

Ľuboslav Lacko

Vývoj aplikací pro iOS

Od základů
po pokročilé techniky

Návrh layoutu
i aplikační logiky

Grafika, animace
a multimédia

Web, mapy
a navigace



computer
press

Vývoj aplikací pro iOS

Vyšlo také v tištěné verzi

Objednat můžete na
www.albatrosmedia.cz



Luboslav Lacko
Vývoj aplikací pro iOS – e-kniha
Copyright © Albatros Media a. s., 2018

Všechna práva vyhrazena.
Žádná část této publikace nesmí být rozšiřována
bez písemného souhlasu majitelů práv.

ALBATROS  **MEDIA**

Ľuboslav Lacko

Vývoj aplikací pro iOS

**Computer Press
Brno
2018**

Vývoj aplikací pro iOS

Luboslav Lacko

Překlad: Martin Herodek

Obálka: Martin Sodomka

Odpoředný redaktor: Martin Herodek

Technický redaktor: Jiří Matoušek

Translation © Martin Herodek, 2018

Objednávky knih:

www.albatrosmedia.cz

eshop@albatrosmedia.cz

bezplatná linka 800 555 513

ISBN tištěné knihy 978-80-251-4942-3

ISBN e-knihy 978-80-251-4947-8

Cena uvedená výrobcem představuje nezávaznou doporučenou spotřebitelskou cenu.

Vydalo nakladatelství Computer Press v Brně roku 2018 ve společnosti Albatros Media a. s. se sídlem Na Pankráci 30, Praha 4. Číslo publikace 34 481.

© Albatros Media a. s., 2018. Všechna práva vyhrazena. Žádná část této publikace nesmí být kopírována a rozmnožována za účelem rozšiřování v jakékoli formě či jakýmkoli způsobem bez písemného souhlasu vydavatele.

1. vydání

 **ALBATROS** MEDIA

Obsah

KAPITOLA 1

Nástroje na vývoj aplikací pro iOS	11
Trocha motivace na úvod	11
Co budete potřebovat	11
Co když nemáte k dispozici počítač s macOS?	12
Vývojové prostředí Xcode	14
Průběžná aktualizace	16
První spuštění	16
Playground	17
Projekt první aplikace pro iOS	18
Spuštění aplikace	25
Spuštění aplikace na simulátoru	26
Spuštění aplikace na reálném zařízení	28
Vývoj aplikací pro iOS ve Visual Studiu 2017	33
Xamarin	37

KAPITOLA 2

Programovací jazyk Swift	39
Programovací jazyk Swift	39
Zpětná kompatibilita s Objective-C	40
První pokusy s jazykem Swift ve vývojovém prostředí Xcode	40
Praktický úvod do programovacího jazyka Swift	43
Názvy objektů	45
Základní datové typy a struktury	46
Konstanty a proměnné	46
Zajímavé funkce na práci s řetězci v jazyce Swift 4	51
Vypočítané hodnoty	51
Pole a slovník	52
Set – množina údajů	53
Tuples	53
Hodnota nil a typ Optional	54
Řízení toku programu – cykly a podmínky	59
Příkaz Switch	61
Funkce	62

Struktury	64
Výčtový typ	65
Assertion debugg	66
Stručný úvod do objektově orientovaného programování (OOP)	66
Třídy	68
Dědičnost	69
Protokoly	69

KAPITOLA 3

Projekt aplikace	71
Životní cyklus aplikace	71
MVC = Model View Controller	72
Jak to funguje?	74
Cvičný příklad	75
Vytvoření projektu aplikace pro iOS	75
Projekt ve vývojovém prostředí Xcode	78
Komponenty projektu: soubory s aplikačním kódem	81
Komponenty projektu: definice uživatelského rozhraní	84
Návrh uživatelského rozhraní	84
Soubor s návrhem uživatelského rozhraní	88
Aplikační logika v souboru ViewController.swift	89
Propojení mezi vrstvami View a Controller	89
Aplikační kód: Intuitivně napsaná verze	92
Aplikační kód: podle zásad MVC	94
Ladění aplikace	95
Navigace mezi zobrazeními	98

KAPITOLA 4

Design uživatelského rozhraní	103
Zvláštnosti mobilní aplikace	104
Uživatelská zkušenost od okamžiku spuštění	104
Variabilita zařízení	105
Ergonomie používání aplikace	105
Principy designu	107
Stručný úvod do estetiky	109
Flat design	110
Typografie	112
Pátkový vs. bezpátkový font	112
Velikost písma	113
Délka řádku	114
Řádkování	114

Navigace v aplikaci	114
Velikost prvků vs. rozlišení	115
Ikony	116
Ikony pro iPhone	117
Ikony pro iPad	117
Ikony pro Spotlight, Nastavení a notifikace	118
Specifika designu aplikace pro iOS 9	120
Specifika designu aplikace pro iOS 10	121
Specifika designu aplikace pro iOS 11	121
Displej 3D Touch	122
Specifika designu aplikace pro iPhone X	122
Doporučení pro vývojáře	123
Uživatelské rozhraní	126
Hierarchie zobrazení	127
Auto Layout	127
Auto Layout s využitím prvků typu Stack	129
Auto Layout – základní principy	131
Priorita pravidel	132
Definování pravidel v návrhovém zobrazení Interface Builder	133

KAPITOLA 5

Interakce s uživatelem	141
Prvky na interakci s uživatelem	141
Spínač (UISwitch)	144
Segmentový přepínač (UISegmentedControl)	145
Posuvný ovladač (UISlider)	146
Rolování obsahu – Picker	147
Přepínač karet Tab Bar	158
Progress View	161
Delegát	163
Skrývání, zakázání, rozjasňování a stmívání prvků	164
UIAlertController	167
Práce s datem a časem	169
View	172
Příklad: Zobrazení vnořených pohledů	173
Scroll View	175
Split View	176
Dotyky a gesta	180
Příklad – rozpoznávání gest posunu (swipe)	182
Příklad – rozpoznávání klepnutí a gesta na pootočení	184
Komplexní příklady	188

Příklad: jednoduchá kalkulačka	188
Rozbor zadání	189
Návrh uživatelského rozhraní	190
Kód v souboru ViewController.swift	194
Kalkulačka využívající MVC	198
Návrh modelu – struktura	199
Publikované rozhraní	202
Návrh modelu – filozofie	203
Dokončení uživatelského rozhraní	208
Přizpůsobení uživatelského rozhraní	210
Příklad: Xylofon	217
Aplikační logika – přehrávání zvukových souborů	219
Příklad 3: Pexeso	221
MVC: Model	221
MVC: View	223
MVC: Controller	224

KAPITOLA 6

Grafika a animace	229
Quartz2D – vykreslování grafických útvarů	229
Zobrazování obrázků	233
Přizpůsobení obrázku vymezenému prostoru	233
Průhlednost obrázků	234
Příklad: Házení kostek	237
Ikony pro aplikaci	237
Obrázky pro aplikaci	238
Uživatelské rozhraní	240
Aplikační kód	241
Vylepšení aplikace	242
Příklad – kreslení dotykem	244
Animace	247
SpriteKit	249
Vytvoření projektu	249
Vytvoření hry	253
Herní logika	255
Kolize objektů	258
SceneKit	263

KAPITOLA 7

Multimédia	269
Přehrávání zvuku	269
Přehrávání videa	271
Přehrávání videa z YouTube	274
Focení a záznam videa	275
Příklad – použití fotoaparátu	275
Příklad – využití třídy UIImagePickerController	280

KAPITOLA 8

Práce s údaji	285
Lokální ukládání údajů	285
Jak přistupovat ke složkám Document, Library a tmp	290
Ukládání údajů do souborů	294
NSUserDefaults	294
Property lists	295
Vytvoření nového objektu Property List	296
Manuální zadávání údajů	298
Zadávání údajů v aplikačním kódu	299
NSCoding a NSKeyedArchiver	300
Framework Core Data	305
Příklad: seznam úkolů	308
Návrh uživatelského rozhraní	309
Zobrazení statických údajů	313
View pro zadávání údajů	315
Datový model	319
Ukládání údajů	321
Zobrazení údajů	323
Změna a mazání záznamů	326
Aplikace typu Master-Detail	328
Aplikace využívající UISplitViewController	329
Projekt s využitím šablony Master-Detail App	336
Databáze SQLite	346
Seznámení s databází SQLite	346
Interakce aplikace s databází	348
Příklad	349
Zpracování a zobrazení údajů ve formátu JSON	354
Příklad – načítání údajů z webu	355
Příklad – zobrazení údajů z JSON v tabulce	360

KAPITOLA 9

Senzory a komunikace 363

Core Motion	364
Údaje o pohybu	367
Akcelerometr, gyroskop a barometr	368
Příklad – zobrazení údajů z akcelerometru	368
Snímání QR kódu	373
Příklad – aplikace na snímání QR kódu	373
Komunikace aplikace přes Bluetooth	378
Podporované profily Bluetooth	379
Příklad komunikace s externím zařízením přes Bluetooth	379
Software pro Arduino 101	379
Aplikace pro iOS na komunikaci přes Bluetooth	383
Projekt aplikace	384

KAPITOLA 10

Web, mapy a navigace 389

Zobrazení webového obsahu	389
Příklad – použití Web View	389
Příklad – použití WebKit View	391
Příklad – webový prohlížeč	392
Příklad – zobrazení PDF dokumentu	395
Mapy a lokalizace	396
Zobrazení mapového podkladu	396
Příklad – zobrazení mapy	398
Příklad – zobrazení anotace zájmového místa	400

KAPITOLA 11

Pokročilá témata 403

Architektura operačního systému iOS	403
Cocoa Touch	404
Media	404
Core Services	404
Core OS	404
Pro migranty z Androidu	405
Frameworky	406
Foundation	406
UIKit	407
Nastavení ochrany soukromí uživatele	407
Nastavení parametrů aplikace	408
Příklad nastavování parametrů pro aplikaci	409
Modifikace implicitních parametrů	411

Symboly pro prvek Slider	414
Zadávání předdefinovaných hodnot	416
Načítání nastavených hodnot ve vaší aplikaci	418
Zjištění verze a sestavy aplikace	420
Implementace strojového učení	423
Lokální vs. cloudová inteligence	423
Koprocesor pro umělou inteligenci	423
Strojové učení	424
Core ML	425
Jak to funguje?	427
Příklad	427
Rozšířená realita	441
Cvičný příklad	442
Vytvoření 3D objektu pro rozšířenou realitu	448
Textury povrchu	451

KAPITOLA 12

Vývojářský účet a publikování aplikací v Apple App Store

Před publikováním	453
Pravidla, která musí aplikace splňovat, aby mohla být schválena	454
Bezpečnost	454
Funkcionalita	455
Byznys	457
Nákup v aplikaci	458
Design	458
Právní náležitosti	459
Duševní vlastnictví	460
Publikování aplikace	460

Rejstřík

Nástroje na vývoj aplikací pro iOS

Trocha motivace na úvod

Významným motivačním faktorem pro zvládnutí vývoje aplikací pro iOS může být skutečnost, že se vlastně jedná o tři platformy. Operační systém iOS se využívá především na všech mobilních platformách Applu, tedy smartphonech iPhone, tabletech iPad a také v multi-mediálním přehrávači iPod Touch. S rozmachem chytrých telefonů však tyto přehrávače ustupují stále více do pozadí.

Inteligentní hodinky Apple Watch využívají operační systém watchOS a Apple TV využívá tvOS. Tržní podíl iOS je tradičně nižší než u konkurenční platformy Android, kde je ale nejvyšší podíl levných zařízení. Naproti tomu zařízení s iOS patří, co se týče kvality a samozřejmě i ceny, mezi špičku, kterou používají hlavně úspěšní lidé s adekvátními příjmy, pro které není problém koupit si i aplikaci. Samozřejmě v případě, že je zaujme.

Co budete potřebovat

Pokud netrpělivě čekáte informaci, jaké vývojové prostředí si máte stáhnout na svůj počítač a jak ho nakonfigurovat, je to trochu komplikovanější. Takto položená otázka je totiž irelevantní. Správně položená otázka by zněla: Jaký počítač budete potřebovat na vývoj aplikací pro iPhone/iPad/iPod Touch? Na tuto otázku je jedna jednoznačná a jedna alternativní odpověď.

Začneme jednoznačnou. Ve finále, tedy na publikování aplikace do aplikačního obchodu Apple App Store, budete potřebovat počítač s operačním systémem macOS, čili ze stolních modelů iMac, iMac Pro, Mac Pro, případně Mac mini, z přenosných modelů MacBook, MacBook Air, případně MacBook Pro.

V této kapitole:

- Budete umět nainstalovat a nakonfigurovat vývojové prostředí Apple Xcode.
- Dokážete vytvořit projekt jednoduché aplikace.
- Budete umět projekt sestavit a spustit na simulátorech různých zařízení s iOS a také na reálném zařízení připojeném k vývojářskému počítači.
- Získáte přehled o možnostech a omezeních vývoje aplikací pro iOS ve Windows s použitím Visual Studio 2017 a technologie Xamarin.

Počítač s operačním systémem macOS budete prakticky celou dobu vývoje potřebovat i v případě, že budete vyvíjet ve Visual Studiu 2017 na platformě Windows. Musíte propojit Xamarin na počítači s Windows s Xamarin agentem na počítači s macOS. Nutnost mít k dispozici macOS je možné chápat ze dvou pohledů. Pesimistický pohled se soustředí na nutnost sehnat si pro vývoj počítač s macOS, ať už na celý vývoj, případně si ho alespoň půjčit na publikování aplikace. Naproti tomu optimistickým pohledem odhalíte, že na podobném principu a ve stejném vývojovém prostředí můžete vyvíjet aplikace nejen pro iPad/iPod/iPhone, ale i pro macOS.

Abyste mohli své aplikace spouštět na více reálných zařízeních a po otestování je šířit přes App Store, je potřeba se zaregistrovat do placeného programu na vývojářském portálu *developer.apple.com*, přičemž si můžete vybrat standardní program za 99 USD, případně Enterprise program, vhodný pro větší firmy, za 299 USD ročně. Studenti se mohou bezplatně zapojit do programu iOS Developer University Program.



Tip: Pokud vám stačí spouštět aplikace pouze na jednom zařízení s iOS bez možnosti publikovat je do aplikačního obchodu, stačí vám bezplatný vývojářský účet.

K samostatnému vývoji v případě, že se rozhodnete pro nativní vývojové prostředí Xcode od Applu, budete také potřebovat počítač s operačním systémem macOS, kterýkoliv z uvedených typů, dokonce to nemusí být ani nejnovější model. Nároky vývojového prostředí na výpočetní výkon i paměťovou kapacitu jsou poměrně rozumné, takže vám postačí i ten nejlevnější přístroj, i když pojem nejlevnější v souvislosti s počítači Apple je hodně specifický pojem.

Migrace vývojářů na jinou platformu je zpravidla jednoduchá a bezproblémová, v tomto případě se ale pro uživatele operačního systému Windows jedná o migraci dvojnásobnou. Nejprve se musí seznámit alespoň se základními principy fungování operačního systému macOS, a potom si osvojit nové vývojové prostředí Xcode. Odměnou za migraci je možnost vytvářet aplikace pro tři mobilní platformy iPad/iPod/iPhone a jednu desktopovou platformu macOS, přičemž všechny platformy jsou momentálně na vrcholu popularity.

Kromě hardwaru potřebného k vývoji předpokládá publikace základní znalosti:

- Znalost základů programování, výhodou je znalost některého z moderních programovacích jazyků, jako jsou Java, C++, C#. Jazyk Swift, používaný na vývoj aplikací pro iOS, je těmto jazykům velmi blízký.
- Výhodou je znalost základních principů objektově orientovaného programování.
- Znalost základů SQL (nejlépe SQLite, ale postačuje základní všeobecný přehled).
- Znalost základů formátů XML a JSON.

Co když nemáte k dispozici počítač s macOS?

Na počítač s macOS, pokud má dostatek paměti a výkonu, můžete bez problémů nainstalovat Windows, buď do jiného diskového oddílu nebo pomocí některého z virtualizačních nástrojů, například Parallels Desktop, což je komerční aplikace, nebo Virtualbox od Oraclu, který je

zdarma. Opačně to ale nejde, na počítač, který nevyrobila společnost Apple, macOS nenainstalujete. Alespoň oficiálně ne. Apple striktně kontroluje dodržování práv ke svému softwaru, proto se řešením typu iATKOS nebudeme zabývat.

Mimochodem část o možnostech spuštění Windows na počítačích Mac nebyla samoučelná. Pokud byste potřebovali spustit některý z nástrojů, který lépe funguje ve Windows, jako například donedávna Visual Studio, může se vám tato informace hodit. Pokud nemáte k dispozici počítač s macOS, tak si ho musíte obstarat (nemusí to být nejdražší iMac či MacBook Pro, vystačí i s nejlevnějším kompaktním Mac Mini).

Jestliže nechcete investovat do hardwaru, můžete využít cloudovou službu *macincloud.com*, ve které můžete vyvíjet aplikace v Xcode a spouštět i jiné aplikace pro macOS.



Obrázek 1.1: Portál služby macincloud.com

K dispozici je několik cenových plánů, například Pay-As-You-Go, ve kterém zaplatíte za hodinu 1 USD, nebo plán Managed Server, kde měsíční poplatek začíná na 20 USD.

The screenshot displays the macincloud.com website with four pricing plans:

- Pay-As-You-Go:** \$1 per hour. Features include Static IP, 100Mbps Network, No Admin/Root Access, Access through RDP, Physical Non-VM Mac Server, Save files on server or sync using: Dropbox, OneDrive, Google Drive, Box, etc., Access to Tools and Applications such as Xcode, iBooks Author, Application Launcher, and much more, and Bring your Own Software Licenses.
- Managed Server:** \$20+ per month (marked as MOST POPULAR). Features include Static IP, 100Mbps Network, No Admin/Root Access, Access through RDP, Physical Non-VM Mac Server, Save files on server or sync using: Dropbox, OneDrive, Google Drive, Box, etc., Access to Tools and Applications such as Xcode, iBooks Author, Application Launcher, and much more, and Bring your Own Software Licenses.
- Dedicated Server:** \$49+ per month. Features include Static IP, 100Mbps Network, Full Admin/Root Access, VNC or 3rd Party Apps, Instance on Physical Mac Server, Save files on server or sync using: Dropbox, OneDrive, Google Drive, Box, etc., Access to Tools and Applications such as Xcode, Jenkins CI, Web Server, and much more, and Bring your Own Software Licenses.
- VSTS Build Agent:** \$29 per month. Features include Static IP, 100Mbps Network, Build Remotely, Non-Interactive VSTS, TFS, Xamarin, Physical Non-VM Mac Server, Unlimited Build Size and Build Time, Access to Tools and Applications such as Xcode, Xamarin, Cordova, and much more, and Bring your Own Software Licenses.

Obrázek 1.2: Cenové plány služby macincloud.com

Vývojové prostředí Xcode

Vývojové prostředí Xcode si stáhnete z aplikačního obchodu pro platformu macOS. Je k dispozici zdarma. Nejnovější verze v době psaní publikace byla 9.2 a tato verze je určená pro počítače s operačním systémem macOS 10.12.6 nebo novějším.



Tip: Doporučujeme aktualizovat macOS na nejnovější verzi, abyste mohli využívat nejnovější verzi Xcode.



Obrázek 1.3: Vývojové prostředí Xcode v aplikačním obchodě

To, že vývojové prostředí Xcode je v aplikačním obchodě pro macOS k dispozici zdarma (v minulosti bylo zdarma jen pro registrované vývojáře, ostatní si ho mohli koupit za symbolickou sumu 5 dolarů), je chytrý tah od společnosti Apple, kterým chce získat zájem začínajících a hobby vývojářů. Pokud máte počítač s macOS, nemusíte už do začátku vývoje aplikace nic investovat. A až po vytvoření první životaschopné aplikace, o které si budete myslet, že by o ni mohl být zájem, a jejím odladění v simulátoru si můžete zaplatit roční vývojářskou licenci a začít aplikaci šířit přes App Store.

V předchozí části vás určitě upoutala informace, že pokud chcete přenést svou aplikaci z vývojového prostředí do reálného zařízení a následně po finálním odladění ji šířit přes App Store, musíte mít vytvořený vývojářský účet, za který se platí ročně 99 USD. V opačném případě můžete své aplikace testovat jen na simulátoru, který je součástí vývojového prostředí Xcode, a maximálně na jednom zařízení s iOS. Takže 99 dolarů ročně a co za to? Prodávat dobré aplikace (skutečně musí být dobré) je skvělý byznys s minimálními počátečními náklady. Celý distribuční systém za vás zařídí Apple.

A ještě jedna věc (jak by řekl zakladatel společnosti Apple Steve Jobs): Kde jinde si za 99 dolarů ROČNĚ můžete najmout tým testerů, kteří vám vaši aplikaci důkladně otestují, čímž v mnoha případech výrazně přispějí ke spokojenosti jejich budoucích uživatelů a vašemu profitu?

Společně s aplikací Xcode se stáhnou a nainstalují všechny potřebné frameworky a nástroje včetně simulátoru zařízení s iOS, ve kterém můžete spouštět a ladit aplikace přímo na vývojářském počítači bez toho, abyste je museli pokaždé přenášet a spouštět na skutečném zařízení.

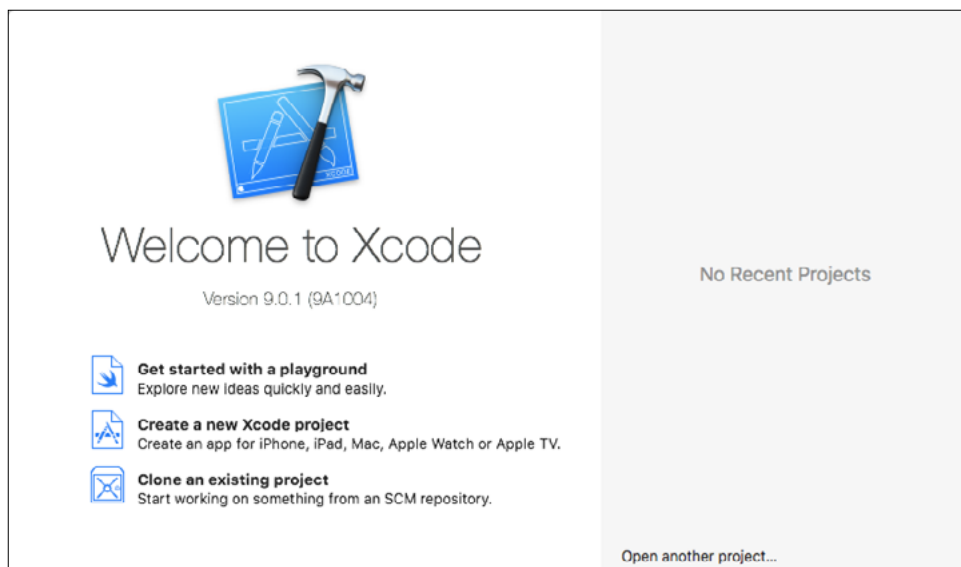
Průběžná aktualizace

Pokud je k dispozici nová verze iOS, je potřeba aktualizovat i vývojové prostředí Xcode. Operační systém macOS vývojářského počítače vám dostupnost aktualizace Xcode oznámí obvyklým způsobem, podobně jako aktualizaci ostatních aplikací. Například červeným odznakem na ikoně aplikačního obchodu. Doporučujeme udržovat vývojové prostředí aktuální. Jedině tak máte jistotu, že vaše aplikace budou naplno využívat všechny možnosti operačního systému.

Pojem „průběžná aktualizace“ můžete chápat ve dvou rovinách. Udržujte aktuální nejen vývojové prostředí, ale i svou aplikaci. Každá nová verze iOS přinese nové a vylepšené funkce, ze kterých by vaše aplikace s vysokou pravděpodobností mohla profitovat. Proto doporučujeme tyto funkce nastudovat, a pokud to má význam, implementovat je formou aktualizace do vaší aplikace.

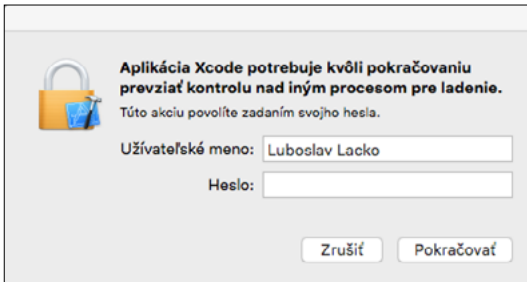
První spuštění

Při prvním spuštění vývojového prostředí Xcode se zobrazí dialog, který vás vyzve k instalaci potřebných komponent. Instalace v závislosti na rychlosti internetového připojení trvá několik minut. Pokud máte otevřenou aplikaci iTunes, musíte ji před zahájením instalace komponent ukončit.



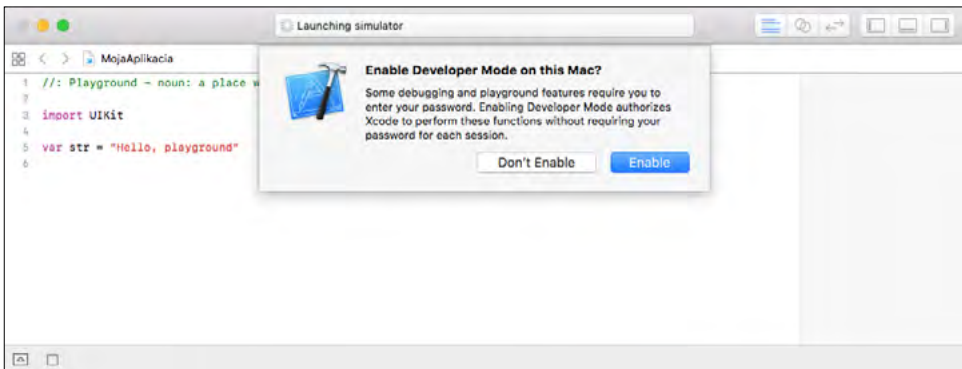
Obrázek 1.4: Úvodní dialog po spuštění vývojového prostředí Xcode

Zobrazí se dialog, ve kterém vám operační systém oznamuje, že aplikace Xcode potřebuje kvůli pokračování převzít kontrolu nad jiným procesem pro ladění a žádá vás, abyste tuto akci potvrdili pomocí svého přístupového hesla.



Obrázek 1.5: Povolení, aby vývojové prostředí mohlo převzít kontrolu nad jiným procesem pro ladění

Následně vás vývojové prostředí Xcode v dalším dialogu vyzve, abyste povolili vývojářský mód pro svůj počítač s macOS. I tuto akci musíte potvrdit pomocí svého přístupového hesla.

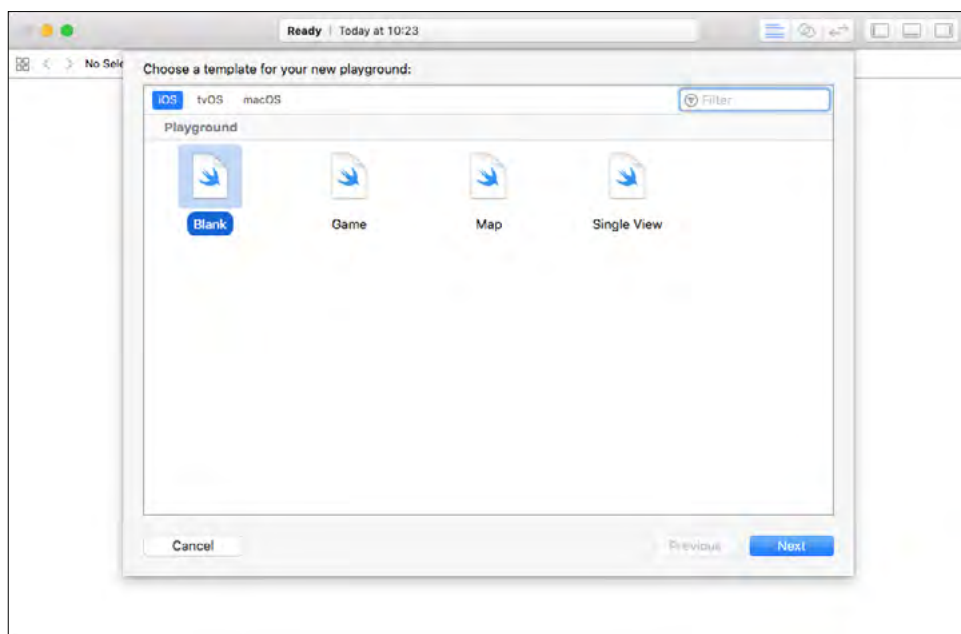


Obrázek 1.6: Povolení vývojářského módu na počítači Mac

Playground

Můžete vytvořit nejen projekt reálné aplikace, ale i takzvaný projekt Playground, ve kterém můžete interaktivně experimentovat s novým programovacím jazykem Swift bez toho, abyste při každém pokusu museli kompilovat, sestavovat a spouštět projekt aplikace pro iOS, a to jak, na simulátoru, tak na reálném zařízení.

Playground funguje jako sofistikovaná grafická konzolová aplikace, ve které po napsání řádku nebo bloku kódu hned vidíte výsledek, ať už v textové nebo grafické podobě. Pokud chcete vytvořit projekt Playground, klepněte na volbu **Get started with a playground** a zvolte vhodný typ projektu.



Obrázek 1.7: Nabídka typů projektu Playground



Poznámka: Playground je klíčový nástroj, který využijete při seznamování se s novým programovacím jazykem Swift. Tomuto tématu je věnovaná druhá kapitola.

Projekt první aplikace pro iOS

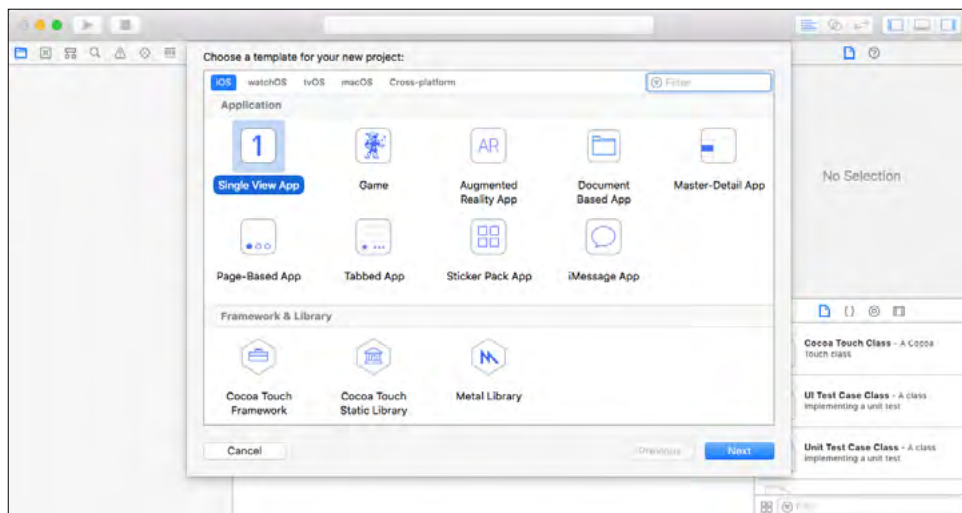
Možná se ptáte, proč se pouštíte do vytváření projektu mobilní aplikace dříve, než vás aspoň v hrubých rysech seznámíme se základními principy designu a vývoje aplikace a v neposlední řadě i nového programovacího jazyka Swift. Důvod je jednoduchý. Pokud se aplikace dá přeložit a spustit nejprve na simulátoru a následně na reálném zařízení, máte jistotu, že máte správně nainstalované a nakonfigurované vývojové prostředí, simulátor a propojení na reálné zařízení. Proto je i koncepce návodu na tento projekt zaměřena na co nejrychlejší dosažení cíle, čili na vytvoření aplikace a její spuštění na simulátoru a následně na reálném zařízení.



Poznámka: První projekt má v tomto případě i motivační význam, jelikož doslova na několik málo klepnutí a bez jakéhokoliv programování vytvoříte aplikaci typu „Hello World“, která vypíše na obrazovku text.

Začátečníci se v tomto případě nemusí snažit pochopit souvislosti. Návod na vytvoření cvičného projektu je uveden jako obrázkový postup krok za krokem.

Vytvoření projektu nové aplikace spustíte v úvodním dialogu **Welcome to Xcode** volbou **Create a new Xcode project**. Zobrazí se nabídka platform a typů projektů. V horní části nabídkového dialogu jsou karty pro jednotlivé platformy. Ujistěte se, že je zobrazena karta iOS.

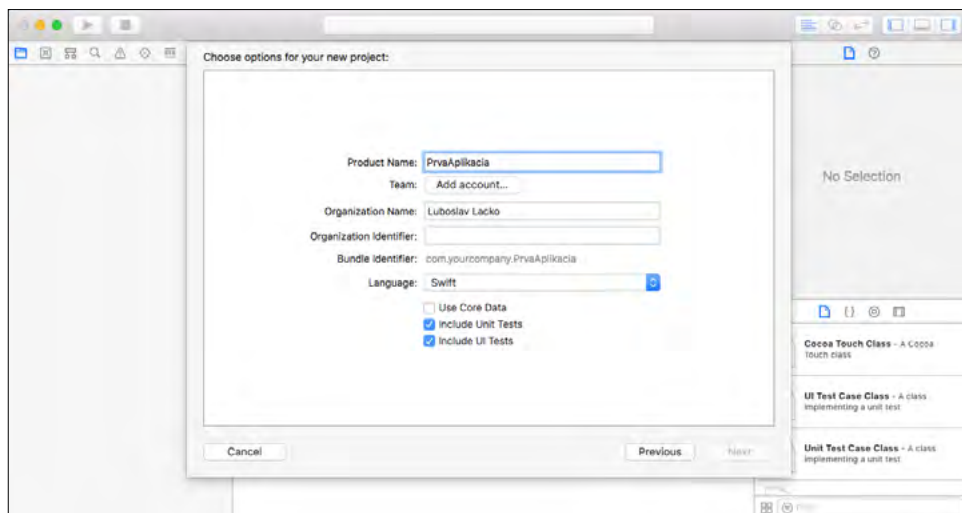


Obrázek 1.8: Nabídka typů projektů pro iOS

V této úvodní seznamovací aplikaci využijete nejjednodušší šablonu **Single View App**. V dialogu průvodce vytvoření projektu je potřeba zadat název projektu, vybrat si programovací jazyk a v případě reálné aplikace vyplnit i několik dalších parametrů.

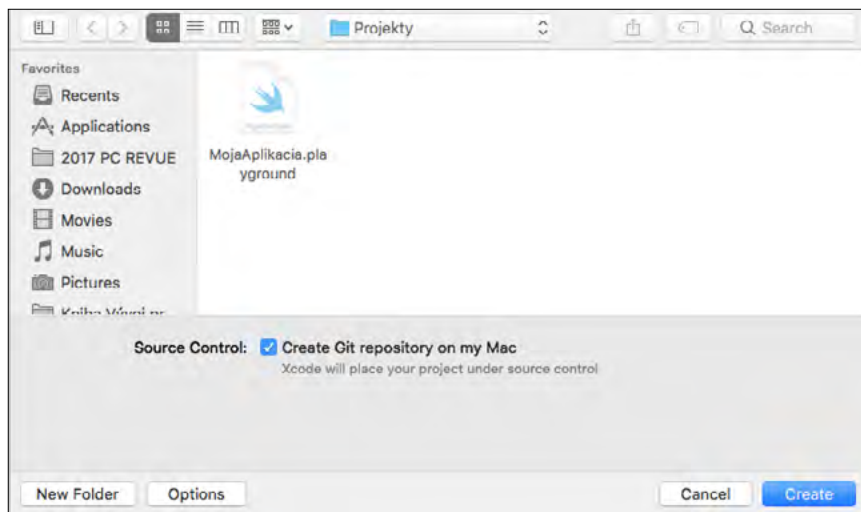
Jako programovací jazyk ponechejte implicitně předvolený **Swift**. Parametr **Organization Name** můžete nechat nevyplněný. Povinné je vyplnění položky **Organization Identifier**. Zadává se ve tvaru *com.doména*, například *com.firma*. Tento parametr slouží k jednoznačné identifikaci vaší aplikace. Na cvičné příklady můžete použít identifikátor *com.example*. Parametr **Team** v tomto příkladu nenastavujte.

Ostatní parametry nechejte tak, jak jsou nastaveny implicitně, případně můžete zrušit označení voleb **Include Unit Tests** a **Include UI Test**. Vývojové prostředí má implementované pokročilé nástroje na testování, v takto jednoduché aplikaci je ale nebudete na nic potřebovat. První aplikace nebude využívat ukládání údajů, takže i volbu **Use Core Data** ponechejte neoznačenou.



Obrázek 1.9: Zadávání parametrů projektu. Na obrázku je implicitní nastavení předvoleb. V tomto projektu nepotřebujete žádnou z nich, takže můžete zrušit implicitně zaškrtnuté předvolby

V dalším kroku vyberte adresář, ve kterém bude projekt aplikace uložen na vašem vývojářském počítači. Doporučujeme vytvořit samostatný adresář pro vaše projekty. Políčko **Source Control: Create Git repository on my Mac** nechejte neoznačené. Pokud ho v reálném projektu zaškrtnete, vývojové prostředí Xcode vytvoří pro váš projekt lokální Git repozitář uvnitř vybraného adresáře.



Obrázek 1.10: Výběr umístění projektu

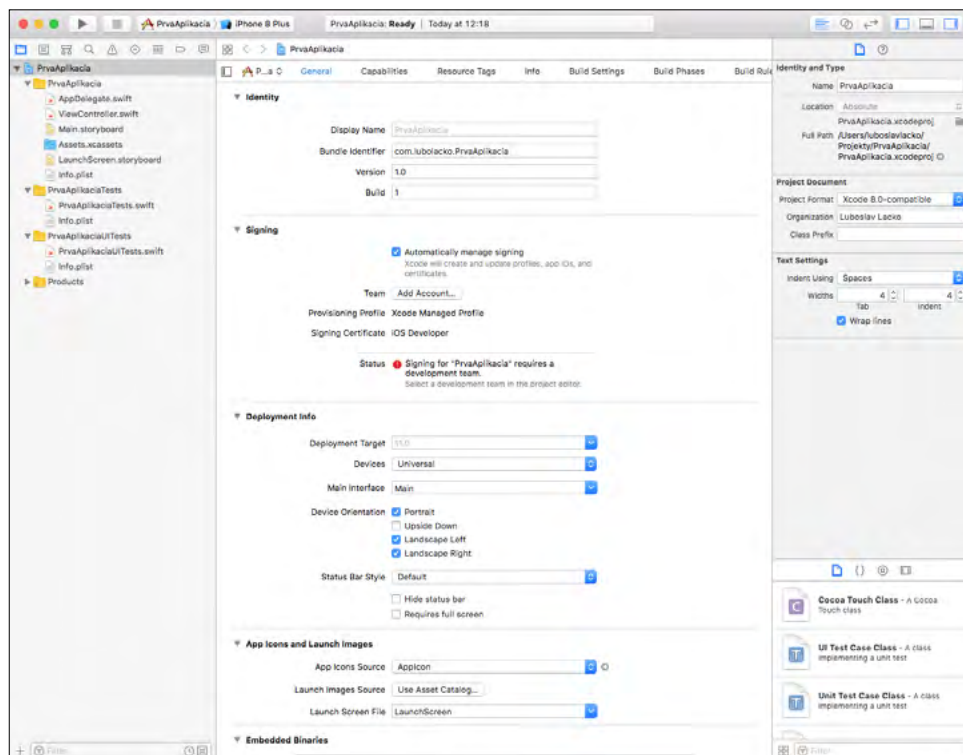
Po zadání všech požadovaných údajů vývojové prostředí Xcode automaticky vytvoří adresářovou strukturu projektu a přidá potřebný základní aplikační framework. Po vytvoření nového projektu je rozmístění oken takové, jako vidíte na obrázku. V levé části okna aplikace je v okně **Navigator** zobrazena hierarchická struktura projektu. Všimněte si ikon na horní liště. Pomocí nich můžete okno **Navigator** přepínat do různých režimů. Lze volit z více karet, v tomto projektu vystačíte s první kartou, **Project navigator**, která je úplně vlevo a má ikonu složky.

V hlavním okně uprostřed jsou po vytvoření projektu sumární informace o projektu včetně verze operačního systému variant plynoucích ze změny orientace displeje (na výšku/na šířku), ikon v různém rozlišení podle typu displeje a podobně. V tomto okně budete vytvářet návrh uživatelského rozhraní a také psát aplikační kód.

V pravém okně, nazývaném **Utilities**, jsou další důležité informace a položky, jimž budeme věnovat pozornost později. Nejdůležitější ovládací prvky najdete na horním panelu, v originální terminologii se nazývá **Toolbar**.



Tip: Úplně vpravo nahoře najdete ikony, pomocí kterých můžete zobrazit nebo skrýt postranní okna **Navigator** a **Utilities**.



Obrázek 1.11: Obrazovka vývojového prostředí po vytvoření projektu

Problematicke návrhu uživatelského rozhraní se budeme věnovat v dalších kapitolách. V tomto úvodním projektu přidáme na plochu aplikace jen jeden pasivní prvek typu **Label**, pomocí kterého aplikace vypíše textový řetězec, abyste se přesvědčili, že aplikace skutečně funguje. Zatím nebudeme řešit interakci s uživatelem.

Všimněte si v levém, svisle orientovaném okně **Navigator**, že projekt aplikace obsahuje (kromě jiného) v hlavní složce, která koresponduje s názvem projektu, čili v tomto případě *PrváAplikácia*, dvě položky s příponou *storyboard*. V souboru *Main.storyboard* je definované uživatelské rozhraní, které aplikace zobrazí na první obrazovce po spuštění. Je to poměrně složitý XML dokument, to vás ale nemusí trápit, protože uživatelské rozhraní aplikace budete vytvářet výhradně v návrhovém zobrazení nazývaném **Interface Builder**.

Je to vizuální prostředí, ve kterém můžete jednoduchým přesouváním a spojováním přidávat a modifikovat objekty tvořící uživatelské rozhraní, přičemž je možné jednotlivým objektům nastavovat a upravovat atributy, definovat přechody mezi pohledy a podobně. Ve složitějších projektech využívajících více pohledů budete moci definovat navigační strukturu mezi pohledy. **Interface Builder** pracuje nad objektem *storyboard*. Umožňuje navrhnout celé uživatelské rozhraní aplikace, přičemž je na jednom místě možné získat přehled o všech zobrazeních a jejich propojení. Je možné zde definovat přechody z jednoho ovladače pohledů na druhý. Tyto přechody jsou graficky znázorněny. Můžete navrhnout celé uživatelské rozhraní v jednom souboru *storyboard* nebo ho pro lepší přehlednost rozdělit na více částí do více souborů *storyboard*.

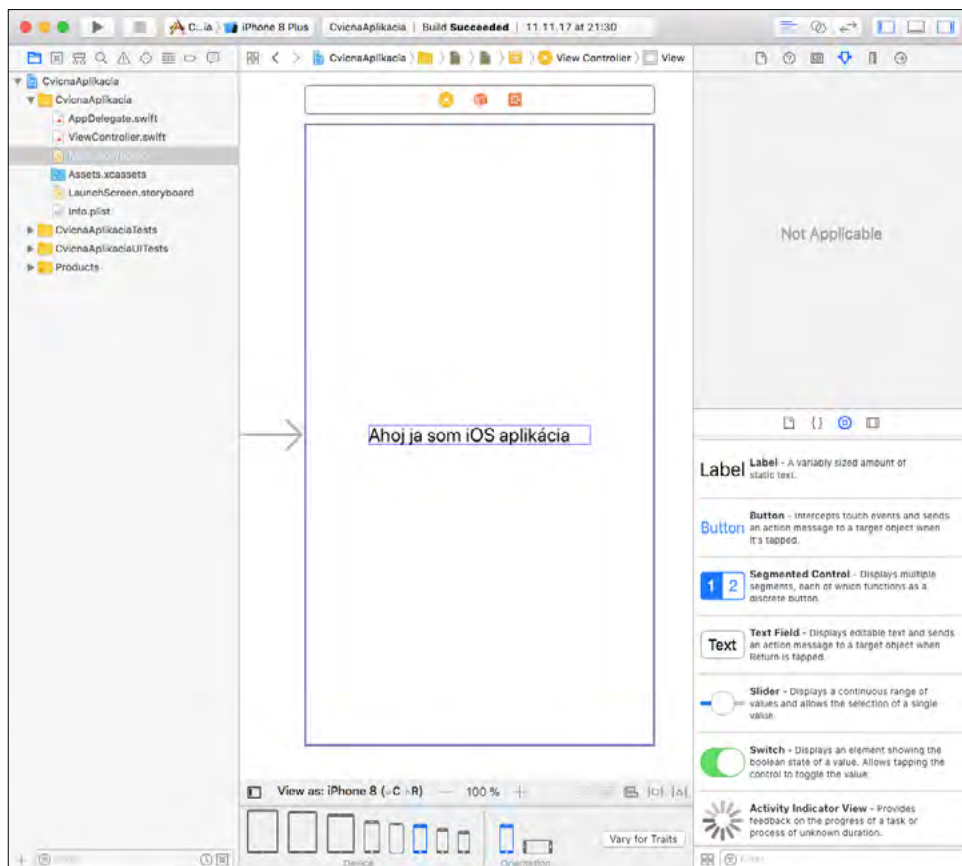
Otevřete položku *Main.storyboard*. Zobrazí se plocha nástroje **Storyboard**, na které budete navrhovat uživatelské rozhraní aplikace. Toto uživatelské rozhraní může sestávat z jedné nebo více obrazovek, které se nazývají pohledy, v originální terminologii *View*.



Poznámka: Nástroj **Interface Builder** umožňuje navrhnout vizuální podobu aplikace a navigační strukturu mezi více obrazovkami.

Jelikož jste vytvořili projekt aplikace typu **Single View App**, bude tato plocha po vytvoření jedinou obrazovkou uživatelského rozhraní aplikace. Cílem bude vytvořit aplikaci, která něco zobrazí. Samozřejmě nic vám nebrání postupně přidávat další pohledy a definovat navigační strukturu přechodů mezi nimi.

V tomto projektu nebudeme řešit žádnou interakci s uživatelem, to bude námětem dalších kapitol. Aplikace zobrazí jen textový řetězec, aby bylo zřejmé, že všechny inicializační záležitosti proběhly v pořádku. Pokud byste aplikaci spustili hned po vytvoření projektu, zobrazila by se na zařízení s iOS bílá prázdná obrazovka a nedalo by se na první pohled říct, zda aplikace běží, nebo se zasekla někde při inicializaci. Jako první a jediný úkon při návrhu uživatelského rozhraní potřebujete přidat na plochu aplikace prvek **Label**, čili textové pole sloužící jen k výpisu. Jinak řečeno, uživatel tento typ ovládacího prvku nemůže použít k zadávání textu.



Obrázek 1.12: Návrh uživatelského rozhraní

Všimněte si okna **Utilities** vpravo. Je vodorovně rozděleno na dvě části. Ve spodní části se přepnete na kartu **Object Library**. Zobrazení můžete přepnout do módu ikon, kde máte zobrazených najednou více prvků, nebo do módu seznamu, kde je u každého prvku jeho název a účel.

Implicitně je vybraná karta **Show the File Template Library**, takže pokud chcete umísťovat vizuální prvky uživatelského rozhraní aplikace, přepněte se na kartu **Show the Object Library**.



Tip: Jelikož **Object Library** obsahuje téměř 70 prvků, vhodný prvek najdete nejrychleji a nejjednodušeji tak, že se tlačítkem dole vlevo (vedle editačního okna **Filter**) přepnete ze zobrazení seznamu na zobrazení ikon nebo zadáním klíčového slova do pole **Filter** přímo najdete požadovaný prvek. U vizuálního návrhu přesouváte prvky z okna **Object Library** na plochu aplikace. Pro většinu prvků je třeba definovat parametry určující jeho vzhled, chování a dopsat kód aplikační logiky určující jeho funkcionalitu.



Obrázek 1.13: Okno na zobrazování položek přepnuté na kartu Object Library. Všimněte si dole vlevo přepínacího tlačítka módu zobrazení a pole na zadání výrazu pro filtr. Vpravo je okno Object Library v módu zobrazování ikon

Umístěte do středu plochy aplikace prvek **Label** na výpis textového oznámení. Klepněte na tento prvek a změňte jeho text v souladu s účelem, v tomto projektu například na „Ahoj ja som iOS aplikácia“. Zatím neřešte ani design prvků, ani to, jak bude uživatelské rozhraní vypadat na různých typech zařízení. Jedno textové pole se vejde i na ten nejmenší displej, například iPhonu SE. Jedinou úpravou je roztažení prvku **Label** do šířky, aby se do něj vešel i delší text.

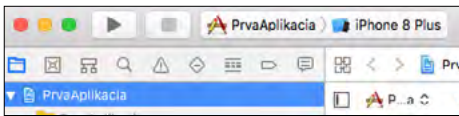
Během návrhu uživatelského rozhraní aplikace vás určitě bude zajímat, jak bude vaše aplikace vypadat na různých typech zařízení, které mají různou velikost úhlopříčky displeje a různé rozlišení, a to nejen při orientaci na výšku, ale i na šířku. V dolní části můžete náhled uživatelského rozhraní přepnout na různé typy iPhonů a iPadů, a to na výšku i na šířku. Po přepnutí třeba na iPhone SE bude výpis nesymetrický, posunutý horizontálně i vertikálně, to však v tomto příkladu nemusíte vůbec řešit.



Poznámka: Ve spodní části je kromě karty **Object Library** zajímavá i karta **Code Snippet Library**, na které najdete předpřipravené bloky kódu pro základní programové konstrukce, jako jsou cykly, podmínky a podobně.

Spuštění aplikace

V horní části vlevo si všimněte tlačítka s ikonou šipky vpravo, pomocí kterého aplikaci spustíte.

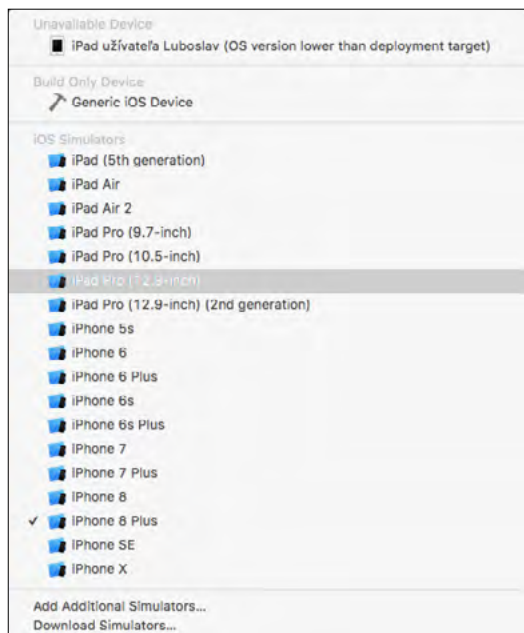


Obrázek 1.14: Tlačítko na spuštění aplikace a napravo od něj je prvek na výběr zařízení nebo simulátoru

Nejprve je ale potřeba vyřešit otázku, na čem vlastně bude aplikace spuštěna, zda na připojeném reálném zařízení, nebo na simulátoru, a pokud ano, tak na simulátoru jakého zařízení a jaké verze. Abychom vaši pozornost předem namířili na nutnost nastavení vhodné verze iOS, znázorňuje obrázek s nabídkou připojených zařízení a simulátorů situaci, kdy je aplikace sestavená pro nejnovější verzi iOS 11 a na připojeném, úmyslně neaktualizovaném iPadu je aktuálně verze iOS 10.3.3. Proto je vedle názvu připojeného zařízení informace o tom, že jeho operační systém je starší (nižší) verze, než pro jakou bude projekt sestaven.



Upozornění: Na zařízení s nižší verzí iOS, než je verze, pro kterou byla aplikace sestavena, danou aplikaci nespustíte.



Obrázek 1.15: Výběr zařízení nebo simulátoru

Spuštění aplikace na simulátoru

Zatím nebudeme řešit spuštění na připojeném zařízení a aplikaci spustíme pomocí implicitní volby, na simulátoru, v tomto případě na simulátoru zařízení iPhone X. Po sestavení a zavedení aplikace do simulátoru se aplikace spustí.

Přesně toto byl primární cíl této aplikace, která nedělá nic užitečného. Co nejrychleji se dostat do fáze, kdy budete moci aplikaci spustit, a aby po spuštění bylo něco na obrazovce, ne pouze bílá plocha.



Poznámka: Aplikaci sestavíte z Xcode pomocí tlačítka se symbolem čtverce, které se po spuštění aplikace zobrazí vpravo od tlačítka, jímž jste aplikaci spustili.

Simulátor v takovémto podobě je sice graficky efektní, znázornění rámu reálného zařízení ale zabírá při ladění cenné pixely, především ve svislém směru. Nad i pod displejem je hodně nevyužitého místa. Proto doporučíme po prvním spuštění simulátoru upravit zobrazení tak, aby se zobrazoval jen obsah displeje bez orámování. V nabídce simulátoru proto deaktivujte volbu **Windows → Show Device Bezels**. Po tomto úkonu simulátor zabírá na displeji vývojářského počítače jen skutečně nezbytný prostor, takže pravděpodobně můžete nastavit větší zvětšení a dosáhnout tak lepší čitelnosti textu.

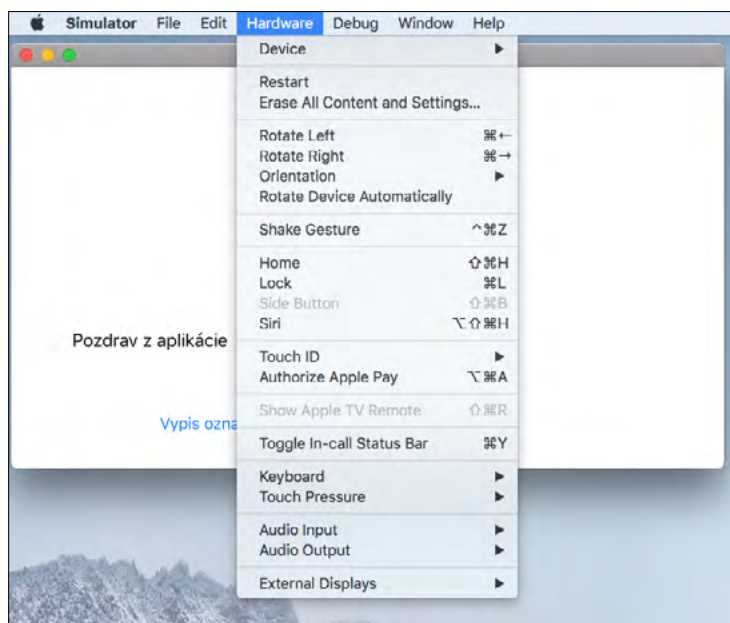


Obrázek 1.16: Spuštění aplikace na simulátoru

Doporučujeme prozkoumat hlavně položky nabídky **Hardware**, abyste zjistili, co všechno dokáže simulátor zařízení s iOS simulovat. Nejčastěji budete zřejmě využívat změny polohy zařízení a přepínat zobrazení „na výšku“ a „na šířku“.

Můžete například simulovat gesto, u kterého zatřesete zařízením. Simulátor iPhone 7 a vyšších verzí umožňuje například pomocí trackpad force simulovat silnější stisk. Samozřejmě pouze v případě, že touto funkcí vývojářský počítač, případně k němu připojený externí trackpad, podporuje. V nabídce **Debugg** najdete pokročilejší funkce, můžete například simulovat nedostatek paměti a podobně.

Simulátor umožňuje otestovat hodně funkcionality aplikace, nikoliv ovšem úplně vše. Některé funkce potřebují hardwarové komponenty, které vývojářský počítač nemá, například akcelerometr, gyroskop, kompas či jiné senzory, nebo se nedají dost dobře využít, protože využívají jiný typ hardwaru, jako je například fotoaparát. Pokud se například pokusíte použít simulátor na otestování interakce s aplikací *Fotoaparát*, program selže. Tyto funkce musíte testovat na skutečném zařízení.



Obrázek 1.17: Položky nabídky Hardware. Zařízení je v režimu „na šířku“ s vypnutým orámování



Upozornění: I v případě, že vaše aplikace nevyužívá nic nad rámec možností simulátoru, byste ji vždy měli důkladně otestovat i na skutečném hardwaru.

Na testování nemusíte mít nejnovější iPhone – v době psaní této publikace to byl iPhone X – spíše naopak. Lepší je otestovat aplikaci na nejmenším a nejméně výkonném zařízení, u něhož ještě připadá v úvahu, že na něm budou uživatelé vaši aplikaci spouštět. Díky kvalitním simulátorům dokážete realizovat celý vývoj a teoreticky i testování většiny aplikací, které nevyužívají specifickou hardwarovou výbavu, bez toho, abyste měli hardwarové zařízení fyzicky k dispozici, avšak v závěrečné fázi před publikováním do aplikačního obchodu doporučujeme aplikaci důkladně otestovat i na „železe“, nejlépe na telefonu i tabletu.

Spuštění aplikace na reálném zařízení

Zařízení s iOS normálně povolují instalaci aplikací pouze z aplikačního obchodu App Store. Jedinou výjimkou je spuštění aplikací během procesu vývoje a ladění. Tehdy se aplikace nainstaluje do zařízení s iOS z vývojářského počítače s vývojovým prostředím Xcode. Jelikož se jedná o určité obcházení běžného postupu instalace z App Store, nazývá se takovéto zavádění a spuštění aplikací Sideloadung.



Upozornění: Apple omezil počet aplikací, které můžete takto nahrát do reálného zařízení, na 10 za týden.

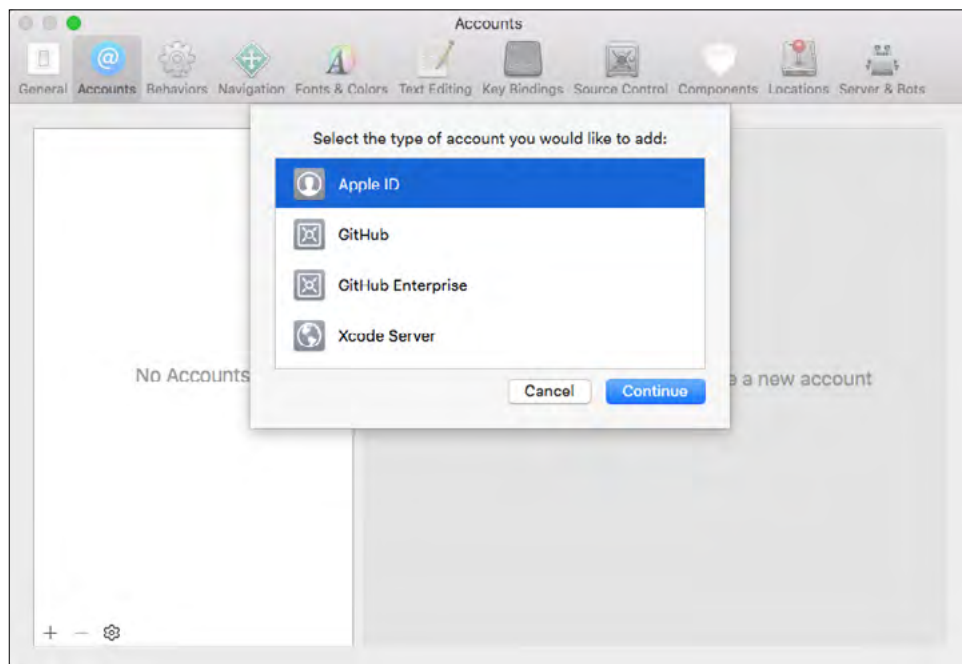
Pro vývojáře reálných aplikací nepředstavuje toto omezení žádný problém, tempem 10 aplikací za týden nevyvíjí nikdo, problém to ale může být při učení. Proto cvičné příklady, které to nepotřebují, spouštějte a testujte přednostně na simulátoru.

Před spuštěním aplikace na fyzickém zařízení musíte mít účet na stránce Apple Developer (developer.apple.com). Zpočátku vám stačí vývojářský účet, který je zdarma. Potřebujete k tomu svůj účet Apple ID. To, že chcete připojit reálné zařízení, logicky znamená, že takového zařízení máte, a určitě k němu máte vytvořený i účet Apple ID. Pokud takovýto účet nemáte, můžete si ho – také bezplatně – vytvořit.



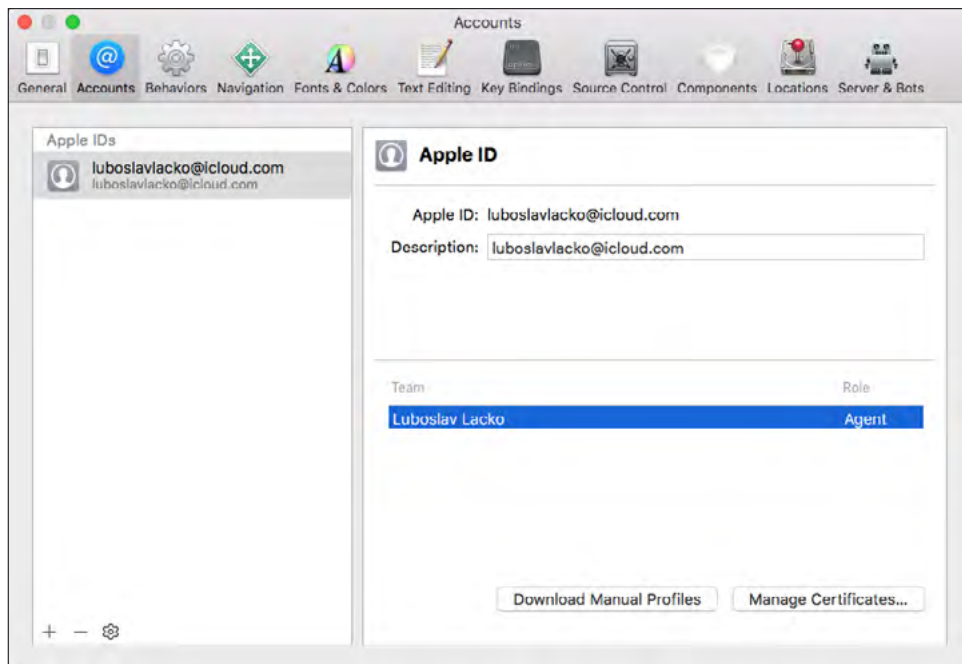
Poznámka: Bezplatný účet umožňuje spustit aplikace pro iOS pouze na jednom fyzickém zařízení. Pokud chcete aplikace distribuovat na více zařízení nebo je zveřejnit v aplikačním obchodě App Store, musíte se zaregistrovat do programu vývojářů společnosti Apple.

Jestliže už máte účet vývojáře, je potřeba s ním propojit vývojové prostředí Xcode. V nabídce Xcode aktivujte volbu **Preferences**. Zobrazí se dialog s několika kartami. Vyberte kartu **Accounts**. Je umístěna jako druhá zleva vedle karty **General**. Klepněte na tlačítko + v levém dolním rohu zatím prázdného seznamu účtů a z nabídky vyberte položku **Apple ID**.



Obrázek 1.18: Přidání vývojářského účtu do vývojového prostředí Xcode

Po zadání přístupových práv se váš vývojářský účet zobrazí v seznamu a odtěd jste připraveni spustit a ladit aplikace na fyzickém zařízení se systémem iOS.



Obrázek 1.19: Vývojářský účet přidáný do vývojového prostředí Xcode

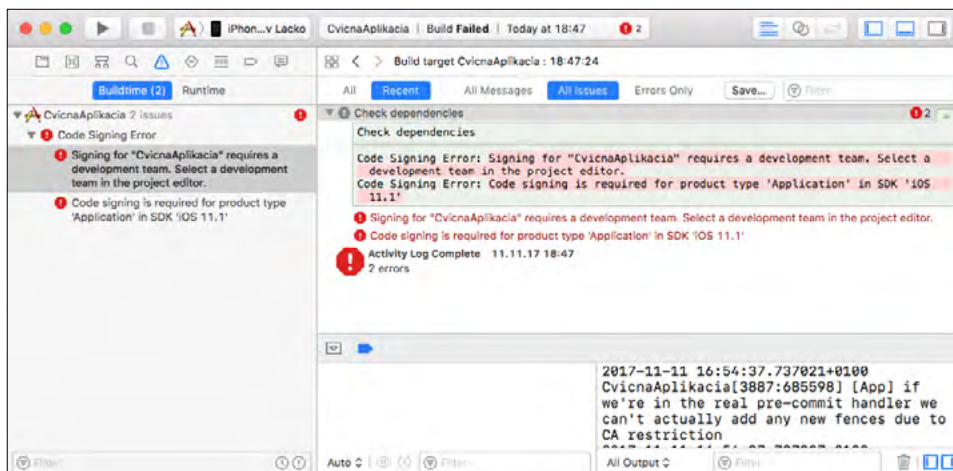
Připojte zařízení se systémem iOS k počítači Mac pomocí vhodného kabelu USB-Lightning. Na nových počítačích MacBook s konektory USB-C budete potřebovat redukci z USB-C na USB-A nebo kabel s konektorem USB-C na jedné a Lightning na druhé straně. Vývojové prostředí Xcode automaticky stáhne potřebné informace ze zařízení a jeho název se zobrazí v nabídce zařízení a simulátorů, na kterých můžete vaši aplikaci spustit.

Hardwarová zařízení se obvykle nacházejí v horní části seznamu nad simulátory. Vyberte připojené zařízení a znovu sestavte a spusťte aplikaci.

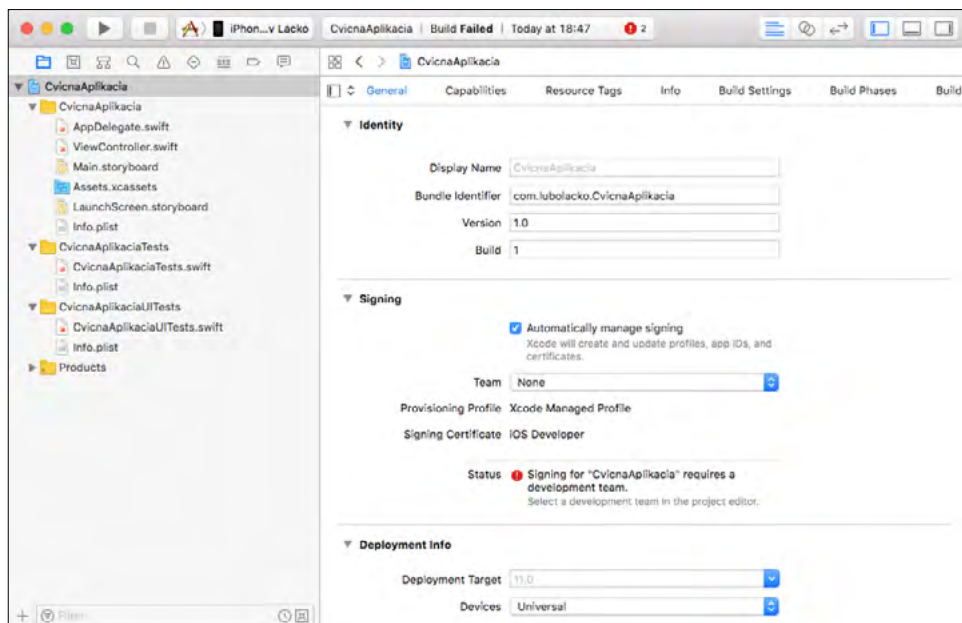
Pravděpodobně se zobrazí různá upozornění, v tomto případě nutnost zadání parametru „Development team“, který jsme při vytváření projektu aplikace nechali nevyplněný. Tento parametr můžete zadat dodatečně. Ve středním pracovním okně aplikace je potřeba zobrazit parametry projektu. Tyto parametry zobrazíte tak, že v levém okně **Navigator** klepnete na první položku v hierarchii – na název projektu. Parametr **Team** se zadává v části **Signing**. Všimněte si, že je zde ikonka výstražky informující o nezadaném parametru **Team**.



Obrázek 1.20: iPhone v nabídce zařízení, na kterých je možné spustit aplikaci ve vývojovém prostředí Xcode



Obrázek 1.21: Upozornění, že v projektu aplikace není přiřazen parametr „Development team“



Obrázek 1.22: Parametr Team se zadává v části Signing

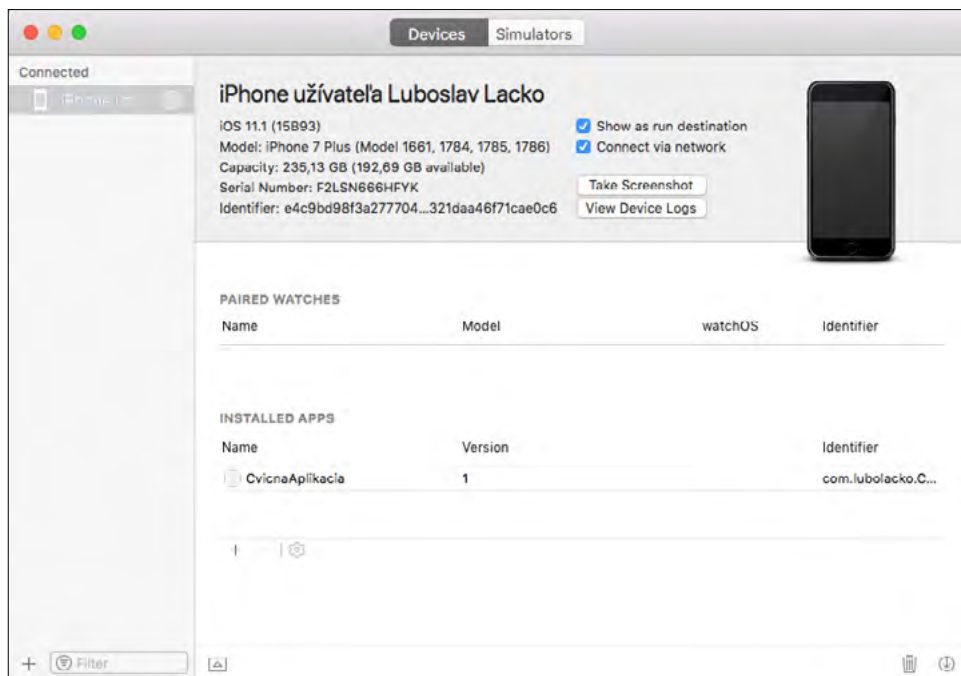
Vývojářský účet, se kterým jste předtím s Xcode spárovali, najdete v rozevíracím seznamu **Team**. Po výběru účtu se v položce **Status** změní upozornění na:

Device "iPhone uživatele'a Luboslav Lacko" isn't registered on the developer portal.
The device must be registered in order to be included in a provisioning profile.

Pod upozorněním je tlačítko **Register Device**, kterým vaše připojené zařízení s iOS zaregistrujete. Cílem registrace je vytvořit podpisový certifikát zařízení. Xcode ještě stáhne informace na ladění aplikací na vašem zařízení a vše je připraveno na spuštění aplikace. V horní části pracovní plochy můžete sledovat průběh sestavení projektu a jeho zavedení do simulátoru.

Svoje zařízení s iOS můžete k vývojářskému počítači Mac připojit i bezdrátově pomocí lokální sítě. Při konfigurování bezdrátového připojení musí být zařízení s iOS připojeno přes USB, aby bylo v seznamu připojených zařízení. V nabídce reálných zařízení a simulátorů aktivujte položku **Add Additional Simulators**. Na kartě **Devices** označte volbu **Connect via network**.

Podmínkou fungování bezdrátového připojení je, aby počítač i zařízení s iOS bylo připojené ke stejné síti Wi-Fi. Nyní můžete odpojit USB a otestovat, zda je zařízení s iOS i nadále v seznamu připojených zařízení.



Obrázek 1.23: Nastavení bezdrátového připojení zařízení s iOS k vývojářskému počítači

Ve většině případů tento postup na konfiguraci bezdrátového připojení stačí. Pokud připojení nefunguje, problém bude v konfiguraci sítě, především ve firmách, kde správce systému zavedl určitá omezení. V takovém případě otevřete okno **Devices Simulators**, podržte tlačítko **Control**, klepněte na zařízení a v zobrazené rozevírací nabídce klepněte na **Connect via IP Address**. Potom budete muset zjistit IP adresu zařízení, zadejte ji a klepněte na tlačítko **Connect**.

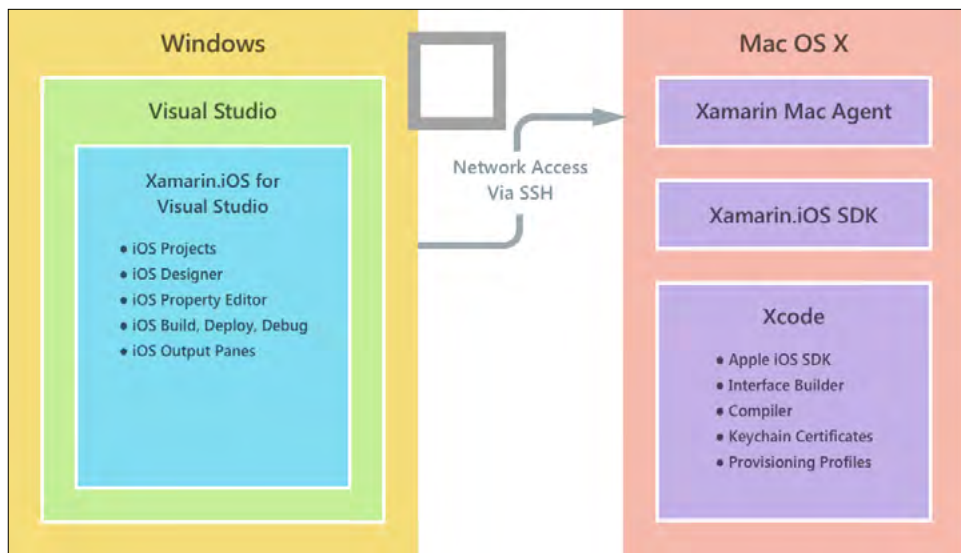
Po spuštění aplikace na simulátoru a na reálném zařízení máte jistotu, že vývojové prostředí je správně nakonfigurované a funguje i připojení reálného zařízení.

Vývoj aplikací pro iOS ve Visual Studiu 2017

Publikace je primárně zaměřená na vývoj aplikací pro iOS ve vývojovém prostředí Xcode. Tato část je koncipovaná více jako informativní než praktická, abyste získali přehled o možnostech vývoje v alternativním vývojovém prostředí.

Jak už bylo zmíněno, přestože můžete vyvíjet aplikaci pro iOS i ve Visual Studiu 2017 s využitím technologie Xamarin, bez počítače s macOS se ani v tomto případě neobejdete. Oba počítače musí být síťově propojené prakticky během celého vývoje. Bez připojení k macOS se spuštěným Xcode nedokážete vizuálně navrhovat uživatelské rozhraní v souboru *storyboard* ani spouštět aplikace na simulátoru. Visual Studio pro Windows můžete mít nainstalované na

počítači Mac ve virtuálním systému Windows spuštěném přes virtualizační platformu (Parallels, VMware, Oracle VirtualBox) nebo na počítači s Windows na té samé síti jako počítač s macOS. Samozřejmě můžete použít vývojové prostředí Visual Studio 2017 nainstalované přímo na počítači macOS. Verze pro tuto platformu je k dispozici na stránkách Visual Studia.



Obrázek 1.24: Propojení vývojářských počítačů s Windows a macOS

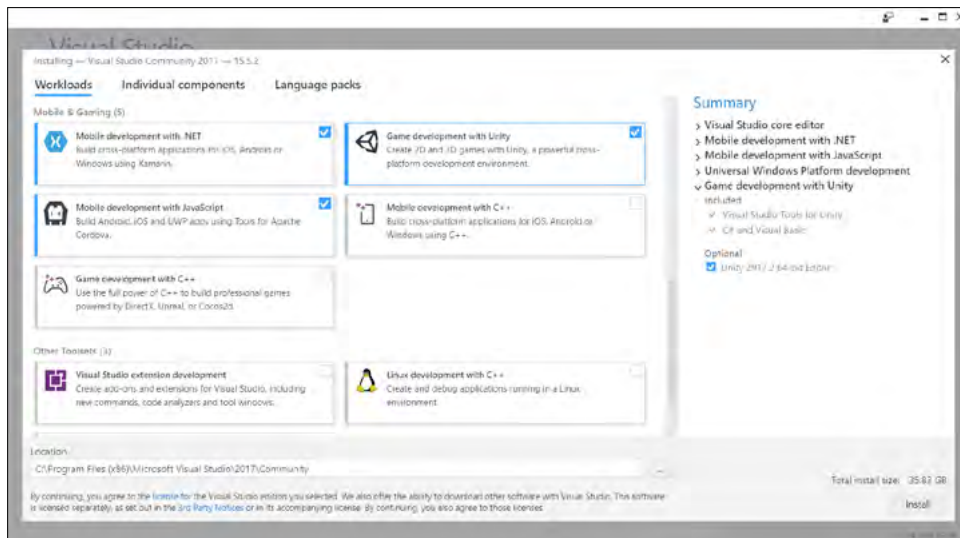
Konfigurace propojení je popsána na https://developer.xamarin.com/guides/ios/getting_started/installation/windows/connecting-to-mac/.



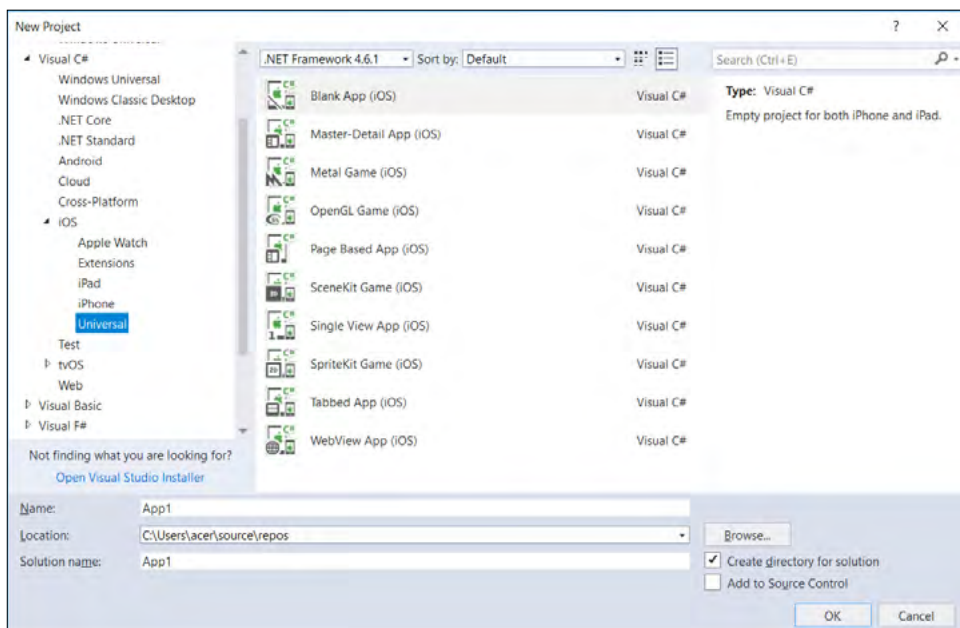
Poznámka: Aplikace pro iOS s využitím technologie Xamarin můžete vyvíjet i ve verzi Visual Studio 2017 Community Edition, které je k dispozici zdarma.

Po úspěšném propojení můžete spustit Visual Studio a vybrat si z některých typů projektů.

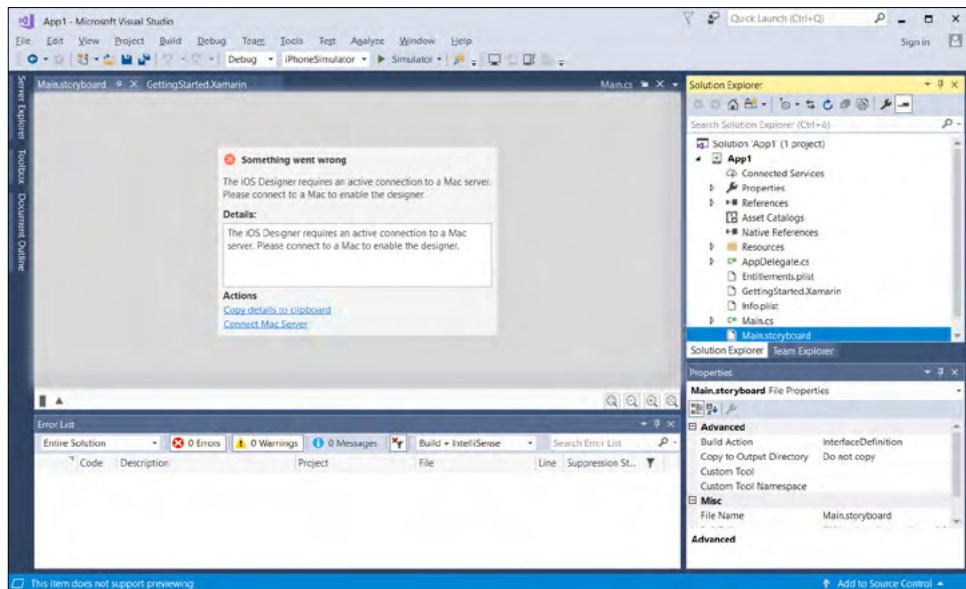
Po vytvoření projektu pro iOS vás Visual Studio upozorní na nutnost propojení s macOS. Bez tohoto propojení nebudete moci otevřít ani soubor *Main.storyboard* a navrhovat uživatelské rozhraní aplikace.



Obrázek 1.25: Komponenty, které je potřeba nainstalovat společně s Visual Studiem



Obrázek 1.26: Bohatá nabídka šablon aplikací pro iOS



Obrázek 1.27: Po vytvoření projektu pro iOS vás Visual Studio upozorní na nutnost propojení s macOS

Při návrhu uživatelského rozhraní si budete muset zvyknout na různé názvy ekvivalentních prvků uživatelského rozhraní pro různé platformy. Ekvivalenty nejčastěji používaných prvků jsou na obrázku. Také různé platformy využívají různé metody navigace a filozofii rozmístění prvků uživatelského rozhraní a jejich přizpůsobení velikosti a rozlišení displejů.

Windows	Android	iOS
Button	Button	UIButton
CheckBox	CheckBox	UISwitch
ComboBox	Spinner	UIPickerView
Image	ImageView	UIImageView
Label	TextView	UILabel
ListBox	ListView	UITableView
ProgressBar	ProgressBar	UIProgressView
Slider	Slider	UISlider
TextBox	EditText	UITextField

Obrázek 1.28: Názvy prvků při multiplatformním vývoji

Xamarin

Základní filozofie fungování aplikace je stejná, i Xamarin využívá architekturu Model-View-Controller. Každá technologie má svoje přednosti. Platforma Xamarin je vhodná hlavně pro aplikace, kde je více aplikační logiky na pozadí než v prezentační vrstvě. Umožňuje vývojářům vytvořit aplikaci pro všechny rozšířené platformy s tím, že uživatelské rozhraní je podobné, respektující hlavní designové principy z platform Windows, Android a iOS. Proto pokud potřebujete vytvořit aplikaci, které dominuje prezentační vrstva a obsahuje poměrně málo kódu aplikační logiky, je vhodné zvážit, zda nepoužít nativní vývoj. Také ne vždy je výhodné, když aplikace na všech platformách vypadá úplně stejně. Přeci jen jsou uživatelé jednotlivých platform zvyklí na specifický design a filozofii ovládání.

Klíčovou technologií, kterou využívá Visual Studio a v něm vytvořené aplikace, je .NET Framework. Projekt Mono je open-source projekt umožňující simulaci .NET frameworku na jiných platformách, než je Windows. Pomocí této technologie je možné vytvářet programový kód, který je psaný v jazyce C#, ale spustitelný i v operačních systémech bez nativní podpory .NET frameworku. Cílem tohoto projektu byla možnost spouštět aplikace psané v .NET frameworku na jiných platformách.

Technologie Mono je základem Xamarinu. V roce 2016 Microsoft koupil Xamarin a integroval ho do vývojového prostředí Visual Studio. Xamarin.Forms je framework, který integruje frameworky Xamarin.Android, Xamarin.iOS a vývoj pro Windows. Kromě datového modelu a prezentační vrstvy unifikuje a umožňuje sdílet i aplikační vrstvu, která se stará o otevírání nových obrazovek, navigaci mezi obrazovkami a předávání dat mezi dvěma uživatelskými rozhraními a business vrstvou.

Xamarin.iOS umožňuje vytvářet aplikace pro operační systém iOS. Framework potřebuje pro svoje fungování propojení s počítačem s operačním systémem macOS s nainstalovaným vývojovým prostředím Xcode. Na kompilaci se používá princip AOT (Ahead of Time Compilation), čili kód v programovacím jazyce C# se kompiluje do nativního kódu pro platformu ARM, kterou používají zařízení s iOS.

Technologie Xamarin umožňuje pracovat s nativními komponentami a periférií systému, pro který je aplikace určena. To je na jedné straně velká výhoda, protože aplikace může využívat všechny komponenty a funkce jako při nativním vývoji, na druhé straně i nevýhoda, protože vývojář musí na odpovídající úrovni znát všechny platformy, pro které kód vyvíjí. V oblasti uživatelského rozhraní si tedy příliš nepomůže, může však mezi verzemi aplikace pro různé operační systémy sdílet kód business logiky.

Výhody:

- Multiplatformní vývoj s využitím stejného vývojového prostředí
- Uživatel má dojem z používání aplikace, jako by byla nativní
- Výkon se blíží nativním aplikacím
- Využití možností hardwaru stejně jako u nativní aplikace. Nejen co se týče výkonu, ale i úsporného provozu či využití hardwarových komponent, například na telefonování
- Microsoft poskytuje technologii Xamarin pro hobby vývojáře a malé týmy k nekomerčnímu využití zdarma

Nevýhody:

- Trochu opožděná podpora funkcí v nových verzích iOS
- Není možné využívat nativní knihovny třetích stran

Programovací jazyk Swift

Programovací jazyk Swift

Ještě před dvěma roky se aplikace pro mobilní zařízení využívající operační systém iOS programovaly ve vývojovém prostředí Xcode v programovacím jazyce Objective-C. Je to jedna z objektově orientovaných variant programovacího jazyka C. Pokrok však jde kupředu a modernizaci se nevyhnou ani programovací jazyky. Cílem modernizace je zjednodušení a zrychlení vývoje a přehlednější kód, ve kterém programátoři už na první pohled odhalí některé chyby. Mezi největší výhody Swiftu patří absence správy paměti, rychlost, redukce délky kódu a vynikající čitelnost kódu.

Programovací jazyk Swift je poměrně mladý, poprvé byl prezentován v červnu roku 2014 na konferenci Apple WWDC (Worldwide Developers Conference). V roce 2014 byla k dispozici i první verze. Byla integrována do vývojového prostředí Xcode ve verzi 6. O rok později Apple zveřejnil zdrojový kód Swiftu, od té doby je tedy tento programovací jazyk typu open-source. Zdrojový kód Swiftu je ve veřejném repozitáři <https://github.com/apple>.

Swift je bezpečný, rychlý a interaktivní programovací jazyk, který kombinuje nejlepší moderní programátorské myšlení s osvědčenými postupy, ať už softwarových inženýrů Applu či open-source komunity. Kompilátor je optimalizovaný na výkon a jazyk je optimalizovaný na efektivní vývoj. Apple jazyk Swift v době jeho uvedení porovnával na svých stránkách s jinými jazyky z hlediska výkonu a tvrdil, že Swift je 2,6krát rychlejší než Objective-C a 8,4krát rychlejší než Python.

Velmi důležité je, že Swift se dá velmi rychle naučit, neobsahuje žádné hůře pochopitelné speciality, jako jsou ukazatele v programovacím jazyce C. Navíc Apple umožnil experimentovat

V této kapitole:

- Naučíte se základní principy, syntaktická pravidla a konstrukce programovacího jazyka Swift.
- Naučíte se, jak vyzkoušet fungování bloku mimo aplikace v konzoli Playground, která je součástí vývojového prostředí Xcode.
- Osvojíte si práci s datovým typem Optional.
- Získáte přehled o cyklech a podmínkách.
- Naučíte se základní principy objektově orientovaného programování.

s kódem ve speciálním prostředí Playground, takže se můžete naučit jazyk bez toho, abyste jako začátečník museli řešit aspekty projektu reálné aplikace. První aplikaci můžete začít vytvářet až po osvojení Swiftu na přiměřené úrovni.



Tip: Programovací jazyk se dá učit i na iPadu. V aplikačním obchodě pro iPad najdete aplikaci *Swift Playground*, se kterou se mohou naučit základy programovacího jazyka Swift začátečníci, a dokonce i děti.

Kód ve Swiftu je kompilovaný a optimalizovaný tak, aby co nejvíce využil moderní hardware mobilních zařízení Applu. Syntaxe a standardní knihovna byly navrženy na základě osvědčeného principu, aby způsob zápisu kódu byl jasný a přehledný. Kombinace bezpečnosti a rychlosti dělá ze Swiftu ideální volbu pro všechny typy aplikací od nejjednodušších až po ty náročné, jako jsou například práce s databázemi či emulátor jiného operačního systému.

Tvůrci Swiftu se kromě jednoduchosti soustředili i na bezpečnost, tento programovací jazyk je tedy z hlediska bezpečnosti poměrně striktní. Vyžaduje například inicializaci proměnných před jejich prvním použitím, definování hodnot konstant či neumožňuje, aby objekty bez toho, aby to vývojář při inicializaci vysloveně připustil, měly hodnotu `nil`. Pokud to pro některý objekt vědomě připustí, musí takovouto situaci v kódu náležitě ošetřit.

Migrace na Swift je pro vývojáře, kteří používají některý z programovacích jazyků C, C++, C#, případně Java, velmi jednoduchá.

Zpětná kompatibilita s Objective-C

Jelikož se do roku 2015 aplikace pro iOS vyvíjely hlavně v jazyce Objective-C, ve vývojovém prostředí Xcode je zabudovaná kompatibilita s Objective-C. V praxi to znamená, že v nových projektech využívajících moderní Swift je možné využívat knihovny a frameworky vytvořené v Objective-C. Na jejich integraci je potřeba definovat hlavičkový soubor, takzvaný bridging header, který umožní použití kódu Objective-C v projektech psaných ve Swiftu. Příchod Swiftu neznamená žádné problémy s vašimi aplikacemi v Objective-C, protože v jedné aplikaci pro iOS může existovat kód Objective-C i Swift. Není problém se psaním nových modulů v programovacím jazyce Swift, které jsou kompatibilní s existujícím kódem v Objective-C.

První pokusy s jazykem Swift ve vývojovém prostředí Xcode

Xcode od verze 6, tedy od integrace Swiftu (aktuální verze v době psaní této publikace byla 9), nabízí nový nástroj s názvem Playground, jehož účel vyplývá už z jeho samotného názvu. Slouží k experimentování s kódem v tomto programovacím jazyce a poskytuje možnost okamžitě vidět změny bez toho, abyste sestavili aplikaci a spouštěli ji na simulátoru nebo na reál-

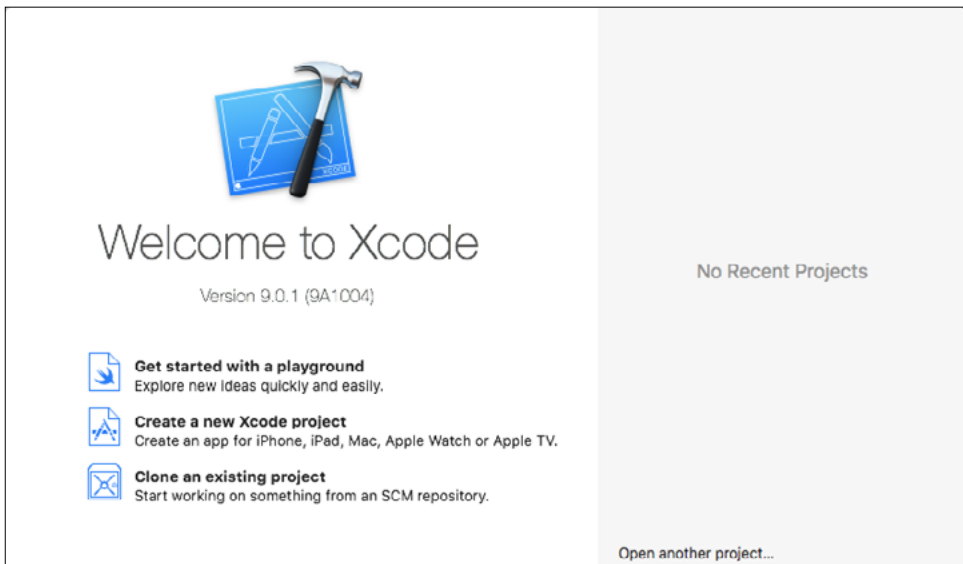
ném zařízení. Playground navíc zobrazuje i případné chyby a upozornění, takže i takto pomáhá při osvojování nového programovacího jazyka. Navíc má nejen textový, ale i grafický výstup.

Playground je vynikajícím nástrojem na pokusy v programovacím jazyce Swift, nejen když se ho učíte. Kdykoliv si v něm můžete vyzkoušet i složitější bloky kódu, zda provádějí to, co se od nich očekává, zda příslušné hodnoty vstupů vyvolají patřičnou odezvu a výstupy během vývoje. Nyní už nemusíte vytvářet prázdný projekt, v něm testovat fragmenty kódu a neustále sestavovat a spouštět aplikaci. Vše potřebné můžete vyzkoušet v interaktivním režimu Playground, kde se kód průběžně kompiluje a provádí. Po každé změně kódu se automaticky spouští překlad a okamžitě se zobrazí výsledek. Kód i výsledek se navíc zobrazují v jednom okně vedle sebe.



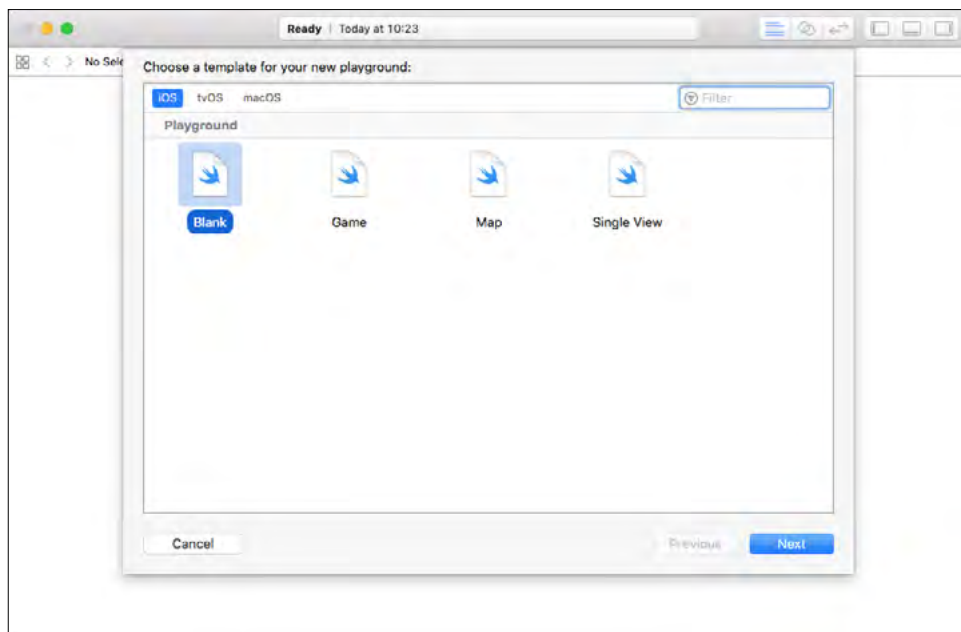
Poznámka: Playground umožňuje otestovat nový algoritmus nebo grafickou rutinu, aniž by bylo nutné vytvářet účelovou aplikaci pro zařízení s iOS.

Po spuštění vývojového prostředí Xcode se zobrazí dialog s nabídkou činností. Na interaktivní pokusy s novým programovacím jazykem Swift aktivujte volbu **Get started with a playground**. Zobrazí se nabídka platform a projektů.



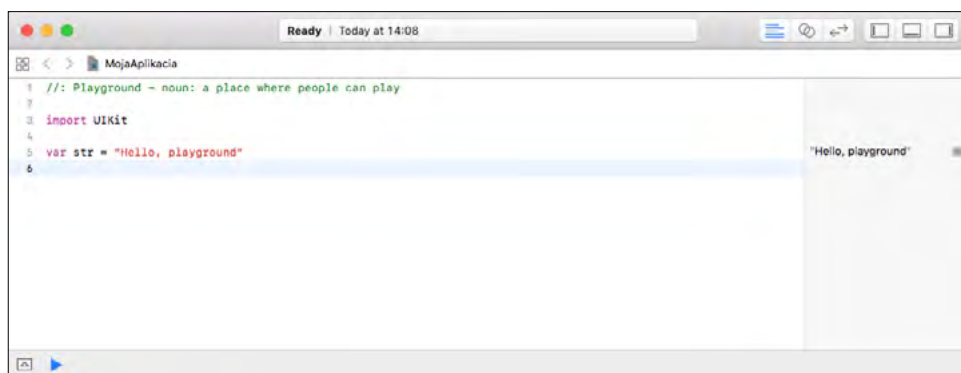
Obrázek 2.1: Úvodní dialog po spuštění vývojového prostředí Xcode

V horní části nabídkového dialogu jsou karty pro jednotlivé platformy. Playground je k dispozici pro iOS, macOS a tvOS. Na kartě iOS jsou k dispozici šablony typu **Blank**, **Game**, **Map** a **Single View**. Na první pokusy s programovacím jazykem Swift se hodí šablona **Blank**.



Obrázek 2.2: Nabídka typů projektů pro iOS

Pokud spouštíte Xcode poprvé, zobrazí se dialog, v němž vám operační systém oznamuje, že aplikace Xcode potřebuje převzít kontrolu nad jiným procesem pro ladění, a žádá vás, abyste tuto akci potvrdili pomocí svého přístupového hesla. Následně vás Xcode vyzve, abyste povolili vývojářský mód pro svůj počítač s macOS. Tyto úkony jsou podrobně popsány v první kapitole. Zobrazí se okno aplikace Xcode, ve kterém můžete zkoušet příklady při učení se programovacího jazyka Swift.



Obrázek 2.3: Vývojové prostředí Xcode v režimu Playground pro iOS

V okně je jednoduchý kód, který obsahuje jeden komentář a řádek, v němž se do proměnné `str` přiřazuje textový řetězec `"Hello, playground"`.

```
//: Playground - noun: a place where people can play
import UIKit
var str = "Hello, playground"
```

Takovýto kód neprovede po spuštění žádnou interakci s uživatelem. Zkuste ho jako úplně první pokus v programovacím jazyce Swift nahradit kódem, který vypíše textový řetězec na obrazovku, například:

```
print("Ahojte!")
```

Všimněte si, že po zadání příkazu se malá ikona s modrým trojúhelníkem vlevo ve spodní části okna na okamžik změní na šedý čtverec, pod lištou se začne zobrazovat ladicí okno, v němž se zobrazí akce, kterou programový kód provedl. V tomto případě se zobrazí text „Ahojte“. Už jste asi pochopili význam tohoto nástroje, je to jakási konzola, která zobrazí to, co provede programový kód v editačním okně.



Tip: Příkaz `Print` budete moci použít i při vývoji reálných aplikací. Vypíše text ne do kontextu uživatelského rozhraní aplikace, ale do ladicího okna vývojového prostředí.

Praktický úvod do programovacího jazyka Swift

Náš popis programovacího jazyka Swift nemá za cíl nahradit syntaktickou příručku, je primárně zaměřen na specifika, která se liší od jazyků Objective-C, Java či C#, dobře ale poslouží i zájemci o vývoj mobilních aplikací – začátečníkovi.

Už v předchozím příkladu byl uveden jediný řádek kódu:

```
print("Ahojte!")
```

I ten může být kompletním programem. Není potřeba importovat knihovny pro funkce, jako jsou vstup či výstup, matematické funkce či manipulaci s textovými řetězci. Kód napsaný na globální úrovni se používá jako vstupní bod pro program, takže nepotřebujete žádný ekvivalent funkce `main()`.

Všimněte si, že na rozdíl od programovacích jazyků C, C++, C# či Java nemusí být na konci příkazového řádku středník. Středníky je třeba psát pouze tehdy, když dáváte na jeden řádek více příkazů.



Upozornění: Na rozdíl od jiných programovacích jazyků musí být přiřazovací operátor, například `=`, oddělen od okolního textu kódu mezerami na obou stranách.

Takže všechny řádky následujícího kódu, kde nejsou kolem přiřazovacího operátoru = mezery na obou stranách, jsou chybné.

```
var str= "Ahoj"
var str ="Ahoj"
var str="Ahoj"
```

Podobně jako u většiny moderních programovacích jazyků, i ve Swiftu se bloky kódu uzavírají do složených závorek.

Komentáře mohou být jednořádkové:

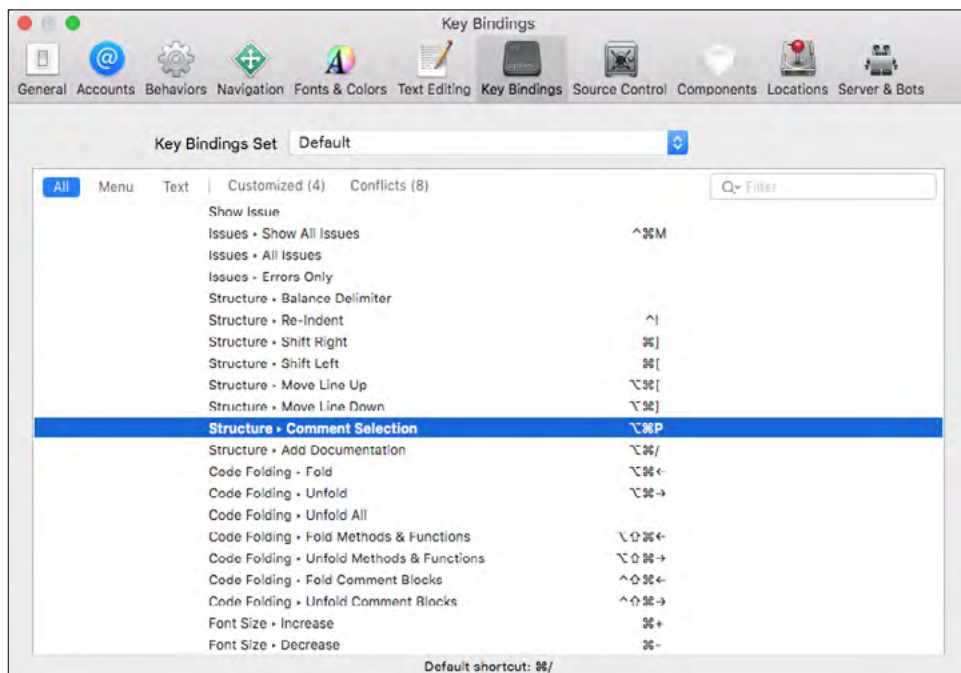
```
//Toto je komentár
```

A samozřejmě i víceřádkové:

```
/* Komentár
na viacerých riadkoch. */
```

Lze použít i komentáře vnořené. Ty s výhodou využijete, pokud potřebujete dát dočasně do komentáře blok kódu, který komentáře již obsahuje.

```
/* začiatok viacriadkového komentára
/* vnorený komentár */
koniec komentára najvyššej hierarchickej úrovne. */
```



Obrázek 2.4: Změna klávesové zkratky na komentování bloku

Často budete potřebovat jednoduchým způsobem umístit do komentáře blok kódu ve vývojovém prostředí. Pomocí lomítek a hvězdiček je to trochu nepraktické. Xcode má na vložení označeného bloku textu do komentáře klávesovou zkratku **cmd+/,** která však na některých evropských klávesnicích, mezi něž patří i česká a slovenská, nefunguje. Klávesovou zkratku tohoto úkonu můžete jednoduše změnit. V nastavení vývojového prostředí Xcode se přepněte na kartu **Key Binding**. V sekci **Editor Menu for playground** najdete položku **Structure – Comment Selection** a definujte pro tuto akci vhodnou klávesovou zkratku, která nekoliduje s jinými zkratkami. Osvědčil se například trojhmat **alt+cmd+P**.

Názvy objektů

Začneme trochu netradičně názvy. Všechny objekty, s nimiž v aplikaci pracujete, musí být pojmenovány, abyste je mohli jednoznačně identifikovat a použít.

Začínající vývojáře to u jednoduchých projektů občas svádí k použití jednopísmenných proměnných. Nejde ani tak o to, že u rozsáhlých projektů by 26 písmen poměrně rychle vyčerpali a museli by přejít na dvoupísmenné názvy proměnných, tento extrém ale ve stručnosti nedoporučujeme, jelikož si těžko po měsíci programování budete pamatovat, co se ukládá do proměnné B a co do proměnné P. A nebudete to tušit ani vzdáleně, když po půl roce budete provádět opravy, případně programový kód upravovat. Z důvodu přehlednosti kódu není příliš vhodný ani druhý extrém, rozsáhlé a vyčerpávající názvy proměnných.

Pro názvy se zaběhlo několik konvencí. Například dvoj nebo víceslovné pojmenování, kde první slovo začíná malým písmenem a druhé slovo na něj navazuje bez mezery a začíná velkým písmenem. Například `pomocnaPromenna` nebo `indikatorPriJateZpravy`. Pro tento způsob se zaběhlo pojmenování *camelCase*. Je odvozené od obrysu velblouda, který stoupá a klesá od krku přes hřbet.

Velmi dobrý a přehledný je způsob označování proměnných nazývaný Maďarská notace (zavedl ji jeden programátor Microsoftu maďarského původu), kde kromě jedno, případně dvouslovného názvu proměnné naznačíte pomocí prvního písmena i její datový typ. Například Proměnnou číselného typu můžete označovat `nSuma`, kde malé `n` symbolizuje numerický typ proměnné (od slova *number*). Proměnnou, do které se budou ukládat textové řetězce, nazvete `sJmeno` a podobně.

V programovacím jazyce Swift můžete používat v názvech i grafické symboly, takzvané emoji. Emotikon může být součástí názvu nebo může být samotně jako název.

```

let 🐶 = 8
let 🐱 = 4
let 🐭 = 2
let 🐹 = 3
let 🐰 = 1
let 🐹 = 1
let 🐱 = 1

let pocetZvieratiek = 🐶 + 🐱 + 🐭 + 🐹 + 🐰 + 🐹 + 🐱
let pocetCicavcov = 🐶 + 🐱 + 🐭 + 🐹

```

Obrázek 2.5: V názvech objektů můžete použít i emotikony (emoji)

Základní datové typy a struktury

Konstanty nebo proměnné mohou být různého datového typu. V této části představíme formou přehledné tabulky základní datové typy.

Datový typ	Rozsah	Popis
Int, UInt	Podle platformy – 32bit nebo 64bit ($\pm 2\,147\,483\,648$)	Celé číslo se znaménkem (Int) nebo bez znaménka (UInt)
Float, Double	32bit, 64bit	Číslo s desetinnou čárkou (Float), případně s dvojnásobnou přesností (Double)
Bool	true nebo false	Logická hodnota. Není možné používat číselné hodnoty 0 a 1
Character	Unicode	Jeden znak
String	Unicode	Řetězec znaků, může obsahovat i grafické emotikony

Čísla můžete zadávat i v hexadecimální či binární soustavě. Prefix pro hexadecimální soustavu je 0x a pro binární 0b. Například:

```

let hodnotaHex = 0xFF
let hodnotaBin = 0b10100011

```

Při zadávání hodnot můžete použít i takzvanou vědeckou notaci, kde je hodnota čísla zapsaná jako mantisa + exponent, například:

```

let a = 1.78e2 // 178.0
let d = 1.78e-1 // 0.178

```

Konstanty a proměnné

Konstanta je, jak vyplývá z názvu, pojmenovaná konstantní hodnota, která se za běhu kódu nemění, například název interního adresáře, kam aplikace ukládá proměnné, URL adresa zdroje údajů a podobně. Hodnoty konstant se definují za klíčovým slovem **let** ve tvaru **let konstanta: DatovýTyp = Hodnota**, přičemž datový typ není povinným údajem. Například: