

Appgefahren

Wir entwickeln eine iPhone App

Prof. Ing. Gerrit Zeissl, 27.6.2023, Wien

Inhalt

- Motivation
- Über mich
- Warum Apple
- Entwicklertools, -APIs, -Frameworks, -Hilfen
 - XCode, Swift Playgrounds, CreateML, Reality Composer, ...
- Der iOS Lifecycle + Human Interface Guidelines
- Swift
- Was werden wir heute machen
- Release und Automatisierung
- Zusammenfassung und Fragen

Motivation

- Meine erste “Projektwoche”
- Wollte nicht die hundertste Schnitzeljagd machen
- Stattdessen etwas, das mich selbst interessiert
- Hoffe, dass Sie einiges mitnehmen können
- Danke vorab für Ihr Interesse
- Lockerer Ablauf, müssen nicht mitmachen, aber bitte nicht die anderen stören
- Wenn Sie Pausen brauchen, einfach bescheid sagen
- Bei Fragen, einfach unterbrechen

Über mich

- “Jung” Lehrer seit diesem Schuljahr
- 2005 maturiert
- Seit 2011 durchgehend Software Entwickler
- 6 Jahre C++
- 4 Jahre iOS (selbst beigebracht)
 - Alles heute gezeigte ist so wie ich es gelernt/verstanden habe.
Ich übernehme keine Verantwortung für 100%ige Richtigkeit
- 1 Jahr Lehrer
- Noch keine eigene App im Appstore, Plan für die nahe Zukunft

 **mikme**
Musik
Entwickelt für iPhone. Nicht für macOS überprüft.



ALTER **4+**
Jahre

KATEGORIE  Musik

ENTWICKLER  Mikme

SPRACHE **DE**
+ 3 weitere

GRÖSSE **94,6**
MB

Neue Funktionen

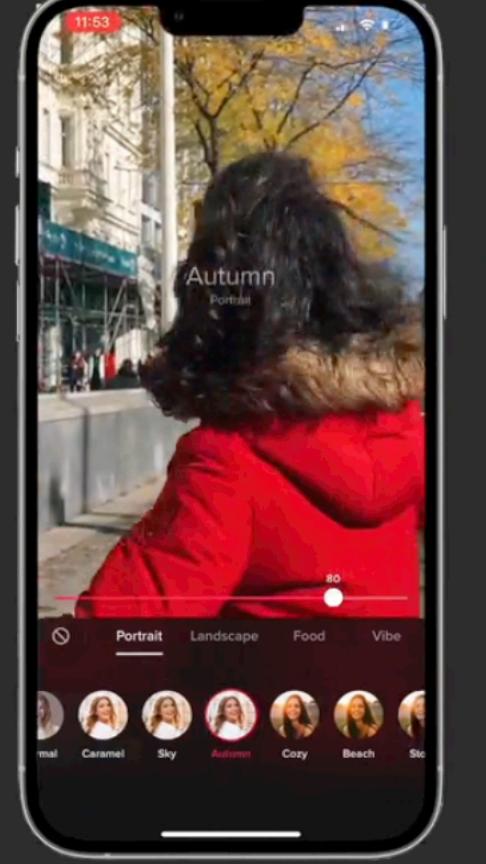
- Easily add a brand logo, intro or outro video in the app settings
- Bug fix when using 4 mikme devices

[Mehr](#) [Versionsverlauf](#)

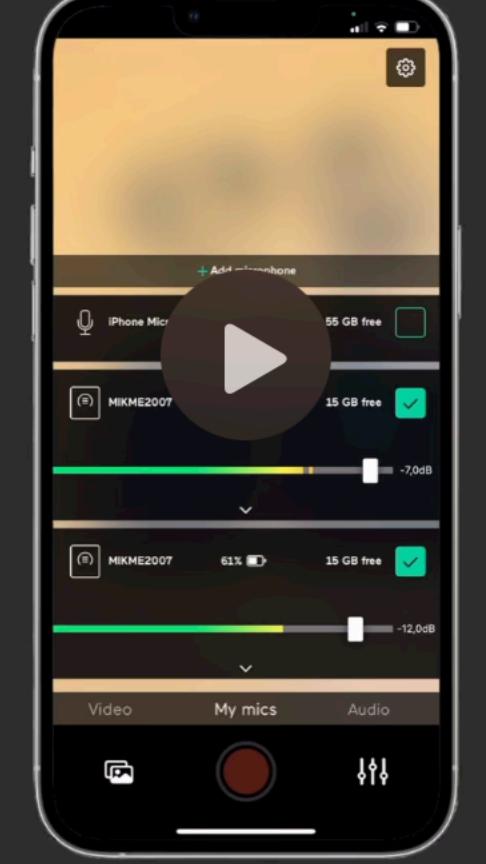
Vor 1 J.
Version 4.2.0

Vorschau

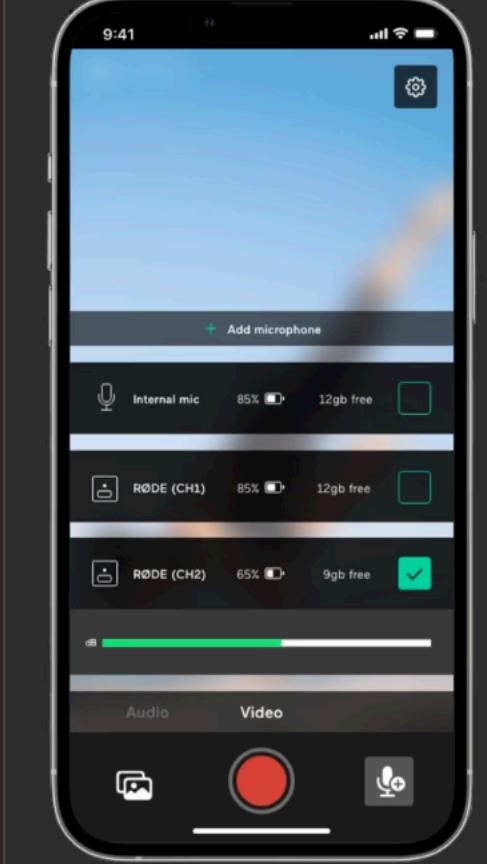
 Nur auf dem iPhone

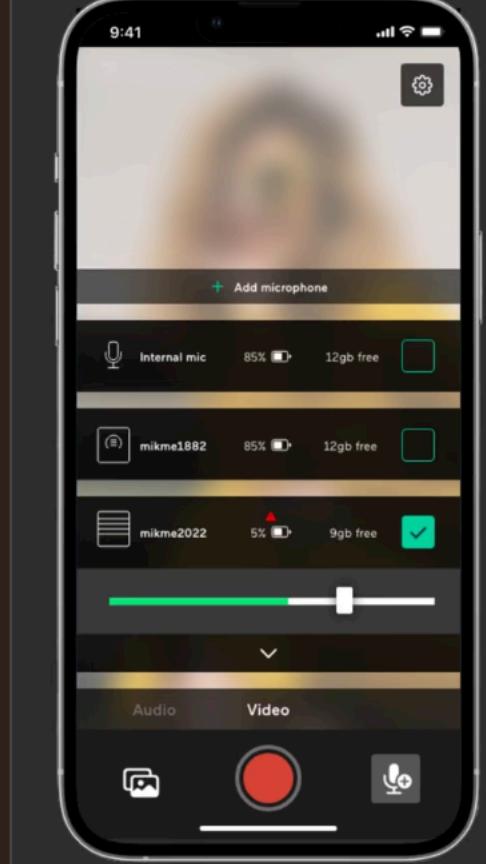
with great audio


Advanced camera settings


Manage all your microphones


Record audio & video wirelessly


Use RØDE or any other ext mics


Pair wireless microphones


 **eID.li**
Dienstprogramme
Entwickelt für iPhone. Nicht für macOS überprüft.

LADEN

ALTER
4+
Jahre

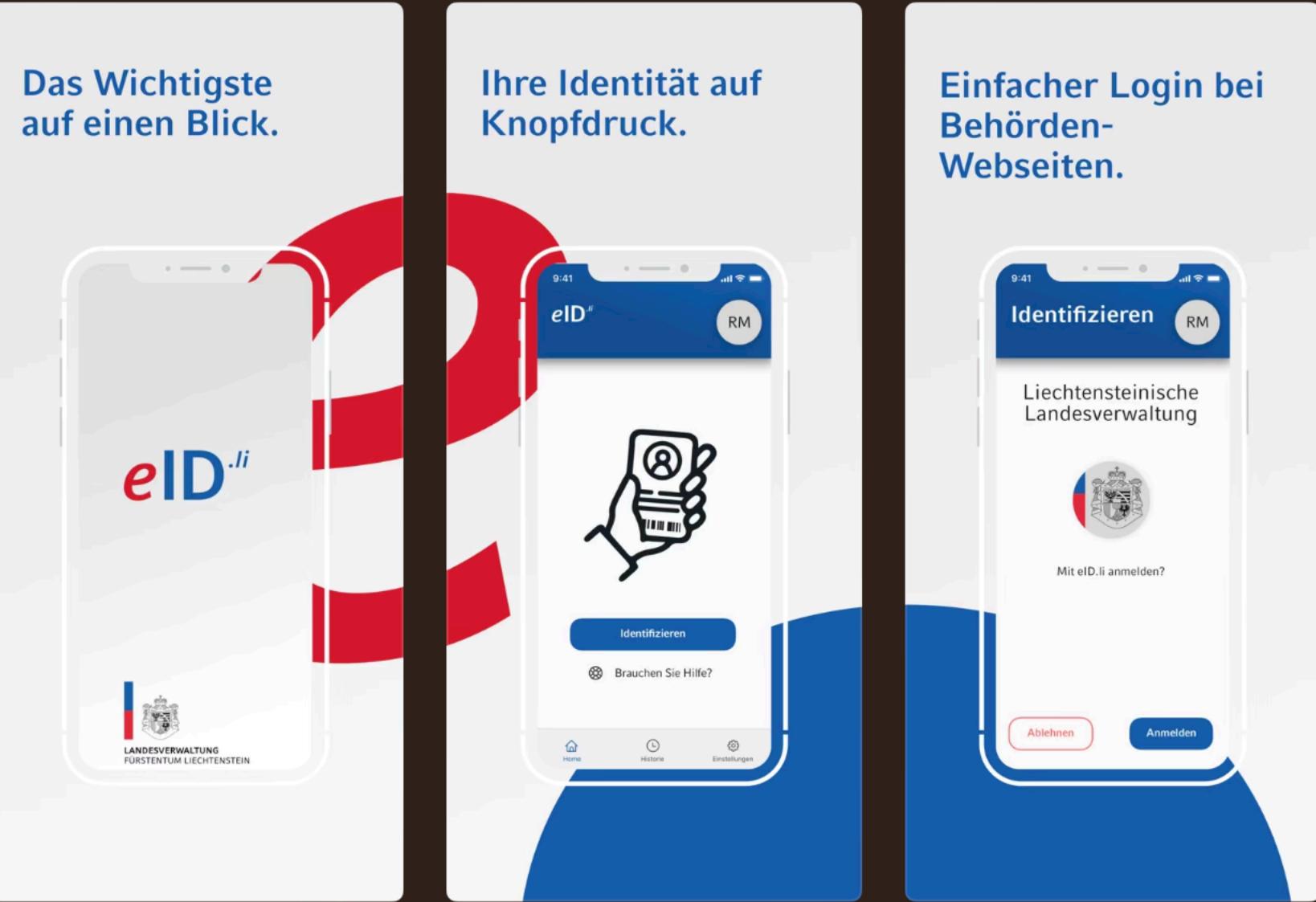
KATEGORIE
 Dienstprogramme

ENTWICKLER
 Liechtensteinische Landesverwaltung

SPRACHE
DE
+ 2 weitere

GRÖSSE
74
MB

 Nur auf dem iPhone



eID.li ermöglicht es, das Mobiltelefon als digitale ID zu verwenden. Damit kann auf sichere Webdienste zugegriffen werden, ohne dass ein Kartenleser erforderlich ist. Neuere Mobiltelefone können so konfiguriert werden, dass sie anstelle eines Passworts für die Anmeldung Fingerabdruck- oder Gesichtserkennung verwenden. Das System basiert auf der neuesten Sicherheitstechnologie, die in modernen Mobiltelefonen implementiert ist. Private Schlüssel werden an einem sicheren Ort in Ihrem Mobiltelefon aufbewahrt, der von außen nicht zugänglich und vor Hacking geschützt ist. Um die eID.li zu erhalten, muss man sich persönlich durch das Liechtensteinische Ausländer- und Passamt in Vaduz registrieren lassen.

[Liechtensteinische Lan...](#)

Website  Support 



eAusweise 4+

Einfach, sicher, digital

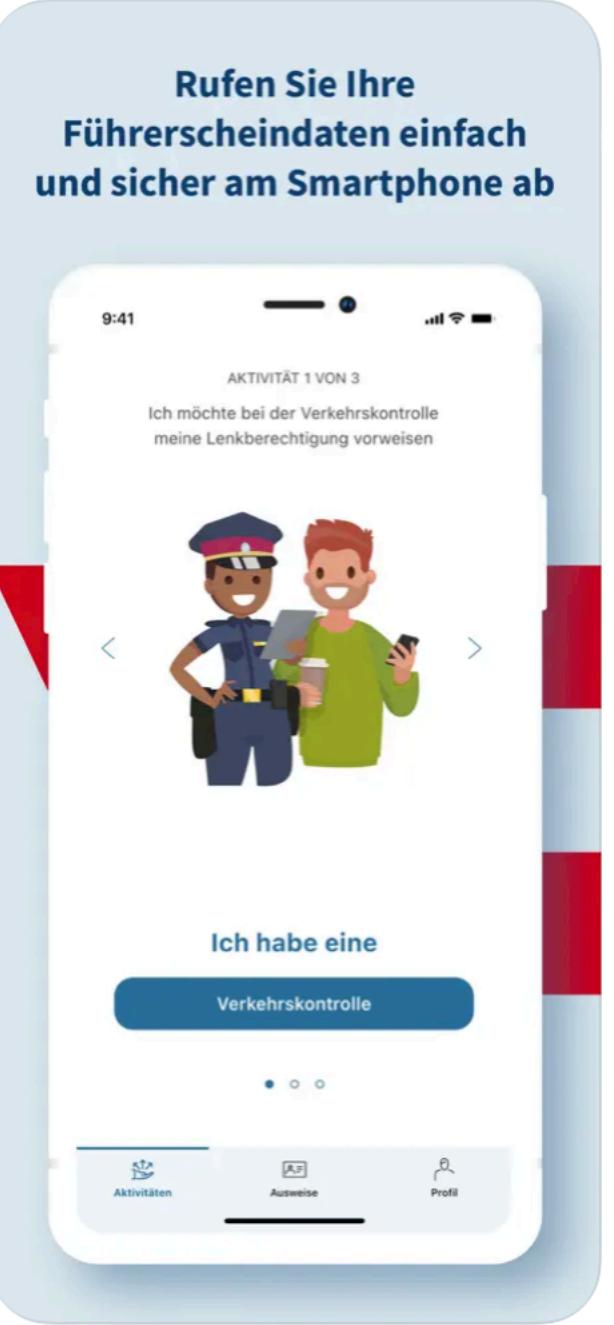
Bundesministerium für Finanzen

Nr. 4 in Produktivität

★★★★★ 2,2 • 622 Bewertungen

Gratis

iPhone-Screenshots



Rufen Sie Ihre Führerscheindaten einfach und sicher am Smartphone ab

AKTIVITÄT 1 VON 3
Ich möchte bei der Verkehrskontrolle meine Lenkberechtigung vorzeigen.

Ich habe eine Verkehrskontrolle

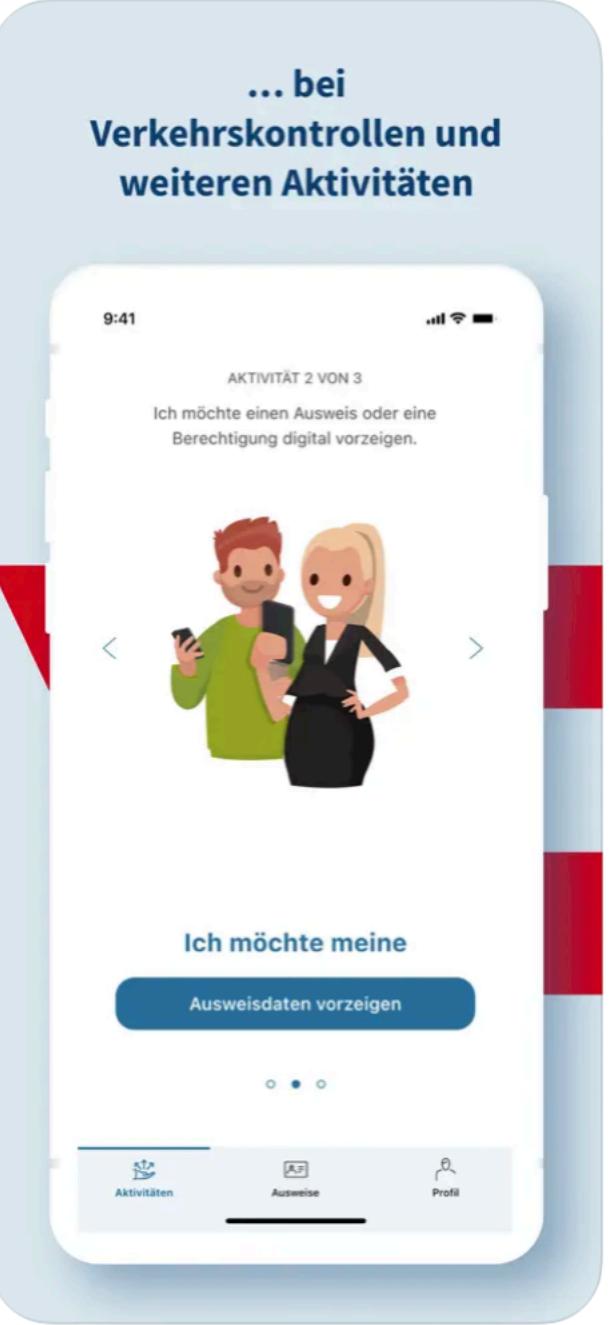


Weisen Sie Ihre Lenkberechtigung mittels QR-Code vor...

Wie und Was wird übergeben?

Datenübergabe-QR-Code
Lassen sie nun den QR-Code durch die prüfende Person scannen.

Abbrechen



... bei Verkehrskontrollen und weiteren Aktivitäten

AKTIVITÄT 2 VON 3
Ich möchte einen Ausweis oder eine Berechtigung digital vorzeigen.

Ich möchte meine Ausweisdaten vorzeigen



Ihre Daten sind mit Fingerabdruck oder Gesichtserkennung geschützt

eAusweise sichern

Ausweise mit Face ID schützen
Um Ihren digitalen Ausweis und Ihre persönlichen Informationen zu schützen, ist es notwendig Face ID zu nutzen.

Mit Face ID schützen

Mit der App „eAusweise“ weisen Sie Ihren Führerschein in Österreich einfach, sicher und digital am Smartphone vor – sowohl bei einer Verkehrskontrolle als auch im privaten Umfeld. Teilen Sie ihre Führerscheindaten einfach per Knopfdruck und komplett kontaktlos. Es findet keine Datenübertragung ohne Ihr Einverständnis statt.

[mehr](#)

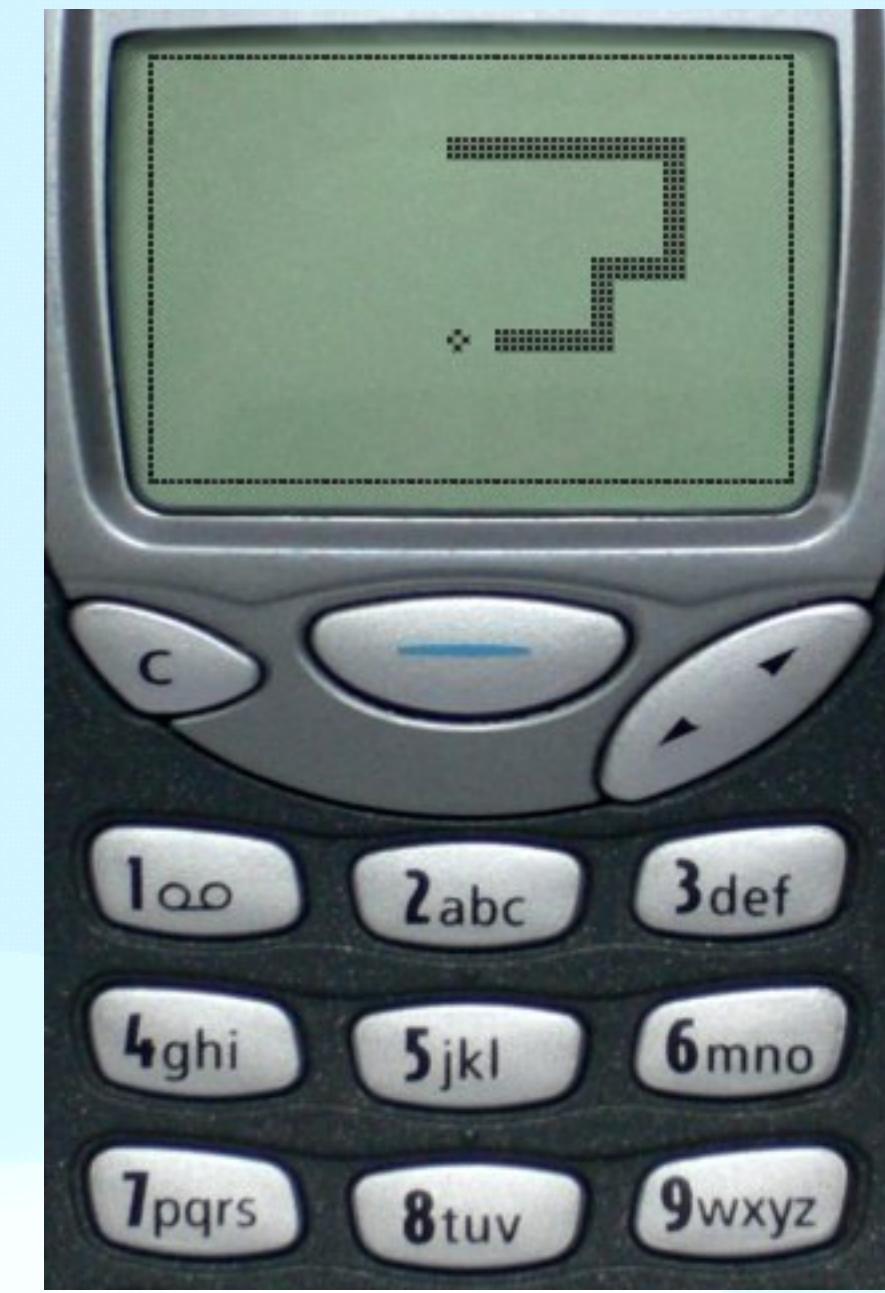
Warum Apple

- Mein Einstieg ins Software Entwickeln war Schadsoftware
- Wendepunkt war Blaster Wurm (2003, Windows XP)
- Angefangen mit Linux
 - Damals sehr komplex und schwer passende Treiber zu bekommen
 - Wollte etwas, dass so sicher ist wie Linux, aber so einfach wie Windows
- Der Umstieg auf MacOS
- Wenn auch teuer, elegante Hardware mit passender Software, leicht verständlich

Warum Apple

Fortsetzung

- Handys damals bei weitem keine Smartphones
- Habe schon damals erkannt was für ein Meilenstein Internet am Handy wäre
- 2007 kam erstes iPhone
- Anfangs noch Android verwendet (kein Beef mit Android, ist auch Linux)
- iPhone 3G mein erstes Apple Smartphone
- Seit dem etwas zum Fanboy entwickelt



Warum Apple

Fortsetzung

- In den letzten 10 Jahre erfolgte der Aufstieg zur reichsten Firma der Welt
- Sind nicht immer die ersten, aber “die, die es richtig machen”
- Gesamtes ECO System
 - zB. Handoff zwischen mehreren Geräten oder Sidecar
- “Es funktioniert einfach”
- Fokus auf Qualität und Sicherheit, lange Software Updates
- Wissen, wie man sich vermarktet
- Alles kommt aus einem Haus, darum aufeinander abgestimmt
- Guter Support, gute Dokumentation

Entwickler- Tools, APIs, Frameworks, Hilfen

- Jeder kennt Keynotes, wo Neues vorgestellt wird
- WWDC - Worldwide Developer Conference, speziell für Entwickler
- Hunderte von Videos wo Neuheiten MIT DEMOS gezeigt werden
 - Online, mit Quellcode verfügbar
 - <https://developer.apple.com/wwdc23/sessions/>
- Hilfeseiten
 - developer.apple.com
 - Stackoverflow, Apple Forum, Google, ChatGPT, etc.

Für Entwickler

- Mac mit Apple Account (AppleID)
 - Gratis (keine Apps im Appstore und weitere Einschränkungen)
 - Paid (99€ im Jahr, 30% revenue für Apple, alle Möglichkeiten)
 - <https://developer.apple.com/support/compare-memberships/>
- Xcode - die Entwicklungsumgebung
 - Kann Apps für macOS, iOS, tvOS, watchOS, iPadOS und ab Version 15 auch visionOS entwickeln
 - Beinhaltet auch einen 2D und 3D Designer für Spiele und AR Erlebnisse, Simulatoren für Geräte, Machine Learning Programm, GIT Client etc. Darum sehr groß (über 10GB)
- Swift Playgrounds - die Mobile Entwicklungsumgebung
 - Entworfen um Programmieren am iPad zu lernen
 - Kann seit Dezember 2021 (Version 4.0) direkt vom iPad aus Apps in den Store laden
- Diese Apps sind Voraussetzung für heute und hoffentlich bereits installiert

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

35°C Do. 22. Juni 11:34

Neat-Test

Neat-Test > iPhone 14

Finished running Neat-Test on iPhone 14

Neat TestApp ContentView

No Selection

Neat-Test > Neat-Test > Neat_TestApp > No Selection

```
1 //  
2 // Neat_TestApp.swift  
3 // Neat-Test  
4 //  
5 // Created by Gerrit Zeissl on 19.06.23.  
6 //  
7  
8 import SwiftUI  
9  
10 @main  
11 struct Neat_TestApp: App {  
12     var body: some Scene {  
13         WindowGroup {  
14             ContentView()  
15         }  
16     }  
17 }
```

No Selection

NODE_5, Type: hidden, Activation: hat, Activation Response: 1.2573473338869743, Position: NPosition(x: 75.0, y: 100.0, z: 400.0)
NODE_11, Type: hidden, Activation: cube, Activation Response: 1.0, Position: NPosition(x: 137.5, y: 50.0, z: 200.0)
NODE_16, Type: hidden, Activation: sigmoid, Activation Response: 1.0, Position: NPosition(x: 168.75, y: 25.0, z: 100.0)

Innovation_1, [from=1 : to=4], Enabled: true, Recurrent: false, Weight: -4.982885573986659 --
Innovation_2, [from=2 : to=4], Enabled: true, Recurrent: false, Weight: 2.426275709277549 --
Innovation_3, [from=3 : to=4], Enabled: true, Recurrent: false, Weight: 1.0 --
Innovation_4, [from=1 : to=5], Enabled: true, Recurrent: false, Weight: 1.0 --
Innovation_5, [from=5 : to=4], Enabled: true, Recurrent: false, Weight: -6.465711659627433 --
Innovation_8, [from=3 : to=5], Enabled: true, Recurrent: false, Weight: 1.0 --
Innovation_9, [from=2 : to=5], Enabled: true, Recurrent: false, Weight: -0.6628118538443959 --
Innovation_20, [from=2 : to=11], Enabled: true, Recurrent: false, Weight: 1.0 --
Innovation_21, [from=11 : to=5], Enabled: true, Recurrent: false, Weight: -0.4317593372966533 --
Innovation_30, [from=2 : to=16], Enabled: true, Recurrent: false, Weight: 1.0 --
Innovation_31, [from=16 : to=11], Enabled: true, Recurrent: false, Weight: 1.8 --
Innovation_34, [from=16 : to=5], Enabled: true, Recurrent: false, Weight: 2.020401260121819 --

Fitness: 15.592050177364312

Info.plist, Project Settings, Signing und Bundle Identifier

The screenshot displays two main views from the Xcode interface: the Info.plist editor and the Project settings.

Info.plist Editor: This view shows the contents of the Info.plist file for the "SwiftUIMVVM" target. It includes sections for "Information Property List" and "App Transport Security Settings". In the "App Transport Security Settings" section, the "Allow Arbitrary Loads" key is set to YES.

Key	Type	Value
NSAppTransportSecurity	Dictionary	(1 item)
NSAllowsArbitraryLoads	Boolean	YES

Project Settings: This view shows the "General" tab of the project settings for the "SwiftUIMVVM" target. It includes sections for "Supported Destinations", "Minimum Deployments", "Identity", and "Deployment Info".

- Supported Destinations:** Lists "iPhone", "iPad", and "Mac (Designed for iPad)" as supported destinations.
- Minimum Deployments:** Set to iOS 15.5.
- Identity:** App Category: None, Display Name: Display Name, Bundle Identifier: com.test.swiftuimvvm.SwiftUIMVVM, Version: 1.0, Build: 1.
- Deployment Info:** iPhone Orientation: Portrait, Landscape Left, Landscape Right; iPad Orientation: Portrait, Landscape Left, Landscape Right; Status Bar Style: Default.

Create ML

The screenshot shows the Create ML application interface. On the left, the Project sidebar lists "FlowerClassifier" with "Model Sources (2)" and "Data Sources (3)". "FlowerClassifier 2" is selected. In the main area, the "Settings" tab is active, showing the "Data" section with "Training Data" (5 classes, 4.307 items), "Validation Data" (Auto, Split from Training Data), and "Testing Data" (5 classes, 10 items). The "Parameters" section includes "Iterations" set to 25 and "Augmentations" (Add Noise, Blur, Crop, Expose, Flip, Rotate) all checked. A status message at the bottom says "Completed 25 iterations". On the right, the "Activity" sidebar shows a timeline of events from Jun 8, 2023, including "Testing Started" at 5:55 PM, "Testing Completed" at 5:55 PM, "Training Completed" at 5:55 PM, "Training Started" at 5:52 PM, "Data Source Created" at 5:52 PM, "Model Source Created" at 5:52 PM, and "Project Created" at 2:00 PM.

Project

FlowerClassifier

Model Sources (2)

FlowerClassifier 1

FlowerClassifier 2

Data Sources (3)

flowers

training

verify

Train

Settings

Training

Evaluation

Preview

Output

Activity

99 % Training | 94 % Validation | 90 % Testing

Data

Training Data

5 Classes | 4.307 Items

View

Validation Data

Auto

Split from Training Data

Automatic

Testing Data

5 Classes | 10 Items

View

verify

Parameters

Iterations 25

Augmentations

- Add Noise
- Blur
- Crop
- Expose
- Flip
- Rotate

Completed 25 iterations

Activity

Jun 8, 2023

Testing Started 5:55 PM

FlowerClassifier 2

Testing Completed 5:55 PM

FlowerClassifier 2

Training Completed 5:55 PM

25 iterations

Training Started 5:52 PM

25 iterations

Training Data Added 5:52 PM

training

Model Source Created 5:52 PM

FlowerClassifier 2

Data Source Created 2:07 PM

verify

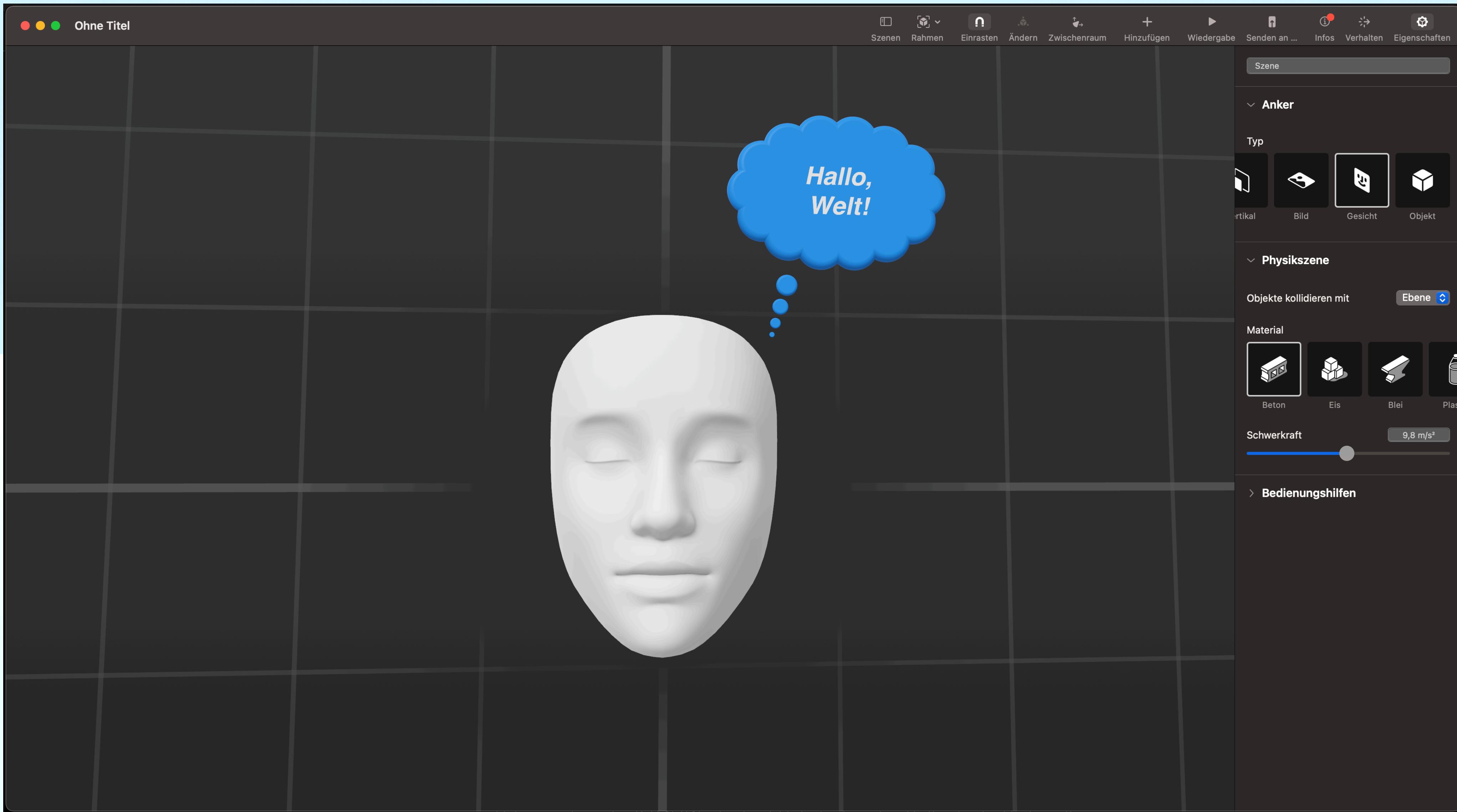
Data Source Created 2:07 PM

training

Project Created 2:00 PM

FlowerClassifier

Reality Composer (Pro angekündigt für Vision Pro)



APIs und (3rd Party) Frameworks

- Massen an APIs die das Arbeiten erleichtern, zum Beispiel:
- UIKit, MapKit, SpriteKit, SceneKit, ARKit, Core Location, Core Motion, GameKit, AV Foundations, Media Player etc.
- Jeder kann eigene Kits entwickeln und anderen zur Verfügung stellen
- Früher via Tools wie Carthage und CocoaPods, heute mittels Swift Package manager, ein in Xcode integriertes Paket Verwaltungs Tool
- All diese APIs ermöglichen das einfache zugreifen auf div. Hardware wie die Kamera, GPS oder die Sensoren sowie Top Level Code für komplexe Programme

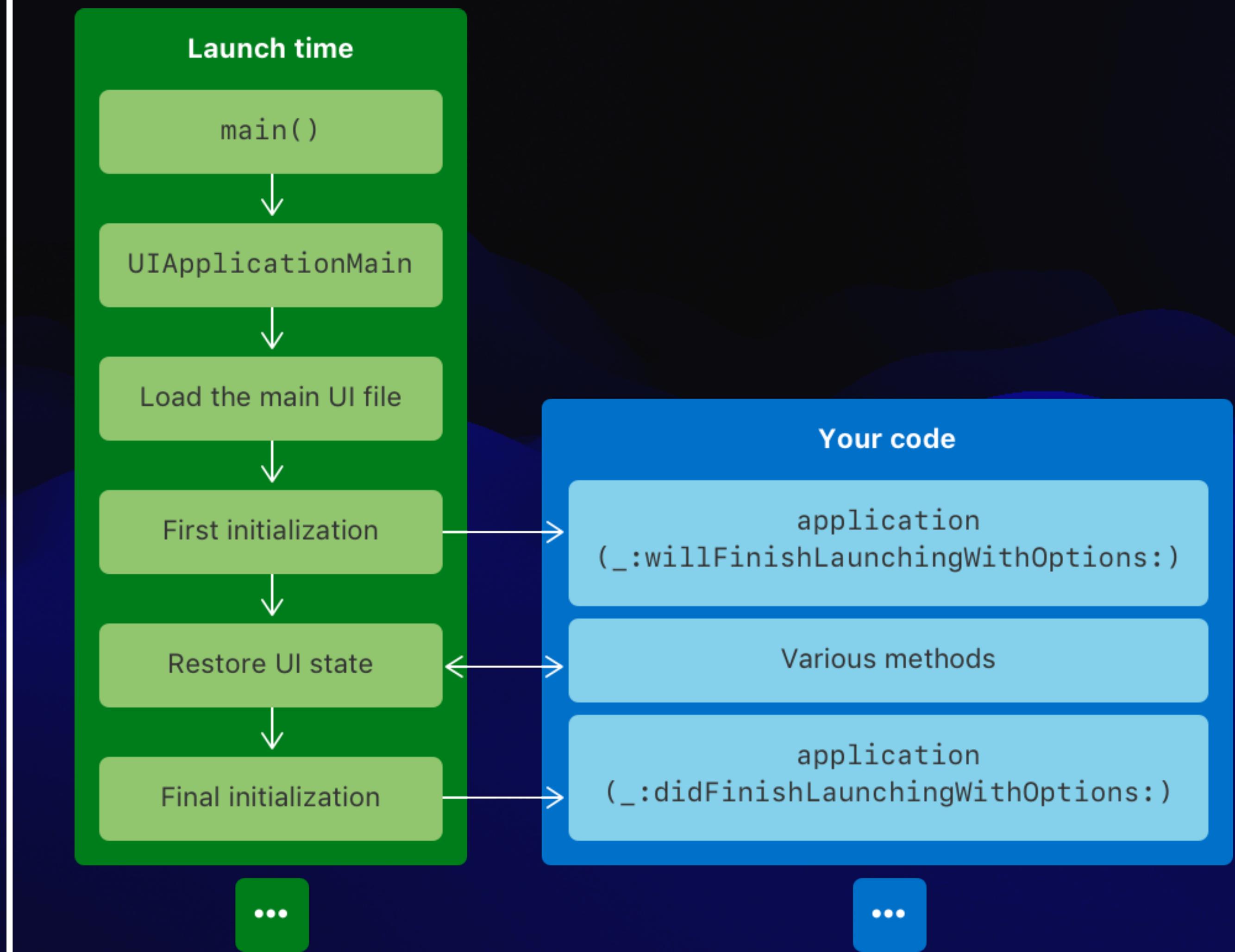
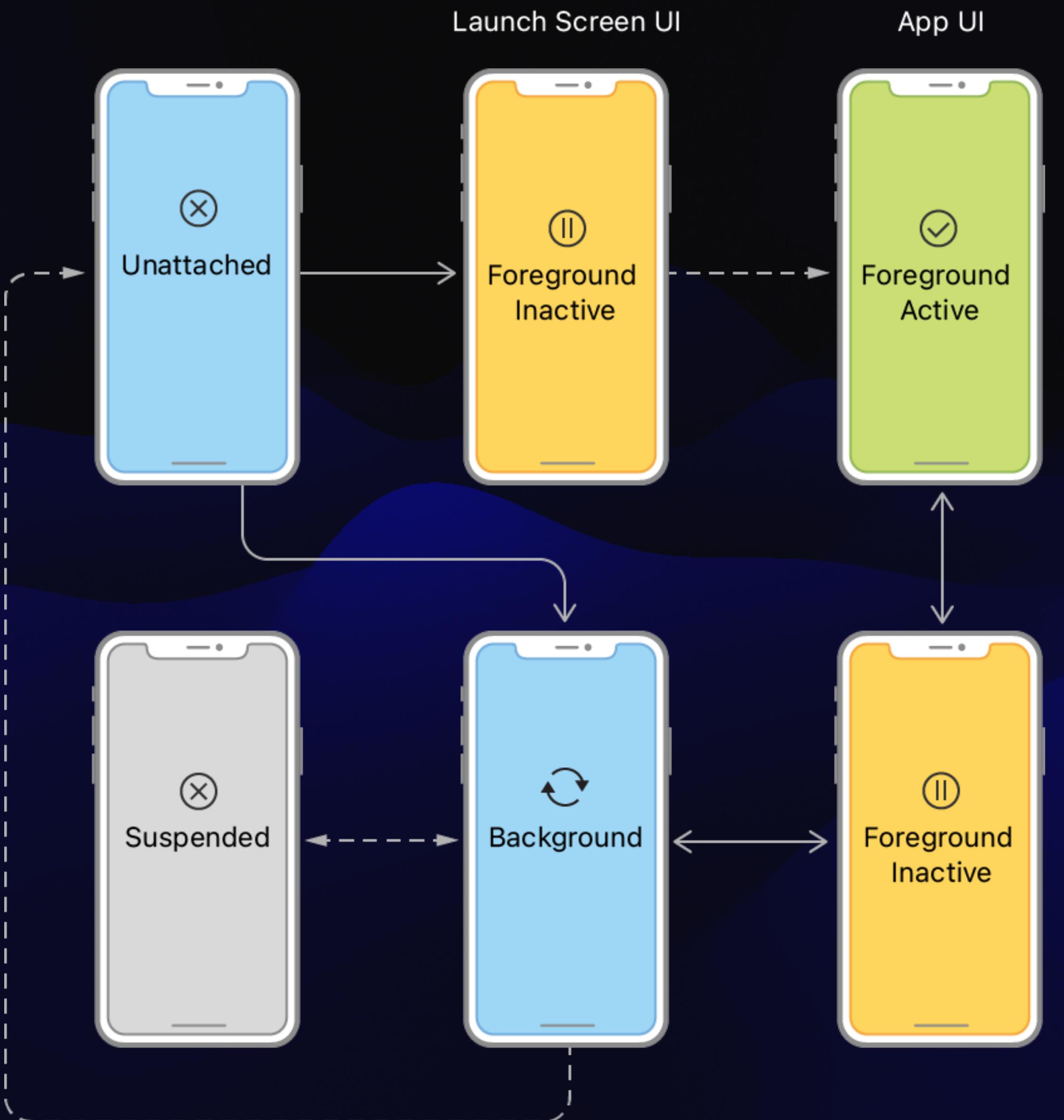
Der iOS Lifecycle + Human Interface Guidelines

- <https://developer.apple.com/ios/planning/>
- UI früher über Storyboards (visuell, UIKit), heute alles via Code (SwiftUI) (ab iOS 13)
- Apple legt hohen Wert auf UI / UX
- <https://developer.apple.com/design/human-interface-guidelines/designing-for-ios>
- App kann beim Review Prozess abgelehnt werden wenn nicht nach Designvorgaben
- Resourcen: <https://developer.apple.com/design/resources/#ios-apps>
- Programme wie Sketch, Adobe XD
- SF Symbols
- Accessibility (Translations, variable Textgröße, Screen reader, Dark Mode, etc)

Der iOS Lifecycle + Human Interface Guidelines

Fortsetzung

- https://developer.apple.com/documentation/uikit/app_and_environment/managing_your_app_s_life_cycle
- Jede App ist eine Sandbox die vom System verwaltet wird
- Alles geschieht über Delegates, Events, Callbacks, Completion Handlers und Closures => Asynchron
- Seit Swift 5.5 (WWDC 2021) auch endlich async await Syntax möglich
- Läuft auf mehreren Threads gleichzeitig (DispatchQueues)
UI Updates müssen immer auf der Main Queue ausgeführt werden
- Keine klassische „Main“ Methode
- Viele OS Methoden / Variablen können überschrieben werden
- https://developer.apple.com/documentation/uikit/app_and_environment/responding_to_the_launch_of_your_app/about_the_app_launch_sequence



Swift + SwiftUI

- Nachfolger von Objective-C
- Aktuell Version 5.8 (ab Xcode 15 mit visionOS dann 5.9)
- Ziel: Sicher, schnell, einfach zu nutzen
- Autom. Speichermanagement, keine Semikolons, sehr ähnlich zu Kotlin
- Plattformunabhängig (läuft auf macOS, Linux und Windows)
- <https://www.swift.org/about/>

Swift + SwiftUI

- <https://docs.swift.org/swift-book/documentation/the-swift-programming-language/guidedtour/>
- var und let (Konstante)
- Typ ist implizit, geht aber auch explizit: var name: String = „bla“
- Public Properties mit getter und setter (sogar willSet und didSet), sowie Computed Properties (Variable mit Funktion dahinter)
- print(„\(\var)“)
- [] = Array, [:] = Dictionary
- If / else - switch case (fallthrough statt break)
- for x in bla - auch über Dictionaries, for i in 0...4 (inkl. 4) oder 0..<4 (exkl. 4)
- while, Do repeat while
- class mit Zugriffsmodifizierer (public, private), init = Konstruktor, self == this, deinit = Destruktor, super = base
 - Ableiten möglich, override zum Methoden oder Variablen überschreiben
- Methoden = func, Return value am Ende mit -> zb: func Addiere(a: Int, b: Int) -> Int { return a + b }
- Können Nested sein, können mehrere Werte zurück geben (Tuple)
- Optionals? (Nullable)
 - If let oder guard let um optionals zu entpacken, ! Für force unwrap (nicht empfohlen)

Swift + SwiftUI

- Funktionen können Funktionen als Parameter bekommen oder zurück geben
 - Wenn diese Aufgerufen = Closure / Completion Handler
 - Kann mit @escaping auch über Context hinaus weiterleben
 - Immer [weak self] Referenzen verwenden um Retain Cycle zu lösen
- Anonyme Funktionen = Lambdas
- Enums werden mit „case“ geschrieben. Diese können Parameter haben
- Enums können auch Methoden haben
- Structs sind Value Types, classes Reference Types
- Async await und Tasks
- Protocol == Interface, kann Variablen und Methoden beinhalten —> Dependency Injection oder Delegates
- Klassen können mit „Extension“ erweitert werden
- nil == null
- Error Handling über Error Klassen und throws (Methodenrumpf) / throw Keyword. Auffangen via do { try } catch {}
 - Oder try? Um nil zu bekommen wenn fehlgeschlagen, try! Für force und nur try um Fehler weiter zu geben an Aufrufenden
- defer Statement wird am Ende einer Methode immer ausgeführt
- Generics gibt es ebenfalls

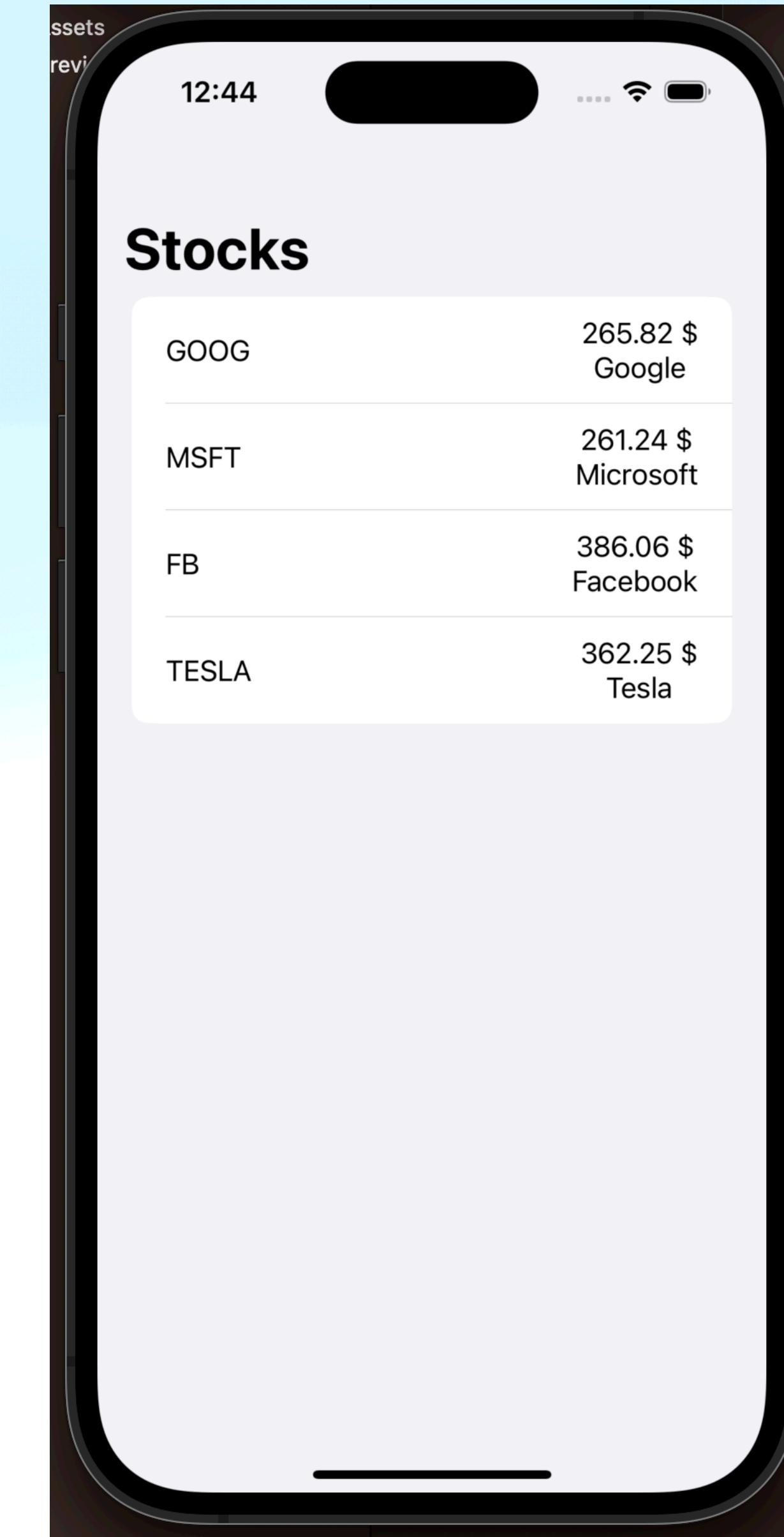
Swift + SwiftUI

- SwiftUI ist neue Art UI zu gestalten
- Alles ist eine View in einem struct
- Preview wird direkt in Xcode angezeigt
- Komplexe UIs können anhand verschachtelter Basiselemente oder eigener Structs realisiert werden
- Beispiele sind: VStack, HStack, ZStack, Spacer, Button, Text, Image, ...
- Combine Framework bzw. Property Wrapper werden benötigt um Views bei Änderungen von Variablen zu refreshen
 - Publisher / Subscriber, Reactive Programming
 - <https://developer.apple.com/documentation/combine>
- Kann animiert werden

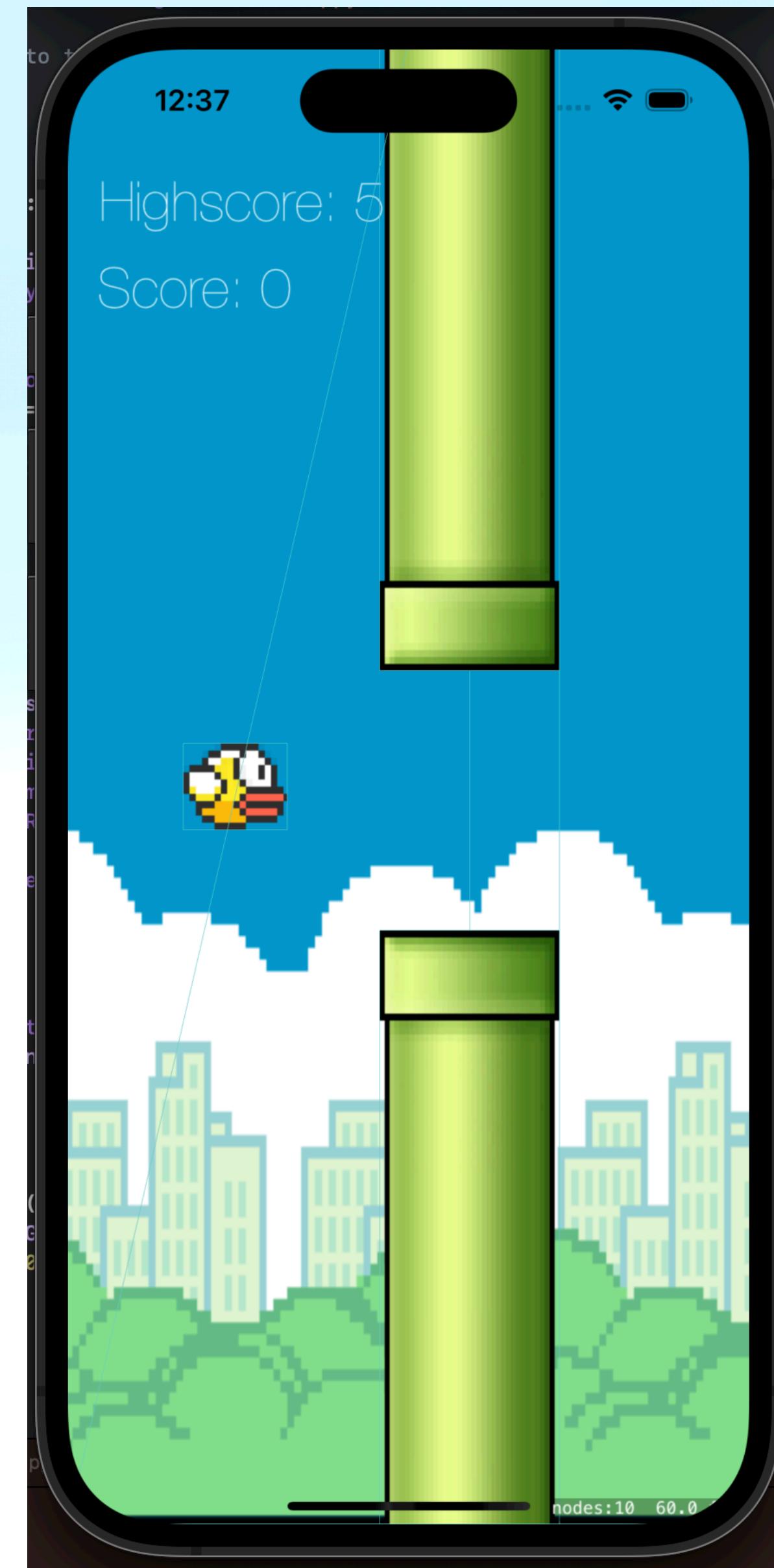
Was werden wir heute machen

- <https://github.com/AceKing47/SwiftSchoolDemo>
- Einführung in SwiftUI und MVVM Pattern (SwiftUIMVVM)
- Flappy Bird Clone (SwiftUI oder nicht) (FlappyClone oder FlappyCloneSwiftUI)
- Augmented Reality App (ARNumbers)
- AI App welche Blumen klassifiziert
(Machine Learning App, Training geht nur am Mac und benötigt Create ML)
- Alle Apps können am Mac in einem Simulator, oder direkt am iPad oder angesteckten iPhone getestet werden

SwiftUI MVVM



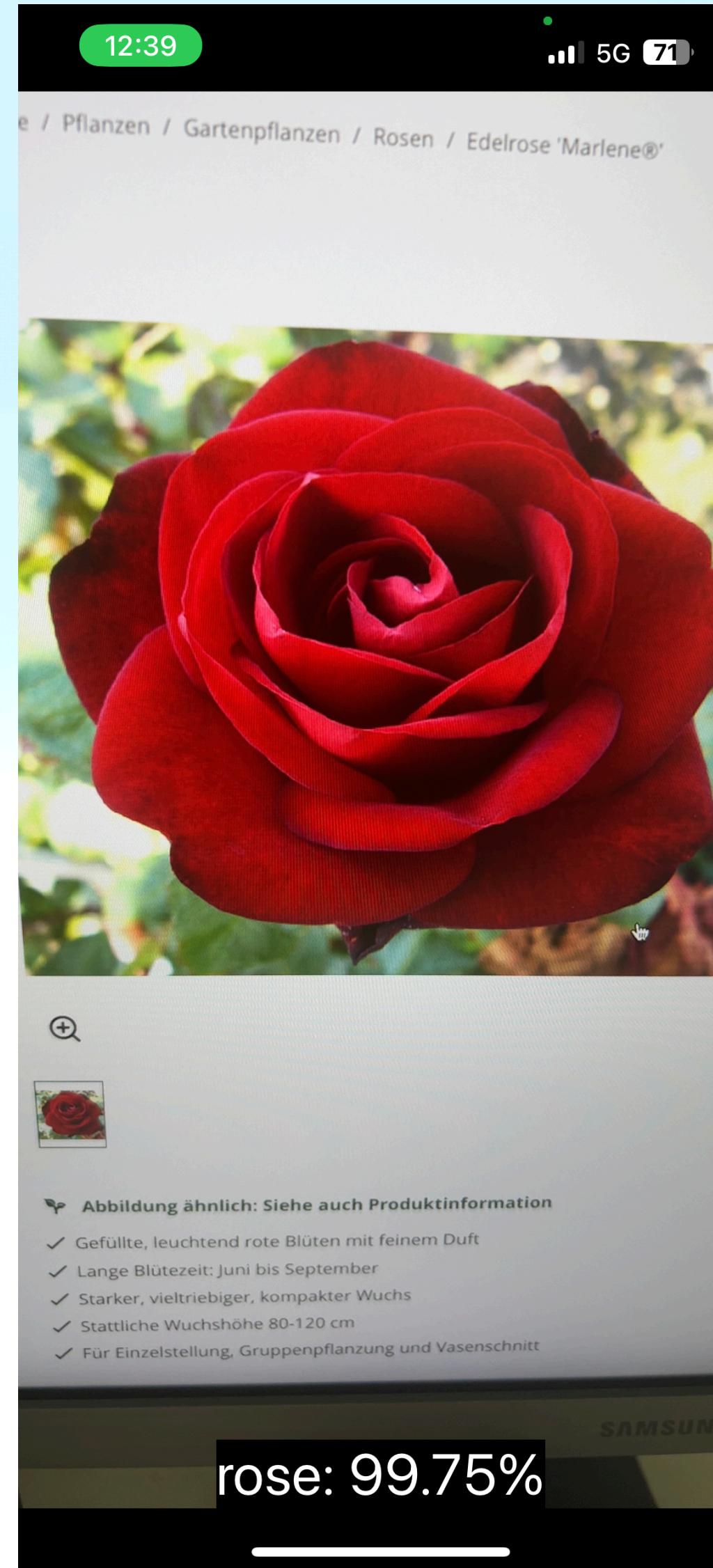
Flappy Bird Clone



AR Numbers



Flower Classifier





LIVE PROGRAMMING

AppStore

- Verwalten der Apps geht über AppStore Connect
<https://appstoreconnect.apple.com>
- Dort Bilder, Beschreibung, Version, Veröffentlich (autom. / manuell)
- Hochladen direkt über Xcode oder einer API (siehe nächste Folie)
- Reviews direkt in Xcode einsehbar (Window -> Organizer)
- TestFlight eigene App für Beta Tests (ebenfalls in Appstore Connect hinzuzufügen)
- Jede App wird von Apple reviewed und kann nur nach deren Freigabe released werden
 - Dadurch herrscht eine ständig hohe Sicherheit und Qualität im AppStore
- Apple behält sich 30% vom Umsatz ein
- Da ich selbst noch keine App im Store habe, kann ich es nicht herzeigen

Release und Automatisierung

- Fastlane - verwendet eine Apple API und kann Tests ausführen, Pakete bauen und sogar in den AppStore pushen
 - <https://docs.fastlane.tools/getting-started/ios/setup/>
- GitLab - kann automatisiert Schritte nacheinander in Pipelines ausführen, die App zb. automatisiert testen und bei Erfolg via Fastlane einen Release anstoßen
 - <https://docs.gitlab.com/ee/ci/pipelines/>
- All das geht auch direkt über Xcode mittels Xcode Cloud (ab 2024 kostenpflichtig)
 - <https://developer.apple.com/xcode-cloud/>

Zusammenfassung und Fragen

- Was zeichnet Apple aus
- Deep Dive in das Apple Eco System und den Lifecycle von iOS Apps
- Welche Tools und Hilfen gibt es für die Entwickler
- Entwicklung eigener iOS Apps mittels Xcode, Create ML und Swift
- Release über den App Store, Testen via TestFlight
- Falls es noch Fragen gibt, stehe ich gerne zur Verfügung
- Ich bedanke mich nochmals für das Interesse und die Aufmerksamkeit und hoffe, Sie konnten einiges aus dem heutigen Workshop mitnehmen