



APPS für iOS 10 professionell entwickeln

- // Sauberen Code schreiben
mit Swift 3 und Objective-C
- // Stabile Apps für iPhone und
iPad programmieren
- // Techniken & Methoden
von Grund auf verstehen



Bleiben Sie auf dem Laufenden!



Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter



www.hanser-fachbuch.de/newsletter



Hanser Update ist der IT-Blog des Hanser Verlags mit Beiträgen und Praxistipps von unseren Autoren rund um die Themen Online Marketing, Webentwicklung, Programmierung, Softwareentwicklung sowie IT- und Projektmanagement. Lesen Sie mit und abonnieren Sie unsere News unter



www.hanser-fachbuch.de/update



Thomas Sillmann

Apps für iOS 10 professionell entwickeln

Sauberen Code schreiben
mit Swift 3 und Objective-C

Stabile Apps für iPhone und iPad
programmieren

Techniken & Methoden
von Grund auf verstehen

HANSER

Der Autor:

Thomas Sillmann, Aschaffenburg

www.thomassillmann.de

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autor und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.



Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2017 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Sylvia Hasselbach

Copy editing: Walter Saumweber, Ratingen

Herstellung: Irene Weillhart

Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de

Umschlagrealisation: Stephan Rönigk

Gesamtherstellung: Kösel, Krugzell

Printed in Germany

Print-ISBN: 978-3-446-45073-8

E-Book-ISBN: 978-3-446-45206-0

Für meinen Vater, Ernst Sillmann.

„Bis zum Mond und wieder zurück haben wir uns lieb.“

Inhalt

Vorwort	XXI
Danksagung	XXV
1 Über iOS	1
1.1 Was ist iOS?	1
1.1.1 iOS und macOS	2
1.1.2 Besonderheiten der iOS-Plattform	3
1.2 iOS für Entwickler	4
1.2.1 Hardware für Entwickler	4
1.2.2 Software für Entwickler	6
1.2.3 Das Apple Developer Program	6
1.3 Der Aufbau von iOS	8
1.3.1 Die vier Schichten von iOS	8
1.4 Die perfekte iOS-App	10
1.4.1 iOS Human Interface Guidelines	11
2 Die (bisherige) Programmiersprache – Objective-C	13
2.1 Über Objective-C und objektorientierte Programmierung	13
2.2 Grundlagen der Programmierung	14
2.2.1 Objekte	14
2.2.2 Primitive Datentypen	14
2.2.3 Variablen	15
2.2.4 Operatoren	17
2.2.5 Abfragen und Schleifen	17
2.2.6 Kommentare	22
2.3 Aufbau einer Klasse	23
2.3.1 Die Header-Datei	23
2.3.2 Die Implementation-Datei	25
2.3.3 Los geht's: Unsere erste Klasse!	26

2.4	Methoden	30
2.4.1	Aufbau von Methoden	30
2.4.2	Methoden in Header- und Implementation-Dateien einer Klasse	32
2.4.3	Implementierung von Methoden	34
2.4.4	Methoden aufrufen	36
2.4.5	Klassen- und Instanzmethoden	37
2.5	Instanzvariablen	38
2.6	Properties	40
2.6.1	Aufbau einer Property	40
2.6.2	Die Punktnotation	42
2.6.3	Optionen	43
2.6.4	Direktzugriff auf Properties	45
2.6.5	Setter und Getter überschreiben	47
2.7	Konstanten	49
2.7.1	Deklaration von Konstanten	50
2.8	Namenskonventionen	51
2.8.1	Klassen	51
2.8.2	Methoden	51
2.8.3	Properties	52
2.9	Strukturen	52
2.9.1	enum	52
2.9.2	typedef	53
2.10	Initialisierung von Objekten	54
2.10.1	alloc und init	55
2.10.2	Zeiger	57
2.11	init im Detail	58
2.11.1	Erstellen eigener init-Methoden	60
2.11.2	Designated Initializer	61
2.12	Vererbung	63
2.12.1	Methoden der Superklasse überschreiben	65
2.13	Kategorien	67
2.13.1	Aufbau von Kategorien	67
2.13.2	Kategorien in Xcode erstellen	68
2.13.3	Verwenden von Kategorien	70
2.14	Erweiterungen	70
2.14.1	Aufbau von Erweiterungen	71
2.14.2	Erweiterungen innerhalb der Implementation-Datei	71
2.14.3	Erweiterungen in Xcode erstellen	72
2.15	Protokolle	73
2.15.1	Aufbau von Protokollen	74
2.15.2	Zuweisen eines Protokolls zu einer Klasse	75
2.15.3	Vererbung in Protokollen	76
2.15.4	Protokolle in Xcode erstellen	76

2.16	#import und @class	78
2.16.1	#import	78
2.16.2	@class	79
2.17	Blöcke	80
2.17.1	Was sind Blöcke?	81
2.17.2	Aufbau eines Blocks	81
2.17.3	Zuweisen eines Blocks zu einer Variablen	82
2.17.4	Nutzen eines Blocks als Parameter einer Methode	83
2.17.5	Blöcke als Properties	85
2.17.6	Blockvariablen	85
2.17.7	Globale Blöcke	86
2.18	Singletons	87
3	Der Neue im Club – Swift	89
3.1	Programmierst du noch oder swifst du schon?	89
3.1.1	Über Swift	89
3.1.2	Voraussetzungen zur Nutzung von Swift	90
3.1.3	Swift und Objective-C	90
3.1.4	Playgrounds	91
3.2	Grundlagen der Programmierung	93
3.2.1	Swift Standard Library und Fundamental Types	93
3.2.2	Variablen und Konstanten	95
3.2.3	Operatoren	97
3.2.4	Abfragen und Schleifen	98
3.2.5	Kommentare	105
3.2.6	print	106
3.3	Fundamental Types und Swift Standard Library im Detail	108
3.3.1	Strings	108
3.3.2	Arrays	111
3.3.3	Dictionaries	116
3.3.4	Any und AnyObject	120
3.4	Aufbau einer Klasse	120
3.4.1	Erstellen einer Instanz einer Klasse	123
3.4.2	Zugriff auf Eigenschaften einer Klasse	123
3.5	Methoden	124
3.5.1	Methoden mit Rückgabewert	125
3.5.2	Methoden mit Parametern	126
3.5.3	Local und External Parameter Names	129
3.5.4	Methodennamen in Swift	131
3.5.5	Aufruf von Methoden einer Klasse	132
3.5.6	Zugriff auf andere Eigenschaften und Methoden einer Klasse	133
3.5.7	Klassen- und Instanzmethoden	134
3.5.8	Verändern von Parametern einer Methode mittels inout	136

3.6	Closures	137
3.6.1	Closures als Variablen und Konstanten	139
3.6.2	Closures als Parameter für Methoden	140
3.6.3	Kurzschreibweise für Closures als Parameter von Methoden	143
3.7	Properties	146
3.7.1	Computed Properties	147
3.7.2	Property Observers	150
3.7.3	Type Properties	151
3.8	Vererbung	152
3.8.1	Überschreiben von Eigenschaften und Methoden der Superklasse	154
3.8.2	Zugriff auf Eigenschaften und Methoden der Superklasse	155
3.9	Optionals	156
3.9.1	Deklaration von Optionals	156
3.9.2	Zugriff auf Optionals	157
3.10	Initialisierung	160
3.10.1	Schreiben von Initializern	161
3.10.2	Designated und Convenience Initializer	166
3.10.3	Initializer und Vererbung	167
3.10.4	Deinitialisierung	169
3.11	Type Casting	170
3.11.1	Typ prüfen	170
3.11.2	Downcasting	172
3.12	Enumerations	174
3.12.1	Zusätzliche Werte in Mitgliedern einer Enumeration speichern	175
3.13	Structures	177
3.14	Generics	178
3.14.1	Generic Function	180
3.14.2	Generic Type	182
3.15	Error Handling Model	184
3.16	Extensions	187
3.17	Protocols	188
3.17.1	Protocol Type	189
3.18	Access Control	191
4	Grundlagen der iOS-Entwicklung	193
4.1	Foundation-Framework	193
4.1.1	Die wichtigsten Klassen aus dem Foundation-Framework und ihre Funktionen	194
4.2	UIKit-Framework	199
4.3	Speicherverwaltung	199
4.4	Besonderheiten von Objective-C	203
4.4.1	Kurzschreibweisen zum Erstellen von Objekten	203

4.4.2	Vergleichen der Werte von verschiedenen Objekten	206
4.4.3	Schlüsselwörter zum Zusammenspiel mit Optionals	207
4.5	Besonderheiten von Swift	208
4.5.1	Zusammenspiel zwischen Fundamental Types und Foundation-Framework-Klassen	208
4.5.2	Playgrounds im Detail	208
4.6	Objective-C und Swift vereint	212
4.6.1	Objective-C-Code in Swift verwenden	213
4.6.2	Swift-Code in Objective-C verwenden	214
4.7	NSError	214
4.7.1	Eigene Methode mit NSError-Parameter erstellen	216
4.8	Dokumentation	217
4.8.1	Besonderheiten bei Methoden	218
4.8.2	Doxygen-Dokumentation in Xcode	220
4.9	Nebenläufigkeit mit Grand Central Dispatch	221
4.9.1	Parallel laufenden Code erstellen	222
4.10	Grundlegende Struktur einer App	224
4.10.1	main.m	224
4.10.2	Info.plist	225
4.10.3	App Delegate	225
4.11	Lebenszyklus einer iOS-App	226
4.11.1	Start einer App	226
4.11.2	Lebenszyklus einer App	227
4.11.3	Die Methoden des App Delegate	228
4.11.4	Start der App	229
4.12	Tipps für die tägliche Arbeit	231
4.12.1	Die netten Kleinigkeiten	231
4.12.2	Fast Enumeration in Objective-C	232
4.12.3	Type Casting in Objective-C	232
4.12.4	Xcode-Beispielprojekte	233
5	Die Entwicklungsumgebung – Xcode	235
5.1	Willkommen bei Xcode!	235
5.1.1	Was ist Xcode?	236
5.1.2	Interface Builder und Xcode – endlich vereint!	236
5.2	Arbeiten mit Xcode	237
5.2.1	Dateien und Formate eines Xcode-Projekts	237
5.2.2	Umgang mit Dateien und Ordnern in Xcode	242
5.3	Der Aufbau von Xcode	245
5.3.1	Die Toolbar	245
5.3.2	Die Navigation Area	247
5.3.3	Die Editor Area	250
5.3.4	Die Utilities Area	252
5.3.5	Die Debug Area	253

5.4	Einstellungen in Xcode	254
5.4.1	Anpassen von Xcode	254
5.4.2	General	254
5.4.3	Accounts	255
5.4.4	Behaviors	256
5.4.5	Navigation	256
5.4.6	Fonts & Colors	257
5.4.7	Text Editing	258
5.4.8	Key Bindings	258
5.4.9	Source Control	259
5.4.10	Components	260
5.4.11	Locations	260
5.5	Projekteinstellungen	261
5.5.1	Grundlagen	261
5.5.2	Einstellungen am Projekt	263
5.5.3	Einstellungen am Target	266
5.5.4	Einstellungen am Scheme	272
5.6	Grafiken und Asset-Bundles	275
5.7	Lokalisierung mit Localizable.strings	277
5.7.1	Grundlagen	277
5.7.2	NSString	277
5.7.3	Erstellen der Localizable.strings-Datei	278
5.7.4	Localized String mit Parameter	280
5.7.5	Alle Localized Strings automatisch auslesen	281
5.8	Der iOS-Simulator	282
5.8.1	Grundlagen	282
5.8.2	Funktionen und Möglichkeiten des Simulators	282
5.8.3	Performance und Einschränkungen des Simulators	286
5.9	Dokumentation	286
5.9.1	Nichts geht über die Dokumentation!	286
5.9.2	Das Documentation-Window	289
5.9.3	Direktes Aufrufen der Dokumentation aus Xcode heraus	292
5.10	Devices	293
5.11	Organizer	295
5.12	Debugging in Xcode	298
5.12.1	Was ist Debugging?	298
5.12.2	Die Debug Area	298
5.12.3	Die Arbeit mit dem Debugger – NSLog und Breakpoints	299
5.12.4	Debug Navigator	308
5.13	Refactoring	309
5.13.1	Grundlagen	309
5.13.2	Refactoring-Funktionen in Xcode	310

5.14	Instruments	312
5.14.1	Über Instruments	312
5.14.2	Nächste Schritte	316
5.15	Tipps für die tägliche Arbeit	316
5.15.1	Man lernt immer was dazu!	316
5.15.2	Code Snippets	317
5.15.3	Open Quickly	318
5.15.4	Caller einer Methode feststellen	319
5.15.5	Speicherorte für Ordner und Dateien ändern	320
5.15.6	Shortcuts für die Navigation Area	320
5.15.7	Run Without Building	321
5.15.8	Clean Build	322
6	MVC – Model-View-Controller	323
6.1	MVC ... was?	323
6.2	MVC in der Praxis	325
6.3	Kommunikation zwischen Model und Controller	325
6.3.1	Key-Value-Observing	326
6.3.2	Notifications	332
6.4	Kommunikation zwischen View und Controller	335
6.4.1	Target-Action	335
6.4.2	Delegation	337
7	Die Vielfalt der (View-)Controller	339
7.1	Alles beginnt mit einem View-Controller	339
7.2	UIViewController – die Mutter aller View-Controller	341
7.2.1	Wichtige Methoden von UIViewController	343
7.2.2	UIView – fester Bestandteil eines jeden UIViewControllers	345
7.3	View-Controller-Hierarchien	346
7.4	View-Controller erstellen mit dem Interface Builder	348
7.4.1	View-Controller mit NIB-File	349
7.5	Storyboards	379
7.5.1	Über Storyboards	379
7.5.2	Das Storyboard-Projekt	380
7.5.3	Die Klasse UIStoryboard	390
7.5.4	Segues	392
7.5.5	Zugriff über den App Delegate	395
7.5.6	Quo vadis – Storyboard oder NIB-File?	396
7.6	Auto Layout	397
7.6.1	Setzen und Konfigurieren von Constraints	397
7.6.2	Constraints bearbeiten und weiter anpassen	399
7.6.3	„Optimale“ Constraints automatisch setzen lassen	401

7.7	UIViewController und seine Subklassen	402
7.7.1	UINavigationController	403
7.7.2	UITabBarController	409
7.7.3	UITableViewController	413
7.7.4	UICollectionViewController	420
7.7.5	UISplitViewController	421
8	Views erstellen und gestalten	425
8.1	Über Views in iOS	425
8.2	UIView – die Mutter aller Views	425
8.3	Arbeiten mit UIView	426
8.3.1	Programmatisches Erstellen einer UIView	426
8.3.2	View-Hierarchien	428
8.3.3	Weiterführendes zu UIView	432
8.4	Views erstellen mit dem Interface Builder	433
8.4.1	Grundlagen	433
8.4.2	View-Klasse mit NIB-File erstellen	434
8.4.3	Beliebiges NIB-File laden und verwenden	438
8.4.4	NIB-File nachträglich erstellen	439
8.4.5	Unterschiedliche NIB-Files für iPhone und iPad erstellen	441
8.5	Die wichtigsten Views und ihre Funktionen	443
8.5.1	Grundlagen	443
8.5.2	UILabel	443
8.5.3	UIButton	443
8.5.4	UISwitch	444
8.5.5	UISegmentedControl	444
8.5.6	UITextField	444
8.5.7	UIImageView	445
8.5.8	UIPickerView	445
8.5.9	UIDatePicker	446
8.5.10	UIWebView	446
8.5.11	UIMapView	447
8.5.12	UIScrollView	447
8.5.13	UITextView	448
8.5.14	UITableView	449
8.5.15	UICollectionView	449
8.5.16	Wichtig und unerlässlich: die Dokumentation!	449
8.5.17	Views und der Interface Builder	450
8.6	Die Grundlage gut gestalteter Views	450
9	Das Model und die Datenhaltung	453
9.1	Die Logik Ihrer App	453
9.2	Benutzereinstellungen sichern und nutzen	454
9.2.1	Über UserDefaults	454
9.2.2	Standardeinstellungen festlegen	457

9.3	Zugriff auf das Dateisystem	457
9.3.1	Das Dateisystem von iOS	457
9.3.2	FileManager	459
9.3.3	File-Sharing-Funktion nutzen	466
9.4	Core Data	467
9.4.1	Datenbankverwaltung mit Core Data	467
9.4.2	Wie funktioniert Core Data?	468
9.4.3	Die Klassen und Bestandteile von Core Data	469
9.4.4	Aufbau eines Standard-Core Data Stacks	470
9.4.5	Der Core Data-Editor	473
9.4.6	Erstellen eines neuen Managed-Objects	481
9.4.7	Löschen eines Managed-Objects	482
9.4.8	Laden von Managed-Objects	482
9.4.9	Was kommt als Nächstes?	484
10	Local und Push Notifications	485
10.1	Was sind Notifications?	485
10.2	Registrieren von Notification Types	487
10.3	Registrieren von Notification Categories und Actions	491
10.3.1	Erstellen einer Action	491
10.3.2	Erstellen einer Kategorie	493
10.3.3	Registrieren von Kategorien	495
10.3.4	Reagieren auf eine Action	496
10.4	Local Notifications	498
10.4.1	Konfiguration des Alerts	498
10.4.2	Konfiguration des Sounds	500
10.4.3	Konfiguration des Badge Values	501
10.4.4	Konfiguration von Audio, Bildern und Videos	501
10.4.5	Speichern zusätzlicher Informationen in einer Local Notification ...	503
10.4.6	Festlegen des Ausführungsereignisses	504
10.4.7	Erstellen von Notification Requests	508
10.4.8	Registrieren von Local Notifications im System	509
10.4.9	Abbrechen bereits registrierter Local Notifications	510
10.4.10	Reagieren auf den Erhalt einer Notification bei aktiver App	510
10.5	Push Notifications	512
10.5.1	Versand von Push Notifications	513
10.5.2	Erstellen einer Push Notification	517
10.5.3	Quality of Service	520
11	Extensions	521
11.1	Verfügbare Typen von Extensions	521
11.2	Erstellen von Extensions in Xcode	524
11.3	Funktionsweise einer Extension	527
11.4	Wichtige Klassen und Objekte	528

11.5	Unterstützte Dateitypen für Share- und Action-Extensions festlegen	529
11.6	Action Extension	530
11.6.1	Action mit User Interface	530
11.6.2	Action ohne User Interface	531
11.7	Content Blocker Extension	532
11.7.1	Konfiguration eines Content Blockers	533
11.7.2	Aktualisieren eines Content Blockers	536
11.7.3	Die Klasse ContentBlockerRequestHandler	536
11.8	Custom Keyboard	537
11.8.1	Erstellen eines Custom Keyboards	537
11.8.2	Arbeiten mit der Klasse UIInputViewController	538
11.8.3	Bearbeiten und Setzen von Text	540
11.8.4	Mehrsprachige Keyboards	541
11.9	Document Provider	541
11.9.1	Document Provider-Extension	542
11.9.2	File Provider	545
11.9.3	Document Provider aufrufen	547
11.10	iMessage Extension	549
11.10.1	Aufbau und Funktionsweise der iMessage Extension	549
11.10.2	Entwicklung einer iMessage Extension	553
11.11	Intents Extension	564
11.11.1	Domains und Intents	565
11.11.2	Bestandteile einer Intents Extension	566
11.11.3	Funktionsweise einer Intents Extension	567
11.11.4	Übersicht über verfügbare Intents	576
11.11.5	Voraussetzungen zur Verwendung von Siri in einer Intents Extension	578
11.11.6	Erweitern von Siris Vokabular	581
11.11.7	Testen einer Intents Extension	587
11.12	Intents UI Extension	588
11.12.1	Vorbereitung der Intents UI Extension	589
11.12.2	Konfiguration des View-Controllers	590
11.12.3	Default-Layout bei Maps und Messaging deaktivieren	592
11.13	Notification Content Extension	592
11.13.1	UNNotificationContentExtension	592
11.13.2	Konfiguration der Info.plist-Datei	594
11.14	Notification Service Extension	595
11.15	Photo Editing Extension	596
11.15.1	Festlegen der unterstützten Typen zur Bearbeitung	600
11.16	Share Extension	601
11.17	Shared Links Extension	602
11.17.1	Erstellen eines NSExtensionItem	603
11.17.2	NSExtensionItem als Shared Link bereitstellen	604

11.18	Sticker Pack Extension	605
11.18.1	Erstellen einer Sticker Pack Extension	606
11.18.2	Komplexere Sticker Pack Extensions erstellen	610
11.19	Today Extension	612
11.19.1	Today Extension testen	614
11.20	Watch App	615
11.21	App Groups	615
11.21.1	Registrieren einer App Group im Apple Developer Portal	616
11.21.2	Registrieren einer App Group innerhalb einer App	617
11.21.3	Zugriff auf eine App Group	618
12	App-Entwicklung für die Apple Watch	621
12.1	Apples neues großes Ding: Die Apple Watch	621
12.2	Möglichkeiten, Einschränkungen, Unterschiede	622
12.3	Das WatchKit SDK	624
12.3.1	WKInterfaceController	625
12.3.2	WKInterfaceObject	625
12.3.3	WKExtensionDelegate	626
12.3.4	Weitere Klassen	626
12.4	Aufbau und Funktionsweise von Apple Watch-Apps	627
12.4.1	iPhone-App	627
12.4.2	WatchKit Extension	627
12.4.3	WatchKit App	628
12.4.4	Verbindung von WatchKit Extension und WatchKit App	628
12.4.5	Notification Scene	628
12.4.6	Complications	629
12.5	Erstellen einer WatchKit App mitsamt WatchKit Extension	629
12.5.1	Dateien der WatchKit Extension	632
12.5.2	Dateien der WatchKit App	633
12.5.3	Ausführen und Testen der Apple Watch-App	634
12.6	Lebenszyklus einer WatchKit App	635
12.7	Der WKInterfaceController im Detail	637
12.7.1	Initialisierung	637
12.7.2	Activation Events	639
12.7.3	Setzen des Titels	639
12.7.4	Ein- und Ausblenden von Interface-Controllern	640
12.7.5	Umsetzen eines Navigation Stacks	642
12.7.6	Reaktion auf Storyboard-Events	643
12.7.7	Weitere Attribute	644
12.7.8	Weitere Funktionen von WKInterfaceController	645
12.8	Arbeiten mit dem Interface-Storyboard einer WatchKit App	645
12.8.1	Erstellen und Konfigurieren eines WKInterfaceController	646
12.8.2	Hinzufügen und Konfigurieren von Interface-Elementen	648

12.8.3	Positionierung und Anordnung von Interface-Elementen	649
12.8.4	Ändern der Größe von Interface-Elementen	649
12.8.5	Unterschiedliche Konfigurationen für verschiedene Apple Watch-Größen	651
12.8.6	Gruppierung von Interface-Elementen mittels WKInterfaceGroup	653
12.8.7	Verbindung von Storyboard und Code	655
12.8.8	Zusammenfassen mehrerer Interface-Controller zu einem page-based Interface	659
12.8.9	Erstellen und Konfigurieren von Segues	659
12.9	Erstellen von Tabellen	662
12.9.1	Hinzufügen einer Tabelle im Storyboard	662
12.9.2	Konfiguration einer Zelle	663
12.9.3	Konfiguration einer Tabelle	667
12.9.4	Verwenden verschiedener Zellen in einer Tabelle	669
12.9.5	Zellen hinzufügen und entfernen	672
12.9.6	Direkt zu einer bestimmten Zelle scrollen	673
12.9.7	Aktuelle Anzahl an Zellen auslesen	673
12.9.8	Auf die Auswahl einer Zelle reagieren	674
12.9.9	Segues von Zellen einer Tabelle über das Storyboard konfigurieren	675
12.10	Erstellen von Menüs	676
12.10.1	Erstellen eines Menüs im Storyboard	677
12.10.2	Erstellen eines Menüs im Code	681
12.10.3	Fazit: Menüerstellung im Storyboard oder im Code?	683
12.10.4	Mischen von Menüpunkten aus Storyboard und Code	684
12.11	Eingabe von Text	684
12.12	Notification Scene	686
12.12.1	Short-Look und Long-Look Interface	687
12.12.2	Long-Look Interface im Detail	688
12.12.3	Erstellen eigener Notification Scenes	689
12.12.4	Testen einer Notification Scene	697
12.13	Complications	697
12.13.1	Was sind Complications?	698
12.13.2	Das ClockKit Framework	698
12.13.3	Aufbau und Bestandteile von Complications	698
12.13.4	Vorbereiten des eigenen Projekts	702
12.13.5	Entwicklung einer Complication	704
12.13.6	Bereitstellen der Complication mittels CLKComplicationDataSource	709
12.14	Kommunikation und Datenaustausch zwischen iOS und watchOS	712
12.14.1	Watch Connectivity	713
12.15	Was sonst noch zu sagen und zu beachten ist	718

13	Tests	721
13.1	Unit-Tests	721
13.1.1	Aufbau und Funktionsweise von Unit-Tests	726
13.1.2	Aufbau einer Test-Case-Klasse	728
13.1.3	Neue Test-Case-Klasse erstellen	730
13.1.4	Ausführen von Unit-Tests	732
13.1.5	Was sollte ich eigentlich testen?	734
13.2	Performance-Tests	734
13.3	UI-Tests	736
13.3.1	Klassen für UI-Tests	737
13.3.2	Aufbau von UI-Test-Klassen	740
13.3.3	Automatisches Erstellen von UI-Tests	740
13.3.4	Einsatz von UI-Tests	741
13.4	Test-Driven Development	741
14	Versionierung	743
14.1	Über Versionskontrolle	743
14.2	Basisfunktionen und -begriffe von Git	744
14.2.1	Begriffe	744
14.2.2	Funktionen	744
14.3	Source Control in Xcode	746
14.4	Version Editor und Source Control	750
15	Veröffentlichung im App Store	753
15.1	Zertifikate, Provisioning Profiles und Ihre App	753
15.1.1	Certificates, IDs & Profiles	755
15.1.2	Erstellen von...	757
15.2	Testen auf dem eigenen Endgerät	770
15.2.1	Setzen des Teams	770
15.2.2	Auswahl Ihres iOS-Geräts	770
15.3	iTunes Connect und Veröffentlichung im App Store	772
15.3.1	Vorbereiten der App in iTunes Connect	774
15.3.2	Upload der App in den App Store	777
15.3.3	Wie geht es weiter?	778
Index		779

Vorwort

iOS ist und bleibt für Entwickler ein spannendes Feld, nicht zuletzt, da Apple mit dem App Store einen Weg geschaffen hat, mit dem auch einzelne Entwickler Software für einen internationalen Markt verbreiten können, ohne dass sie sich um Dinge wie Bezahlsysteme, Abrechnungen und Download Gedanken machen müssen. Ich selbst habe mich aufgrund dessen vor über fünf Jahren für die iOS-Entwicklung begeistern lassen und habe bis heute nichts von der Faszination für diese spannende und innovative Plattform verloren.

Beim Einstieg in die iOS-Entwicklung habe ich eines aber schmerzlich vermisst: einen einfachen, übersichtlichen und professionellen Einstieg. Ich habe mich mit viel Literatur auseinandergesetzt, war in vielen Foren unterwegs und habe schlicht und einfach viel ausprobiert. Da vieles von diesen Anfängen aber sehr umständlich oder im Nachhinein betrachtet sogar gänzlich falsch war, hat mich das viel Zeit und Lehrgeld gekostet. Und ich habe mir oft gewünscht, man hätte mich von Beginn an an die Hand genommen und mir nicht nur gezeigt, *wie* ich Apps für iOS entwickle, sondern wie ich *gute und professionelle* Apps entwickle, welche Besonderheiten, Best Practices und Design Patterns es gibt und wie ich effektiv und effizient mit der Entwicklungsumgebung arbeiten kann. Und dieser Wunsch hat den Grundstein für dieses Buch gelegt.

Dieses Buch vermittelt Ihnen alle essenziellen Grundlagen und Kenntnisse über die Entwicklung für iOS. Angefangen bei der Vorstellung des Betriebssystems selbst geht es weiter über die Programmiersprachen Objective-C und Swift, deren Struktur und jeweiligen Besonderheiten und all das, was sie ausmacht. Danach rückt die eigentliche Entwicklung für iOS in den Fokus. Sie erfahren alles über die wichtigsten Frameworks von Apple für die App-Entwicklung und lernen typische Best Practices kennen. Besonders wichtig ist hier auch die Vorstellung der Dokumentation, die für Sie als App-Entwickler Bestandteil Ihrer täglichen Arbeit sein wird. Denn letztlich beherbergt die Apple-Dokumentation alle Antworten auf die Fragen, wie Sie bestimmte Probleme in der iOS-Entwicklung angehen und welche Möglichkeiten Ihnen die einzelnen Frameworks von Apple liefern.

Nach der Vorstellung der Plattform und der Programmiersprache(n) geht es weiter mit der Entwicklungsumgebung Xcode, mit der wir unsere Apps für iOS entwickeln. Dabei war es mir besonders wichtig, den genauen Aufbau und die Struktur hinter Xcode vorzustellen sowie alle spannenden Möglichkeiten und Kniffe aufzuzeigen, die Xcode Ihnen bietet und Ihre tägliche Arbeit erleichtern. Auch der Umgang mit Grafiken oder die Lokalisierung Ihrer Anwendung sowie Debugging und Refactoring habe ich in dieses Kapitel mit aufgenommen.

Bis zu diesem Punkt habe ich mich rein mit den essenziellen Grundlagen beschäftigt und ich finde es wichtig, dass auch Sie diese Grundlagen verinnerlicht und verstanden haben, denn sie sind die Grundpfeiler für gute und erfolgreiche Apps. Dazu abschließend folgt im sechsten Kapitel die Vorstellung von MVC – Model-View-Controller. Dabei handelt es sich um eines der wichtigsten Design-Patterns in iOS und ist essenziell für die App-Entwicklung. Aufgrund dessen widme ich MVC ein eigenes Kapitel, stelle es im Detail vor und erkläre, wie und warum Sie es in Ihren eigenen Apps anwenden sollen.

Anschließend geht es im Speziellen um die Entwicklung für iOS und die Nutzung der Funktionen und Frameworks für Apple, unterteilt auf die drei Bereiche Controller, View und Model aus dem MVC-Pattern. Auch in diesen Kapiteln geht es darum, die grundlegenden Besonderheiten zu erläutern und aufzuzeigen, wie Sie mit den einzelnen Elementen arbeiten und diese in Ihren eigenen Apps verwenden. Sie lernen die wichtigsten Elemente aus den jeweiligen Bereichen kennen und erfahren, wie Sie selbstständig mit ihnen arbeiten können und worauf bei der jeweiligen Verwendung zu achten ist. Mit diesem Wissen gewappnet sind Sie imstande, sich selbst in neue Frameworks, Technologien und Themen anhand der Apple-Dokumentation einzuarbeiten und selbstständig Probleme zu lösen. Und genau das ist es, was einen guten und professionellen iOS-Entwickler ausmacht.

Kapitel 10 setzt sich im Detail mit den sogenannten Local und Push Notifications auseinander, die es Ihnen erlauben, Nachrichten aus Ihren Apps an Ihre Nutzer zu senden. Im darauffolgenden Kapitel erstelle ich Ihnen ergänzend dazu dann Extensions vor, die uns Entwicklern ganz neue und innovative Möglichkeiten an die Hand geben, unsere Apps zu erweitern und über das gesamte iOS-System heraus zugänglich zu machen.

Im Anschluss daran widme ich ein eigenes und umfangreiches Kapitel der Entwicklung für die Apple Watch. Die Apple Watch ist die neue spannende Plattform in Apples Ökosystem und ist – was die grundsätzliche App-Entwicklung betrifft – in vielen Dingen recht ähnlich zur iOS-Plattform. Da eine Apple Watch-App immer eine iPhone-App voraussetzt, war es nur passend, auch die Entwicklung für die Apple Watch in diesem Buch im Detail zu beleuchten und zu zeigen, wie Sie Ihre eigenen iPhone-Apps um ein Pendant für die Apple Watch erweitern können. Sie lernen alle Möglichkeiten und Einschränkungen kennen und werden so in die Lage versetzt, selbst mit der Entwicklung eigener Apple Watch-Apps zu beginnen.

Anschließend folgen die Themen Tests und Versionsverwaltung mit Git, die in jeder neuen Xcode-Version mehr und mehr in die Entwicklungsumgebung integriert und unterstützt werden. Ich zeige Ihnen dabei, was Unit-Tests sind, warum Sie sie in Ihren Apps verwenden sollten, wie Sie Unit-Tests schreiben und wie Sie Xcode in Sachen Unit-Tests unterstützt und Ihnen unter die Arme greift. Auch UI-Tests finden dabei Beachtung. Bei der Versionsverwaltung mit Git erfahren Sie alles über die integrierten Funktionen zur Arbeit mit Git in Xcode und wie Sie Änderungen im Code verfolgen und nachvollziehen können.

Zu guter Letzt geht es noch – wie könnte es anders sein? – um die Veröffentlichung Ihrer Apps im App Store und die integrierten Tools in Xcode, die Ihnen bei diesem Prozess unter die Arme greifen. Insbesondere erfahren Sie hier etwas über die Erstellung und Verwaltung Ihrer Apps in Apples Developer Program.

Bei allen diesen Themen soll dieses Buch Sie unterstützen und Ihnen die Grundlagen und das essenzielle Praxiswissen vermitteln und mit auf den Weg geben. Es soll Ihnen nicht Beispielprojekte aufzeigen und deren Verhalten und Eigenschaften erklären (davon gibt es

nämlich von Apple selbst mehr als genug), sondern Ihnen das nötige Wissen mitgeben, um Sie in die Lage zu versetzen, Problemstellungen selbstständig zu lösen und zu verstehen, wie Sie gute und professionelle iOS-Apps entwickeln. Denn wenn Sie diesen Status erreicht haben, können Sie darauf aufbauen, experimentieren und eigene spannende und innovative iOS-Projekte umsetzen. Und ich bin gespannt, welche großartigen Apps wir von Ihnen erwarten dürfen.

Ich wünsche Ihnen viel Freude beim Lesen dieses Buches und viel Erfolg mit all Ihren iOS-Projekten.

Thomas Sillmann



ios-buch.thomassillmann.de

Unter dieser Adresse finden Sie zusätzliche Informationen und Services, die ich ergänzend zu diesem Buch anbiete. Dazu gehören vollständige Code-Beispiele und Projekte, anhand derer Sie Ihr erlangtes Wissen testen und vertiefen können, sowie verschiedene Frameworks, die Sie direkt in eigene Projekte integrieren können. Ebenso stelle ich dort Lern-Videos zur App-Entwicklung bereit und berichte in Blog-Beiträgen über Neuerungen und Aktualisierungen; ein Besuch lohnt sich also. ☺

Danksagung

Für mich als Autor ist es eine ungemeine Freude und ein großartiges Glück, wenn ein Buch wie jenes, das Sie gerade in Händen halten, bereits in einer dritten Auflage erscheint und bereits viele zufriedene Leser gefunden hat. Als die erste Auflage des Buches im Jahr 2014 zu iOS 8 erschien, hätte ich niemals damit gerechnet, zwei Jahre später an einer dritten Auflage zu iOS 10 zu arbeiten. Genau so ist es aber gekommen, und hier sind wir also.

Nachdem ich Ihnen einiges über das Buch an sich und dessen Entstehung und Inhalte erzählt habe, möchte ich diese Stelle nutzen, kurz ein paar ausgewählten Menschen zu danken.

Da wäre Daniel Bocksteger, der all meine bisherigen Bücher rezensierte und mich dabei als Autor immens stolz und glücklich gemacht hat.

Da wäre meine ehemalige Lektorin Sieglinde Schärl vom Hanser Verlag, ohne die ich niemals da wäre, wo ich heute bin und die mich in meiner Tätigkeit als Autor bestärkt und immer aufs Beste unterstützt hat.

Da wäre meine jetzige Lektorin Sylvia Hasselbach, dank der diese dritte Auflage überhaupt erst zustande kam und die das Projekt so erst möglich gemacht hat.

Da wäre Irene Weilhart, die mir bei Fragen und Problemen während der Arbeiten am Manuskript immer zuverlässig, hilfsbereit und schnell zur Seite stand.

Da wäre Walter Saumweber, der dieses Buch so großartig und genau korrigiert hat, um es so fehlerfrei wie nur irgend möglich umzusetzen.

Und natürlich ist da meine Frau Michaela, die in jeder noch so stressigen Phase dieses Buchprojekts hinter mir stand und mir den Rücken stärkte, die meinen Problemen und Sorgen lauscht und immer die richtigen Worte findet, um mich voranzutreiben und den Glauben nicht verlieren zu lassen. Auch wenn es bisweilen finster um mich herum ist, du scheinst immer hell und leuchtest mir den Weg.

Und selbstverständlich sind da Sie, liebe Leserin, lieber Leser. Nur Ihnen habe ich es zu verdanken, heute primär als Autor tätig zu sein und mir damit einen persönlichen Traum zu erfüllen. Ich danke Ihnen für Ihr zahlreiches konstruktives Feedback und die vielen Verbesserungsvorschläge, die ich nach bestem Wissen und Gewissen versuche umzusetzen und in meine Bücher einfließen zu lassen.

Ohne Leser ist ein Autor nichts, und daher danke ich Ihnen von Herzen, das Sie hier bei mir sind und meinen Worten lauschen. Und nun wünsche ich Ihnen viel Spaß und Freude mit dieser dritten Auflage meines iOS-Buches und hoffe, es ist Ihnen ein guter Begleiter und hilft Ihnen bei Ihrer Tätigkeit als App-Entwickler!

■ 1.1 Was ist iOS?

Auch wenn diese Frage in der heutigen Zeit möglicherweise überflüssig erscheint (und auch in Anbetracht dessen, dass Sie sich dieses Buch gekauft haben), möchte ich zu Beginn doch zumindest kurz darauf eingehen, was eigentlich dieses iOS ist, für das ich mich – und Sie sich offensichtlich auch – als Entwickler so sehr interessiere. Dabei werde ich auch direkt den Spagat schlagen und Ihnen die Geräte vorstellen, auf denen iOS verfügbar ist, und beschreiben, wie sich das System im Laufe der Jahre entwickelt hat.

Zunächst einmal ist iOS ein Betriebssystem der Firma Apple. Seinen ersten großen Auftritt hatte es im Jahr 2007 zusammen mit der Vorstellung des allerersten iPhone, denn genau auf diesem Gerät lief und läuft bis heute iOS (auch wenn es damals noch iPhone OS hieß). Mit dem iPhone krempelte sich der Markt der Smartphones maßgeblich um und heutzutage sieht man Touch-Smartphones mit dem Bildschirm als Hauptbedienelement allerorten.

Nach mehreren Hardware-Sprüngen des iPhone folgte im Jahr 2010 das nächste iOS-Device von Apple: Das iPad, welches – ebenso wie das iPhone zuvor den Smartphone-Markt – nun den Tablet-Markt ordentlich aufmischte und bis heute den Quasistandard im Bereich Tablets setzt. Auch auf dem iPad läuft Apples Mobil-Betriebssystem iOS (dessen Namensänderung ebenfalls im Jahr 2010 mit dem Erscheinen des iPad von iPhone OS zu iOS erfolgte).

Darüber hinaus läuft iOS auch auf dem iPod touch. Alle Apps, die Sie für das iPhone entwickeln, sind prinzipiell ebenfalls auf dem iPod touch lauffähig, lediglich die zugrunde liegende Hardware unterscheidet sich ein wenig; Telefonieren ist beispielsweise mit dem iPod touch nicht möglich. So kann sich aber ein iPod touch durchaus als günstiges Testgerät für iOS-Apps anbieten (das iPhone spielt da nun mal doch in einer etwas anderen Preisklasse).

Und direkt zu Beginn noch eine kleine Randnotiz: Das von Apple im Jahr 2015 neu vorgestellte und überarbeitete Apple TV besitzt viele Parallelen zu iOS, was die App-Entwicklung betrifft. Zwar verfügt dieses neue Apple TV über ein eigenes Betriebssystem (bekannt unter dem passenden Namen tvOS), dieses besitzt aber viele Gemeinsamkeiten mit iOS. All das Wissen, das Sie sich mithilfe dieses Buches zur iOS-Entwicklung aneignen, können Sie somit auch als perfekte Grundlage für etwaige Apps für das Apple TV heranziehen.



Bild 1.1 iPhone und iPad sind die erfolgreichsten Geräte mit dem Betriebssystem iOS. Daneben verfügt auch Apples iPod touch über iOS als Betriebssystem.

1.1.1 iOS und macOS

Zusammengefasst lässt sich also einfach sagen: iOS ist das Betriebssystem von Apples iPhone-, iPad- und iPod touch-Familie. Sicherlich wissen Sie aber auch, dass Apple nicht nur iOS-Geräte entwickelt und veröffentlicht (auch wenn das aktuell das Geschäft ist, das Apple den größten Umsatz einbringt). Daneben gibt es noch – neben der Apple Watch und dem Apple TV, auf die ich an dieser Stelle ausnahmsweise (noch) nicht eingehe – die Mac-Familie, die Apples Produktplatte aus Notebooks und Desktop-PCs darstellt. Und besonders spannend ist hierbei, dass iOS zu großen Teilen auf macOS – dem Betriebssystem der Macs und ehemals unter dem Namen OS X bekannt – basiert. So sind viele Frameworks, mit denen wir in der iOS-Entwicklung arbeiten werden, unter macOS in derselben oder in einer leicht abgewandelten Form verfügbar. Das bedeutet umgekehrt auch, dass der Einstieg in die macOS-Entwicklung leichter fällt, wenn Sie bereits für iOS entwickelt haben – und umgekehrt. Das aber nur als kleine Randnotiz und mögliche Motivation, sich nach der Lektüre dieses Buches eventuell auch mit der macOS-Entwicklung näher auseinanderzusetzen; Sie werden sehen, über das nötige Rüstzeug verfügen Sie dann. ©

1.1.2 Besonderheiten der iOS-Plattform

Auch wenn iOS auf macOS basiert, so gibt es doch mannigfaltige Unterschiede zwischen den beiden Betriebssystemen (auch wenn sie sich unter der Haube relativ ähnlich sind).

Entscheidend anders sind die Bedienoberflächen und das Bedienkonzept gestaltet. Während macOS und jedes andere Desktop-Betriebssystem typischerweise mittels Maus und Tastatur gesteuert werden, verfügen iOS-Geräte lediglich über einen Touchscreen, über den mittels Fingergesten und Berührungen alle Aktionen gesteuert werden. Hier gibt es also ganz neue Aspekte, auf die wir als Entwickler achten müssen, um gut funktionierende und intuitiv bedienbare Apps zu entwickeln. Denn ein Finger zum Bedienen eines Touchscreens ist nun mal etwas gänzlich anderes als eine Maus, die ich pixelgenau an jede Position bewegen kann. Ein Finger besitzt wesentlich mehr Fläche und allein das muss bereits beim Konzipieren und Entwickeln eigener Anwendungen für iOS maßgeblich beachtet werden.

Auch sind die Nutzer mit iPhone und iPad mobil unterwegs, was in heutigen Zeiten mit sehr gutem Ausbau des Mobilfunknetzes nichtsdestotrotz bedeutet: Nicht immer ist Internet verfügbar (mal ganz davon abgesehen, dass es das iPad auch in einer reinen WLAN-Version ohne Mobilfunkverbindung gibt) und den Nutzer dazu zu zwingen, eine Internetverbindung herzustellen, sollte nur wirklich dann erforderlich sein, wenn es gar nicht anders geht und ein Internetzugang zwingend für die Nutzung der eigenen App (oder der gerade benötigten Funktion) notwendig ist.

iPhone und iPad sind Mobilgeräte, und genau so werden sie auch genutzt, soll heißen: Viele Nutzer holen ihr Smartphone nur für den Bruchteil eines Augenblicks hervor, checken aktuelle Facebook- oder WhatsApp-Nachrichten und lassen das Handy anschließend wieder verschwinden. Auch für Sie als App-Entwickler gilt: Halten Sie den Nutzer bei dem, was er mit Ihrer App tun will, nicht auf. Weniger ist hier ganz klar mehr. Ihre App soll eine eindeutige Funktion erfüllen, bieten Sie diese darum dem Nutzer so komfortabel, übersichtlich und leicht zugänglich wie nur irgend möglich an.

Daneben gibt es noch einen weiteren wichtigen Aspekt, den wir als Entwickler immer berücksichtigen sollten: Schonender Umgang mit den Akku-Ressourcen. Wenn wir ununterbrochen Bluetooth in Beschlag nehmen und nach anderen Geräten suchen, saugen wir den Akku des Nutzers damit sehr schnell leer und dürfen uns wahrscheinlich im Umkehrschluss über schlechte Kritiken unserer App im App Store „freuen“. Hier gilt ganz klar: Weniger ist mehr, und Ihre App sollte sich immer auf genau die Aufgabe konzentrieren, für die sie gedacht ist.

Sie sehen also, Apps für iOS zu entwickeln besteht nicht nur darin, die Programmiersprache(n) und SDKs zu beherrschen; es geht auch darum, zu verstehen, wie die iOS-Gerätefamilie funktioniert, wie sie genutzt wird und wie Sie mit Ihren Apps den Nutzern das Leben erleichtern.