen Teil ihrer Oberfläche aus dem Web nach. Findet REA in der Datei wenig über die Funktion, die du willst, nimm zusätzlich Weg B mit der Web-Version derselben App.

ELECTRON-APP ANALYSIEREN

Analysier mit REA die App [Name der App].

1. Nimm die Datei app.asar im App-Paket (auf dem Mac: /Applications/[Name der App].app/Contents/Resources/app.asar) und ruf `analyze_javascript_application` mit `detail summary` auf.
2. Meldet REA den Fehler `artifact_integrity_mismatch`, wiederhol den Aufruf mit `integrity_policy record-and-continue`.
3. Erklär mir danach in einfachen Worten: welche Fenster und Funktionen die App hat, wie die Oberfläche aufgebaut ist, mit welchen Servern sie spricht und welche Teile nur online laufen.
4. Schreib alles in die Datei `analyse.md` in diesem Ordner.

WEG B: WEB-APP ANALYSIEREN

REA beobachtet eine Seite, während du sie benutzt. Dafür braucht es ein eigenes Chrome-Fenster mit einem eigenen, leeren Profil. Seit Chrome 136 lässt sich dein normales Chrome-Profil nicht mehr so anzapfen, und das ist auch gut so: Deine gespeicherten Passwörter und Cookies bleiben außen vor.

1. Schick den Prompt. Claude öffnet ein neues Chrome-Fenster.
2. Log dich in diesem Fenster selbst bei der Web-App ein. Dein Passwort tippst du selbst, nie Claude.
3. Schreib Claude „Bin eingeloggt“. Dann hast du 60 Sekunden, in denen du die Funktion durchklickst, die du nachbauen willst.

REA schaut dabei nur zu. Passwörter, Cookie-Werte und gespeicherte Daten nimmt es nicht in die Analyse auf.

ACHTUNG

Schließ das zweite Chrome-Fenster, wenn du fertig bist. Solange es offen ist, können Programme auf deinem Rechner darauf zugreifen. Der Port 9222 ist nur auf deinem eigenen Rechner erreichbar, nicht aus dem Internet.

WEB-APP ANALYSIEREN

Ich will die Web-App [Adresse der Seite] mit REA analysieren.

1. Öffne Google Chrome als neues Fenster mit einem eigenen, neuen Profil im Ordner `rea-chrome-profil` in meinem Benutzerordner, mit `--remote-debugging-port=9222`, und lade darin [Adresse der Seite]. Nutz nie mein normales Chrome-Profil.
2. Warte, bis ich mich eingeloggt habe und Bescheid gebe.
3. Hol dir mit `list_browser_targets` die Seite und starte `observe_web_session` für 60 Sekunden. In der Zeit klicke ich die Funktion durch, die du nachbauen sollst.
4. Erklär mir danach, wie die Seite aufgebaut ist, welche Schritte im Hintergrund passieren und welche Daten vom Server kommen.
5. Schreib alles in die Datei `analyse-web.md` in diesem Ordner.

WEG C: ANDROID-APP ANALYSIEREN

Für Android braucht REA zwei kostenlose Zusatzprogramme:

Java, genauer ein volles JDK in Version 17 oder neuer. Wir nehmen das JDK 21 von Eclipse Temurin, mit genau dieser Version ist REA getestet.

JADX als Datei `jadx-headless-mcp-0.7.1-all.jar`. JADX übersetzt den Code einer Android-App zurück in lesbaren Java-Code.

Beides installiert Claude in einen Ordner **rea-tools** in deinem Benutzerordner, ohne Admin-Rechte, und trägt die Pfade in die REA-Einstellungen von Claude Code ein. Danach startest du Claude neu.

ANDROID-ANALYSE EINRICHTEN

Richte die Android-Analyse für REA ein, ohne Admin-Rechte:

1. Prüf, ob ein volles JDK 17 oder neuer installiert ist. Wenn nicht, lad das JDK 21 von Eclipse Temurin passend zu meinem System über [https://api.adoptium.net/v3/binary/latest/21/ga/\[System\]/\[Prozessor\]/jdk/hotspot/normal/eclipse](https://api.adoptium.net/v3/binary/latest/21/ga/[System]/[Prozessor]/jdk/hotspot/normal/eclipse) herunter (System: mac oder windows, Prozessor: aarch64 oder x64) und entpack es in den Ordner `rea-tools` in meinem Benutzerordner. Auf dem Mac ist `JAVA_HOME` dann der Unterordner `Contents/Home`.
2. Lad <https://github.com/1013503897/jadx-headless-mcp/releases/download/v0.7.1/jadx-headless-mcp-0.7.1-all.jar> in denselben Ordner und prüf die SHA-256-Prüfsumme. Sie muss `6e5eacf500b64292bfb73c49797c1958f6ee44646e43e868039ae7feb573ff75` sein. Stimmt sie nicht, lösche die Datei und hör auf.
3. Mach eine Sicherheitskopie der Datei `.claude.json` in meinem Benutzerordner. Trag dann im Eintrag `mcpServers.rea` unter `env` zwei Werte ein: `REA_JADX_MCP_JAR` mit dem absoluten Pfad zur JAR-Datei und `JAVA_HOME` mit dem Pfad zum JDK. Lass alles andere in der Datei unverändert.
4. Setz dieselben zwei Werte als Umgebungsvariablen und prüf mit `npx -y rea-agents@6.5.0 doctor --provider jadx --json`, ob JADX bereit ist.
5. Sag mir, ob alles geklappt hat und dass ich Claude jetzt neu starten muss.

DIE APK-DATEI BESORGEN

Die APK ist die Installationsdatei einer Android-App. Am sichersten holst du sie dir von deinem eigenen Android-Handy, auf dem die App schon installiert ist. Lad APKs nicht von fremden Download-Seiten, dort steckt oft Schadsoftware drin.

Am Handy einmal einschalten:

1. Öffne **Einstellungen > Über das Telefon** und tipp siebenmal auf **Build-Nummer**, bis „Du bist jetzt Entwickler“ erscheint. Bei manchen Herstellern liegt die Build-Nummer in einem Untermenü wie **Softwareinformationen**.
2. Geh zurück und öffne **Einstellungen > System > Entwickleroptionen**. Schalte dort **USB-Debugging** ein.
3. Verbinde das Handy per Kabel mit deinem Rechner. Wenn das Handy fragt, ob es USB-Debugging zulassen soll, tipp auf **Zulassen**.

Schalte USB-Debugging wieder aus, wenn du fertig bist.

APK VOM HANDY HOLEN

Hol mir die APK der App [Name der App] von meinem Android-Handy. Das Handy ist per Kabel verbunden und USB-Debugging ist an.

1. Lad die Android SDK Platform Tools von developer.android.com/tools/releases/platform-tools in den Ordner `rea-tools` in meinem Benutzerordner, ohne Admin-Rechte.
2. Such mit `adb shell pm list packages` das Paket der App.
3. Hol mit `adb shell pm path` den Speicherort und kopier die `base.apk` mit `adb pull` in diesen Ordner.
4. Sag mir, wie die Datei heißt und wo sie liegt.

DIE ANDROID-APP ANALYSIEREN

REA startet die App dabei nicht und braucht keinen Emulator. Es liest die APK-Datei und zeigt dir den zurückübersetzten Code. Ersetz im Prompt die Funktion durch das, was du nachbauen willst, zum Beispiel „das Anlegen einer neuen Aufgabe“.

ANDROID-APP ANALYSIEREN

Analysier mit REA die Datei `base.apk` in diesem Ordner.

1. Starte mit `inspect_android_package` und erklär mir, was die App ist und welche Berechtigungen sie verlangt.
2. Such mit `search_android_classes` die Klassen zur Funktion [Funktion] und schau dir die wichtigsten mit `inspect_android_class` an.
3. Dekompilier die entscheidenden Methoden mit `inspect_android_method`.
4. Erklär mir in einfachen Worten, wie die Funktion abläuft und welche Daten sie an Server schickt.
5. Schreib alles in die Datei `analyse.md` in diesem Ordner.

WEG D: NATIVE MAC-APP ANALYSIEREN

Native Mac-Apps sind direkt für den Mac geschrieben, meist in Swift, und haben keine app.asar. Für sie braucht REA ein Analyse-Programm. Auf dem Mac nimmst du **Hopper**: REA installiert es selbst, ohne Admin-Rechte, in den Ordner Programme in deinem Benutzerordner.

1. Schick den ersten Prompt. REA lädt Hopper vom Hersteller und öffnet es.
2. Hopper fragt dich beim ersten Start, ob du den Demo-Modus nutzen oder eine Lizenz aktivieren willst. Wähl den **Demo-Modus**. Das musst du selbst klicken.
3. Starte Claude komplett neu und schick den zweiten Prompt.

Was REA dir zeigt, ist Pseudocode: ein zurückübersetzter Code, der zeigt, was das Programm tut, nicht der Originalcode des Herstellers.

TIPP

Fragt macOS, ob Claude oder REA Hopper steuern darf, erlaub es. Falls die Frage nicht kommt und REA Hopper nicht erreicht: **Systemeinstellungen > Datenschutz & Sicherheit > Automation** öffnen und dort den Zugriff einschalten.

HOPPER INSTALLIEREN

```
Installier Hopper für REA. Führe dafür aus: npx -y rea-agents@6.5.0 setup --client  
claude_code --yes --install-hopper
```

Sag mir danach, was ich in Hopper klicken muss, und dass ich Claude neu starten soll.

NATIVE MAC-APP ANALYSIEREN

MAC-APP ANALYSIEREN

Analysiere mit REA die App `/Applications/[Name der App].app`.

1. Öffne das Programm mit `open_binary` und verschaff dir mit `binary_overview` einen Überblick.
2. Suche gezielt nach den Funktionen, die zu [Funktion] gehören, und schau dir die wichtigsten mit Dekompilierung, Aufrufern und Querverweisen an.
3. Erkläre mir in einfachen Worten, wie die Funktion abläuft.
4. Schreibe alles in die Datei `analyse.md` in diesem Ordner und schließe die Analyse am Ende mit `close_binary`.

WEG E: NATIVES WINDOWS-PROGRAMM ANALYSIEREN

Nur auf einem Windows-PC mit 64-Bit-System.

Für Windows-Programme ohne Electron nutzt REA **Ghidra**, ein kostenloses Analyse-Programm, dazu Java (JDK 21). REA kennzeichnet die Windows-Unterstützung mit Ghidra noch als experimentell und nur zum Lesen. Hopper gibt es für Windows nicht.

WINDOWS-PROGRAMM ANALYSIEREN

Richte Ghidra für REA ein, ohne Admin-Rechte:

1. Lad Ghidra 12.1.4 (Datei `ghidra_12.1.4_PUBLIC_20260921.zip`) von github.com/NationalSecurityAgency/ghidra/releases und das JDK 21 von Eclipse Temurin für Windows x64 in den Ordner `rea-tools` in meinem Benutzerordner und entpack beides.
2. Mach eine Sicherheitskopie der Datei `.claude.json` in meinem Benutzerordner. Trag im Eintrag `mcpServers.rea` unter `env` drei Werte ein: `GHIDRA_INSTALL_DIR` mit dem Ghidra-Ordner, `JAVA_HOME` mit dem JDK-Ordner und `REA_ANALYSIS_PROVIDER` mit dem Wert `ghidra`. Lass alles andere unverändert.
3. Prüf mit `npx -y rea-agents@6.5.0 doctor --provider ghidra --json`, ob Ghidra bereit ist.
4. Sag mir, dass ich Claude jetzt neu starten muss.

Danach: Analysier mit REA das Programm [Pfad zur .exe]. Öffne es mit `open_binary`, verschaff dir mit `binary_overview` einen Überblick, schau dir die Funktionen zu [Funktion] an und schreib alles in `analyse.md`.

DEINE EIGENE VERSION BAUEN

Jetzt kommt der zweite Prompt. Danach weiß Claude, wie die App funktioniert, und baut dir daraus deine eigene. Mit deinen eigenen Wünschen, einfach auf Deutsch. Schreib deine Wünsche in die eckigen Klammern, zum Beispiel „Aufgaben mit Fälligkeitsdatum, Projekte, eine Tagesansicht und Tastenkürzel“.

Claude baut die App zuerst als Web-App, die auf deinem Rechner im Browser läuft. Das ist der schnellste Weg zu einem Ergebnis, und daraus wird später auch die iPhone- oder Android-Version.

TIPP

Änderungen sagst du einfach im Chat: „Mach die Seitenleiste schmaler“ oder „Füg eine Suche über alle Aufgaben hinzu“. Claude baut um und testet neu.

EIGENE VERSION BAUEN

Lies `analyse.md` (und `analyse-web.md`, falls vorhanden). Bau mir daraus im neuen Unterordner `meine-app` meine eigene Version dieser App, als Web-App, die im Browser auf meinem Rechner läuft.

- Schreib den ganzen Code selbst neu. Übernimm keinen Code, keine Grafiken, keine Texte, kein Logo und nicht den Namen der Original-App. Gib der App einen eigenen Namen und ein eigenes Design.
- Bau zuerst nur diese Funktionen: [deine Wünsche].
- Speicher die Daten erst mal lokal auf meinem Rechner.
- Start die App am Ende, öffne sie im Browser und test jede Funktion selbst.
- Erklär mir danach, was noch fehlt und was Funktionen wie Sync zwischen Geräten oder KI kosten würden.

DER EHRliche TEIL

REA gibt dir nicht den Originalcode. Es zeigt dir, wie eine App aufgebaut ist: Fenster, Funktionen, Abläufe, Serververbindungen, bei nativen Apps zurückübersetzten Pseudocode. Den Code deiner Version schreibt Claude neu.

Was es kostet: REA, Java, JADX, Ghidra und die Hopper-Demo kosten nichts. Den Teil einer App, der auf deinem Rechner läuft, baust du also kostenlos nach. Sobald Daten zwischen Geräten synchronisiert, mit anderen geteilt oder von einer KI verarbeitet werden, brauchst du einen Server oder eine KI-Schnittstelle, und die kosten auch bei deiner eigenen Version Geld. Große Analysen und lange Bau-Sessions ziehen außerdem spürbar an deinem Claude-Kontingent.

Was erlaubt ist: In Deutschland darfst du eine App, die du rechtmäßig nutzt, beobachten, untersuchen und testen, um ihre Ideen und Grundsätze zu verstehen (§ 69d Abs. 3 UrhG). Ideen und Funktionen sind nicht geschützt (§ 69a Abs. 2 UrhG), du darfst sie mit eigenem Code nachbauen. Nicht erlaubt ist, Code, Grafiken, Texte, Namen oder Logos zu übernehmen oder einen Kopier- oder Abo-Schutz zu umgehen. Das Dekompilieren erlaubt das Gesetz nur in engen Grenzen, etwa damit Programme zusammenarbeiten (§ 69e UrhG), nicht um eine Kopie zu bauen. REA selbst ist für Software gedacht, die dir gehört oder die du untersuchen darfst.

Hopper-Demo: Eine Sitzung läuft höchstens 30 Minuten, und du kannst die Analyse nicht speichern. Für die meisten Fragen reicht das.

DEINE APP VERÖFFENTLICHEN

Deine App läuft auf deinem Rechner. Jetzt bringst du sie zu anderen. Der einfachste Weg ist das Web, dafür reicht ein kostenloser Account bei Netlify. Für den App Store brauchst du einen Mac, ein bezahltes Apple-Entwicklerkonto und etwas Geduld bei der Prüfung durch Apple.

01

DER EINFACHSTE WEG: ALS WEBSITE VERÖFFENTLICHEN

Claude macht aus deiner App einen fertigen Ordner mit allen Dateien. Leg dir auf netlify.com einen kostenlosen Account an (Free-Plan, kostenlos), melde dich an und zieh den Ordner dann auf app.netlify.com/drop. Deine App ist danach online. Ohne Anmeldung schützt Netlify die Adresse mit einem vorläufigen Passwort, dann sieht sie niemand außer dir.

TIPP

Vercel ist eine Alternative, dessen kostenloser Hobby-Plan ist aber nur für private, nicht kommerzielle Nutzung.

FÜR DAS WEB FERTIG MACHEN

Mach meine App im Ordner `meine-app` fertig für die Veröffentlichung als Website. Bau einen fertigen Ordner, der direkt im Browser läuft, mit einer `index.html`. Prüf, dass alles ohne lokalen Server funktioniert, und sag mir genau, welchen Ordner ich auf app.netlify.com/drop ziehen soll.

01

APPLE-ENTWICKLERKONTO ANLEGEN

Diesen Schritt machst du selbst, weil es um deine Identität und eine Zahlung geht.

1. Prüf, dass in deinem Apple-Account die Zwei-Faktor-Authentifizierung an ist. Am iPhone:
Einstellungen > dein Name > Anmeldung & Sicherheit.
2. Geh auf developer.apple.com/programs/enroll und melde dich als Einzelperson an. Du musst volljährig sein und deinen echten Namen angeben. Dieser Name steht später im App Store als Anbieter.
3. Bezahl die 99 € im Jahr. Apple zeigt dir den genauen Betrag bei der Anmeldung. Die Mitgliedschaft verlängert sich automatisch, bis du sie kündigst.

Die Bestätigung sollte innerhalb von 24 Stunden kommen. Bevor du die erste App anlegen kannst, musst du in App Store Connect unter **Business** die aktuelle Vereinbarung von Apple akzeptieren.

01

XCODE INSTALLIEREN

Xcode ist Apples kostenloses Programm, mit dem iPhone-Apps gebaut und hochgeladen werden. Claude Code steuert es für dich, installiert sein muss es aber.

1. Öffne den Mac App Store, such **Xcode** und installier es. Die aktuelle Version Xcode 27.1 braucht macOS Tahoe 26.6 oder neuer. Prüf deine Version oben links über **Apple-Menü > Über diesen Mac**.
2. Öffne Xcode einmal. Wenn es fragt, welche Plattformen du installieren willst, wähl **iOS**. Das dauert je nach Internet eine Weile.

DIE IPHONE-VERSION BAUEN UND IM SIMULATOR TESTEN

Claude baut aus deiner Web-App eine echte iPhone-App und testet sie im iOS Simulator, einem virtuellen iPhone auf deinem Mac. In der Claude Desktop-App auf dem Mac gibt es dafür eigens einen Simulator-Bereich. Darin siehst du, wie Claude die App startet und sich durch die Bildschirme tippt. Der Bereich ist noch in der öffentlichen Beta, läuft ab Claude Pro und braucht die Desktop-App ab Version 1.24012.0 mit installiertem Xcode.

Die Bundle ID ist der eindeutige Name deiner App bei Apple, zum Beispiel `com.deinname.meineapp`. Nach dem ersten Upload lässt sie sich nicht mehr ändern, also wähl sie in Ruhe.

ACHTUNG

Die Screenshots aus dem Simulator gehen zur Auswertung an Anthropic. Melde dich im Simulator deshalb nicht mit echten Accounts an. Claude steuert nur das virtuelle iPhone, kein echtes Gerät.

IPHONE-APP BAUEN

Bau aus meiner App im Ordner `meine-app` eine native iPhone-App mit SwiftUI im neuen Ordner `meine-app-ios`. Gleiche Funktionen, gleiche Logik, eigenes Design.

1. Leg ein Xcode-Projekt an mit der Bundle ID `com.[deinname].[appname]` und iOS 17 als Mindestversion.
2. Aktivier unter Signing & Capabilities `Automatically manage signing`.
3. Bau die App und starte sie im iOS Simulator auf einem iPhone 17. Tipp dich durch jede Funktion, prüf, dass nichts abstürzt, und zeig mir am Ende Screenshots.
4. Sag mir, was noch fehlt, bevor die App in den App Store kann.

STORE-TEXTE, DATENSCHUTZ UND SCREENSHOTS VON CLAUDE VORBEREITEN LASSEN

Apple verlangt für jede App ein paar Pflichtangaben. Fast alles davon schreibt Claude dir:

Datenschutzerklärung als Link, im App Store und in der App selbst.

App-Datenschutz-Angaben: welche Daten deine App sammelt, auch über eingebaute Fremd-Bibliotheken.

Altersfreigabe über einen Fragebogen. Ohne Altersfreigabe wird keine App veröffentlicht.

Screenshots: mindestens einer für das iPhone mit Dynamic Island, maximal zehn pro Gerätegröße, ohne Transparenz.

Account löschen: Kann man in deiner App ein Konto anlegen, muss man es in der App auch wieder löschen können.

APP STORE VORBEREITEN

Bereite alles vor, was ich für den App Store für meine App in meine-app-ios brauche, und speicher es im Ordner app-store:

1. Eine Datenschutzerklärung auf Deutsch als einfache Website im Ordner app-store/datenschutz mit einer index.html, passend zu dem, was die App wirklich speichert und sendet. Sag mir, dass ich den Ordner angemeldet auf app.netlify.com/drop ziehen soll, und bau den Link danach in die App ein.
2. Name, Untertitel, Beschreibung und Keywords für den App Store, auf Deutsch, ohne den Namen der Original-App.
3. Meine Antworten für die App-Datenschutz-Angaben und den Fragebogen zur Altersfreigabe, als Liste zum Abtippen.
4. Fünf Screenshots aus dem Simulator auf einem iPhone 17. Prüf, dass sie 1206 x 2622 Pixel groß sind, und speicher sie in app-store/screenshots.
5. Falls die App Konten hat: Bau eine Funktion zum Löschen des Kontos in die App ein.

DIE APP HOCHLADEN

Das Hochladen machst du in Xcode mit ein paar Klicks. Den Eintrag für deine App in App Store Connect musst du vorher nicht anlegen: Fehlt er beim ersten Upload, fragt Xcode dich nach den nötigen Angaben und legt ihn für dich an.

1. Schreib Claude: „Öffne das Projekt meine-app-ios in Xcode.“
2. Klick im Projekt links auf deine App, dann oben auf **Signing & Capabilities**. Wähl im Feld **Team** dein bezahltes Entwickler-Team. Taucht dein Team nicht auf, schreib Claude: „Zeig mir, wie ich meinen Apple-Account in Xcode hinzufüge.“
3. Wähl oben in der Leiste als Ziel ein Gerät statt eines Simulators, zum Beispiel **Any iOS Device (arm64)**.
4. Klick im Menü auf **Product > Archive**. Xcode baut das Archiv und öffnet danach den Organizer. Falls nicht: **Window > Organizer**.
5. Wähl links **Archives**, markier dein Archiv und klick auf **Distribute App**.
6. Wähl **TestFlight & App Store** und klick auf **Distribute**. Xcode signiert, verpackt und lädt die App hoch.

BEI APPLE ZUR PRÜFUNG EINREICHEN

Geh auf appstoreconnect.apple.com und trag dort ein, was Claude im Ordner app-store vorbereitet hat.

1. Klick auf **Apps** und wähl deine App.
2. Klick links auf die App-Version, die du einreichen willst.
3. Füll die Felder aus: Beschreibung, Keywords, Link zur Datenschutzerklärung, Screenshots, Altersfreigabe und App-Datenschutz.
4. Scroll rechts runter zum Bereich **Build** und prüf, dass dein hochgeladener Build ausgewählt ist. Direkt nach dem Upload kann es eine Weile dauern, bis er dort erscheint.
5. Klick oben rechts auf **Add for Review**.
6. Klick auf **Submit for Review**. Der Status wechselt zu In Review, sobald Apple mit der Prüfung beginnt.

Apple prüft nach eigenen Angaben mindestens 50 Prozent der Einreichungen in weniger als 24 Stunden und 90 Prozent in weniger als 48 Stunden. Hat deine App einen Login, gib Apple im Bereich App Review Information einen Demo-Zugang.

ACHTUNG

Apple lehnt reine Kopien ab. Die Richtlinien verbieten ausdrücklich, eine beliebte App einfach zu kopieren oder nur Name und Oberfläche leicht zu ändern (Regel 4.1). Apps, die sich nicht von bereits Verfügbarem unterscheiden, werden ebenfalls abgelehnt (Regel 4.3). Und deine App darf nur Inhalte enthalten, die du selbst erstellt hast oder nutzen darfst (Regel 5.2). Gib deiner App deshalb einen eigenen Namen, ein eigenes Design und mindestens eine Funktion, die das Original nicht hat. Wer andere Apps imitiert, riskiert den Ausschluss aus dem Entwicklerprogramm.

OPTIONAL: IN DEN GOOGLE PLAY STORE

Wenn deine App auch auf Android-Handys laufen soll.

Für Android läuft es ähnlich, mit drei Unterschieden:

Das Entwicklerkonto in der [Google Play Console](#) kostet einmalig rund 22 €. Du brauchst einen Ausweis und eine Kreditkarte auf deinen Namen, Prepaid-Karten gehen nicht.

Neue private Entwicklerkonten müssen vor der Veröffentlichung einen geschlossenen Test machen: mindestens 12 Tester, die mindestens 14 Tage am Stück dabei sind. Springt einer ab, beginnen die 14 Tage neu.

Google will die App als Android App Bundle (.aab). Das baut [Android Studio](#), kostenlos.

ANDROID-APP BAUEN

Bau aus meiner App im Ordner meine-app eine native Android-App mit Kotlin im neuen Ordner meine-app-android. Gleiche Funktionen, eigenes Design. Installier dafür, was fehlt, ohne Admin-Rechte. Erzeug am Ende ein signiertes Android App Bundle (.aab), speicher den Signatur-Schlüssel sicher im Ordner meine-app-android/schlüssel und sag mir, dass ich davon eine Sicherheitskopie machen soll.

WENN DAS NICHT KLAPPT

Claude kennt die REA-Werkzeuge nicht: Claude Desktop-App komplett beenden und neu öffnen, dann eine neue Session im Code-Tab starten. Hilft das nicht, den Reparatur-Prompt aus Schritt 2 schicken.

REA meldet artifact_integrity_mismatch: Das passiert bei manchen Electron-Apps, weil ihre Hersteller Dateien nach dem Verpacken noch signieren. Schreib Claude: „Wiederhol die Analyse mit integrity_policy record-and-continue.“

Die Antwort ist riesig oder bricht ab: Schreib Claude: „Nutz detail summary und schau dir danach nur die Module zu [Funktion] einzeln an.“

Chrome öffnet sich, aber REA findet keine Seite: Meist läuft Chrome ohne eigenes Profil. Schließ alle Chrome-Fenster und lass Claude Chrome neu mit dem Profilordner rea-chrome-profil starten.

JADX oder Ghidra nicht bereit: Schreib Claude: „Prüf den env-Eintrag von mcpServers.rea in .claude.json und führ den doctor-Befehl für den Provider aus.“ Danach Claude neu starten.

Hopper reagiert nicht: Hopper einmal selbst öffnen und den Demo-Modus wählen, dann die Analyse neu starten. Läuft eine Demo-Sitzung über 30 Minuten, endet sie. Starte dann eine neue.

Xcode lässt dich nicht hochladen: Prüf unter Signing & Capabilities, dass dein bezahltes Team ausgewählt ist und nicht „Personal Team“. Mit dem kostenlosen persönlichen Team kannst du nur auf eigenen Geräten testen.

HÄUFIGE FRAGEN

BEKOMME ICH MIT REA DEN CODE DER ORIGINAL-APP?

Nein. REA zeigt dir, wie die App aufgebaut ist und wie ihre Funktionen ablaufen. Bei nativen Apps siehst du zurückübersetzten Pseudocode. Deine eigene Version schreibt Claude komplett neu.

GEHT DAS AUCH MIT IPHONE-APPS?

Bei Apps, die es nur im iPhone App Store gibt, geht das aktuell nicht. Bei Android oder im Web schon. Gibt es die App auch als Web-Version oder Desktop-App, nimm Weg A oder B.

WAS PASSIERT MIT MEINEN DATEN?

Die App-Dateien bleiben auf deinem Rechner, REA liest sie dort. Die Ergebnisse der Analyse, deine Nachrichten und Screenshots aus dem Simulator gehen an Claude. Bei Web-Apps nimmt REA keine Passwörter, Cookie-Werte oder gespeicherten Daten in die Analyse auf.

BRAUCHE ICH PROGRAMMIERKENNTNISSE?

Nein. Du kopierst die Prompts in den Claude Code Chat und bestätigst, wenn Claude fragt. Selbst klicken musst du nur bei Hopper, beim Einloggen in Chrome, am Android-Handy, beim Apple-Konto und beim Hochladen in Xcode.

ACHTUNG

Hinweis: Dieser Guide dient zu Informations- und Bildungszwecken. Alle Inhalte spiegeln den Stand bei Erstellung wider. Oberflächen, Funktionen und Preise der genannten Tools können sich ändern. Prüfe Ergebnisse von KI-Tools immer selbst nach. @GPTMarlon übernimmt keine Haftung für Entscheidungen auf Grundlage dieser Inhalte. Dieser Guide stellt keine Anlage-, Steuer-, Rechts- oder Gesundheitsberatung dar. Lass Ergebnisse bei Bedarf fachlich prüfen.

Datenschutz: Analyse-Ergebnisse, Nachrichten und Simulator-Screenshots werden an Anthropic übertragen. Analysiere keine Apps oder Seiten, in denen personenbezogene Daten Dritter sichtbar sind, und melde dich im Simulator nicht mit echten Accounts an.