Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Tvorba mobilních aplikací pro platformu Apple iPhone

Michal Kabíček

Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Akademický rok 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Michal Kabíček

obor Informatika

Vedoucí katedry Vám ve smyslu Studijního a zkušebního řádu ČZU v Praze čl. 16 určuje tuto bakalářskou práci.

Název práce: **Tvorba mobilních aplikací pro platformu Apple iPhone**

Osnova bakalářské práce:

1. Úvod
2. Cíl práce a metodika
3. Rozbor aktuálního stavu problematiky - rešerše
4. Popis platformy iPhone a specifik vyvoje aplikací na této platformě
5. Analýza ukázkové aplikace
6. Vývoj ukázkové aplikace
7. Závěr
8. Seznam použitých zdrojů
9. Přílohy

Rozsah hlavní textové části: 30 - 40 stran

Doporučené zdroje:

Oficiální dokumentace a SDK pro Apple iPhone
Dave Mark & Jeff LaMarche - Beginning iPhone 3 Development: Exploring the iPhone SDK, Apress, 2009, 978-1-4302-2459-4.
Ian Piper - Learn Xcode Tools for Mac OS X and iPhone Development, Apress, 2010, 978-1-4302-7221-2.
James Brannan - iPhone SDK Programming: A Beginner`s Guide, McGraw-Hill, 2010, 978-0-07-162650-7.
Mark Dalrymple & Scott Knaster - Learn Objective–C on the Mac, Apress, 2009, 978-1-4302-1815-9.

Vedoucí bakalářské práce: **Ing. Jiří Brožek**

Termín odevzdání bakalářské práce: březen 2011

.....
Vedoucí katedry

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
Provozně ekonomická fakulta

.....
Děkan

V Praze dne: 28. 2. 2011

Evidováno děkanátem PEF pod č.j.: KII-177-10B

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Tvorba mobilních aplikací na platformu Apple iPhone" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. března 2011 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Brožkovi za poskytnuté odborné rady a za pozornost, kterou věnoval mé bakalářské práci.

Tvorba mobilních aplikací pro platformu Apple iPhone

Mobile application development for Apple iPhone platform

Souhrn

Tato bakalářská práce se zabývá problematikou vývoje aplikací pro platformu Apple iPhone. Práce je rozdělena na dvě části. První část je teoretická a je v ní obecně představena mobilní platforma a podrobněji pak platforma iPhone. Dále pak obsahuje popis vývojového prostředí Cocoa Touch a základních nástrojů iOS SDK, které se při vývoji aplikací nejčastěji využívají. Druhá část je praktická a obsahuje popis vývoje aplikace pro platformu iPhone s využitím nástrojů popsaných v první části.

Summary

This bachelor thesis deals with application development for Apple iPhone platform. The thesis is divided into two parts. The first part is theoretical and it contains general descriptions of the mobile platform and more specifically the iPhone platform. Also includes a description the of application programming interface Cocoa Touch and basic iOS SDK tools, which are usually used for application development. The second part is practical and it includes a description of the application development for the iPhone platform using the tools described in the first part.

Klíčová slova: iPhone, Apple, aplikace, vývoj, SDK, iOS, Xcode, smartphone, Kalkulačka, Cocoa Touch;

Keywords: iPhone, Apple, application, development, SDK, iOS, Xcode, smartphone, Calculator, Cocoa Touch;

1 Úvod

V poslední době jde vývoj v oblasti mobilních zařízení velmi rychle kupředu stejně jako v celé sféře informačních technologií. Zejména vývoj mobilních telefonů je v posledních letech velmi výrazný. Mobilní telefony, které před několika lety naplňovaly naší představu o budoucnosti mobilních zařízení, jsou dnes minulostí.

Tyto telefony byly nahrazeny malými a elegantními smartphony. Smartphony v sobě obsahují hned několik zařízení – telefon, kapesní počítač, fotoaparát, multimediální přehrávač, navigaci atd. Jsou vybaveny výkonným hardwarem a jejich výpočetní výkon se blíží stolním počítačům před ani ne deseti lety. Tyto vlastnosti otevírají nové možnosti jejich využití. Jednou z nich, velmi oblíbenou, se stal vývoj aplikací samotnými uživateli. Tvorbu aplikací třetích stran jako první umožnila společnost Apple Inc. s platformou iPhone, která velmi výrazně ovlivnila vývoj na trhu s mobilními telefony.

Vývoj aplikací na mobilní platformu však vyžaduje odlišný přístup než při tvorbě aplikací na ostatní platformy.

2 Cíl práce a metodika

Práce je rozdělena na dvě části. První část je zpracována jako literární rešerše z tištěných a online zdrojů. Zde je obecně představena mobilní platforma a podrobněji pak platforma Apple iPhone. Dále je popsáno vývojové prostředí včetně popisu základních nástrojů, které se při tvorbě aplikací pro tuto platformu využívají. Druhá část je praktická a obsahuje jednotlivé kroky vývoje aplikace pro tuto platformu.

2.1 Cíl práce

Cílem této práce je popsat vývoj aplikace na mobilní platformu Apple iPhone. Tento cíl lze rozdělit na tři části:

- Představit platformu iPhone a specifika vývoje na tuto platformu
- Popsat základní nástroje a vývojové prostředí
- Vytvořit ukázkovou aplikaci Kalkulačka

2.2 Metodika

Vývoj aplikace lze shrnout do následujících dílčích kroků:

- Tvorba zdrojového kódu jednotlivých tříd aplikace
- Návrh a tvorba vzhledu aplikace
- Propojení zdrojového kódu a grafické části
- Testování aplikace

Aplikace byla vytvořena s využitím nástrojů iOS SDK společnosti Apple Inc. Byly použity tyto nástroje: Xcode, Interface Builder a iOS Simulator.

3 Přehled řešené problematiky

3.1 Mobilní platforma

Mobilní telefon lze definovat jako zařízení fungující na principu normálního telefonu s možností použití ve volném prostoru. První komerční mobilní telefon, který se shodoval s touto definicí, byla Motorola DynaTAC 8000X. Na trh byla uvedena v roce 1983. Její cena se v přepočtu pohybovala okolo 100 000 Kč, vážila téměř 800g, měřila 25 cm a při plně nabité baterii vydržela v provozu asi 30 minut. Dnes se tyto parametry mohou zdát směšné, ale v době jejího vzniku se jednalo o technický zázrak.[1]

Postupem času šel technický vývoj mobilních telefonů rychle kupředu (viz Obrázek 1). V 90. letech se začaly celosvětově rozšiřovat také mobilní sítě, vznikaly nové a pokrývaly stále větší území. Nabízely nové služby, například SMS nebo WAP, který byl implementován do mobilních sítí a telefonů koncem 90. let a umožnil přístup k internetu. V té době začaly mít mobilní telefony klasickou podobu. Zářným příkladem byl model Nokia 5110. Mezi dnešní trendy patří hlavně dotykové telefony, které v podstatě nahradily klasickou konstrukci telefonů. Dalším významným mezníkem je nástup smartphonů, neboli „chytrých“ telefonů s operačním systémem.

Obrázek 1 - Vývoj mobilní platformy

[1] Motorola, *Cell Phone Development*

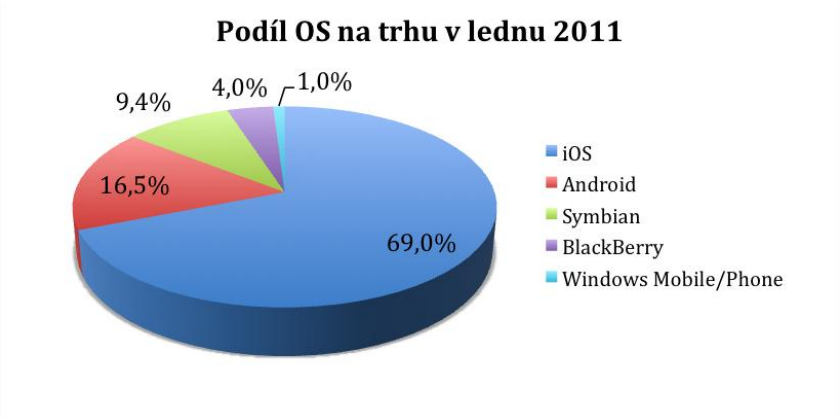
3.2 Smartphone

Smartphone je telefon, který má operační systém (OS) rozšiřitelný o další aplikace, dotykový displej, softwarovou nebo hardwarovou QWERTY klávesnici a stálé připojení k internetu. V roce 2001 uvedla firma Nokia na trh telefon, který měl vlastnosti smartphonu. Byl to model Nokii 9210 Communicator s operačním systémem Symbian OS v6.0 a v podstatě se jednalo o kombinaci mobilního telefonu a PDA (Personal Digital Assistant), které se dříve běžně používaly samostatně.[2] Konkurence na sebe nenechala dlouho čekat a na trhu se objevilo mnoho nových smartphonů a operačních systémů. V současné době smartphony ovládly trh mobilních telefonů.

3.2.1 Operační systémy smartphonů

Dnes existuje celá řada operačních systémů. Mezi ty konkurenceschopné patří Symbian, Windows Phone, bada, iOS a Android (viz Obrázek 2). Jak už bylo řečeno, jednou z vlastností operačního systému je jeho rozšiřitelnost. To umožňuje jednotlivým vývojářům nebo společnostem rozšiřovat operační systém o vlastní aplikace, ale není to tak jednoduché. Každý operační systém je většinou spojený s konkrétním výrobcem mobilních telefonů a každý smartphone má určitý hardware. Tím je trh pro vývojáře velmi nejednotný a vyvíjet aplikace pro všechny smartphony na trhu je obtížné, ne-li nemožné. Naopak, pro uživatele je trh velmi pestrý a dokáže vyhovět různorodým požadavkům.

Jednotlivé operační systémy využívají různé programovací jazyky, vývojové prostředí a možnosti distribuce.



Obrázek 2 - Graf podílu OS na trhu[3]

[2] LITCHFIELD S., *Defining the Smartphone*
[3] MARKETSHARE, *Mobile Operating System Market Share*

3.2.1.1 Windows Phone 7

Microsoft v roce 2000 vydal první verzi Windows Mobile, která vycházela z rodiny operačních systému Windows CE určených pro platformu Handheld PC a další mobilní zařízení. Windows Mobile byl systém určený přímo pro platformu smartphonů. Byl oblíbený hlavně ve firemní sféře podobně jako RIM od BlackBerry. Po příchodu iOS a Androidu Microsoft se svým mobilním OS včas a adekvátně nezareagoval a ztratil tak svoji pozici na trhu. V únoru 2010 byla představena nová platforma Windows Phone 7, která by měla znovu oživit mobilní platformu od Microsoftu. Co se týče vývoje aplikací, je možné je vytvářet ve Visual C++ za využití .NET Compact Frameworku a prostředí Silverlight. Distribuce a nákup nových aplikací jsou zajišťovány internetovým obchodem Windows Marketplace, který byl spuštěn v roce 2009.[4]

3.2.1.2 Android

Operační systém Android začala vyvíjet společnost Android Inc. V roce 2005 převzala tuto společnost firma Google, která dále pokračovala ve vývoji tohoto operačního systému. V roce 2008 byl uveden na trh první smartphone s operačním systémem Android. Aktuálně je Android pod open-source licencemi. Právě open-source licence přispěly k rychlému rozšíření a popularitě mezi uživateli i vývojáři. V současné době zaujímá Android společně s iOS největší část trhu. Oběma operačním systémům se jako jediným daří expandovat na větší mobilní zařízení, jako jsou tablety nebo netbooky. Vývoj aplikací je možný prostřednictvím Google API (Application Programming Interface) určeného přímo pro Android. Distribuci a zdroj nových aplikací zajišťuje Android Market, který se svou velikostí a konkurenceschopností nejvíce blíží App Storu.[5]

3.2.1.3 Symbian

Mezi nejstarší operační systémy patří Symbian. Vznikl v roce 1998 spojením společností Psion, Motorola, Ericsson a Nokia a navázal na operační systém EPOC, který vyvíjela společnost Psion. Dnes je tento operační systém pod licencí open-source a využívá ho hlavně Nokia, ale dříve se objevoval i v telefonech Siemens, Motorola nebo Sony Ericsson. Díky své tradici a společnosti Nokia donedávna patřil k nejrozšířenějšímu operačnímu

[4] HAVRYLUK M., *Mobilní OS pro experty: Vzestup a strmý pád Windows Mobile*
[5] HAVRYLUK M., *Mobilní OS pro experty: Skvělý nástup a slibná budoucnost Android OS*

systému na trhu. Možnosti vývoje na tuto platformu jsou díky licenci open-source široké. Aplikace lze vyvíjet pomocí mnoha jazyků a technologií, například: Qt C++, Symbian C++, JavaMe, Python, Flash Lite nebo Ruby. Distribuce aplikací je realizována internetovou službou Ovi Store, která ale není zdaleka tak vyspělá jako u konkurence. Ovi Store byl spuštěný v květnu 2009 a stále prochází celou řadou inovací.[6]

3.2.1.4 bada

Další z řady operačních systému je bada. Jedná se o nejnovější operační systém, který vznikl v roce 2010 a vyvinula ho společnost Samsung. Tento systém je značně podobný systému iOS a Android. Z každého si vzal to nejlepší a proto má veliký potenciál prosadit se na trhu, bohužel zatím, není moc rozšířený. Systémem bada je vybaveno jen šest modelů telefonů. Společnost Samsung však hodlá tento systém nasadit do téměř 90 % svých telefonů.[7] Současně s uvedením systému bada na trh společnost Samsung uvolnila SDK (Software Development Kit) pro vývojáře. Vývojové prostředí se nazývá Eclipse CDT a je postaveno na jazyku C/C++.[8]

3.2.1.5 iOS

Největší zastoupení na trhu má v současné době iOS. Jedná se o operační systém společnosti Apple Inc., který v roce 2007 odstartoval společně s iPhonem revoluci v operačních systémech a dotykových smartphonech vůbec. Operační systém iOS byl navržen přímo pro mobilní platformu a dnes ho lze najít ve všech mobilních zařízeních společnosti Apple Inc. jako je iPhone, iPad nebo iPod Touch.[9] Vývoj aplikací probíhá pomocí iOS SDK, který umožňuje vytvářet, testovat a ladit graficky orientované nativní aplikace. Aplikace lze distribuovat pomocí internetového obchodu App Store.[10]

3.3 Platforma iPhone

iPhone byl poprvé představen v lednu 2007 a koncem června téhož roku se začal prodávat. Vzhledem k vynikající reklamní kampani a všeobecné oblíbenosti produktů Apple se iPhone od počátku stal fenoménem a to i přes významné technické nedostatky. Hlavním

[6] HAVRYLUK M., *Mobilní OS pro experty: nahlédněte pod pokličku úspěšnému Symbianu*
[7] bada, *What is bada*
[8] bada Developers, *Getting started with the SDK*
[9] Apple Inc., *iOS 4 is the world's most advanced mobile OS*
[10] HAVRYLUK M., *Mobilní OS pro experty: zatracovaný a revoluční iOS pro iPhone*

důvodem rychlého úspěchu byl způsob ovládání iPhonu. Ovládal se pouze prsty pomocí dotykového displeje. Výjimkou je pouze tlačítko na návrat do hlavního menu a tlačítko na ovládání hlasitosti. Prostředí systému iOS bylo uživatelsky velmi přívětivé a intuitivní. Konkurenci trvalo poměrně dlouho, než stihla na tyto novinky zareagovat a tak iOS pomocí iPhonu získal velký podíl na trhu mobilních telefonů.

Jak už bylo zmíněno výše, první verze iPhonu – iPhone 2G, se začal prodávat v červnu 2007. iPhone využíval operační systém iOS 1.0. V hardwaru oproti konkurenci iPhone trochu zaostával. Využíval procesor o taktu 400 MHz, grafický koprocesor PowerVR MBX 3D a operační paměť o velikosti 128 MB RAM a paměť pro data ve variantách 8 GB nebo 16 GB. Mezi hlavními nedostatky byla chybějící podpora 3G sítí, GPS, multitasking, MMS, nízká výdrž baterie a nemožnost nahrávání videa. I přesto byl iPhone nedostatkovým zbožím.[11]

Další verze nesla název iPhone 3G. Do prodeje se dostal v červenci 2008. Hlavní změnou byla podpora 3G sítí a GPS. Současně byla navýšena kapacita baterie a díky novému iOS verze 2.0 přibyla možnost tvorby vlastních aplikací. Stále však přetrvávala nemožnost nahrávání videa nebo posílání MMS.[12]

Inovací verze 3G byl iPhone 3GS. Tento model byl na trh uveden v červnu 2009. Nabízel rychlejší procesor o taktu 600 MHz, operační paměť RAM o velikosti 256 MB a kvalitnější grafický procesor s podporou OpenGL 2.0. Další novinkou byl nový 3Mpixelový fotoaparát s automatickým ostřením a možností nahrávat videa. Přibyla také podpora MMS, opět se zvýšila kapacita baterie a také byly nabízeny nové varianty vnitřní paměti - 16 GB nebo 32 GB.[13]

Nejnovější a prozatím poslední verzí iPhonu je model iPhone 4 (viz Obrázek 3), který se začal prodávat v červnu 2010. iPhone 4 přinesl mnoho zásadních změn. Změnil se kompletně design telefonu, byly použité nové materiály a úplně nový displej s rozlišením 960x640 pixelů, což je čtyřnásobek oproti modelu 3GS. Kompletně nový je také hardware. iPhone 4 využívá procesor Apple A4 s taktem 1 GHz, 512 MB eDRAM, výkonný grafický procesor a nový 5Mpixelový fotoaparát, který umí nahrávat videa v HD rozlišení. Velkou novinkou je také podpora multitaskingu a nová verze iOS 4.0. Díky těmto vlastnostem je

[11] NOVÁK A., *Pozor, revoluce právě začala - velký test Apple iPhone*
[12] NOVÁK A., *Exkluzivně: první česká recenze Apple iPhone 3G*
[13] NOVÁK A., *Recenze nového iPhone 3GS: koupi si pořádně rozmyslete*

o iPhone 4 obrovský zájem a Apple Inc. z tohoto důvodu není schopen uspokojit všechny zájemce. Dosud se prodalo několik desítek milionů kusů.[14]

Obrázek 3 - Vývoj iPhonu

3.3.1 Specifika vývoje aplikací na platformu iPhone

Mobilní platforma je odlišná a má svá specifika. Jedná se především o omezení, která se musí při vývoji aplikací brát v úvahu. V případě platformy iPhone se jedná například o pouze jednu aktivní aplikaci. Některé aplikace samozřejmě mohou běžet na pozadí, protože iOS podporuje multitasking, avšak jejich funkčnost je omezena. Nejedná se zde o plnohodnotný multitasking jako například na počítačových platformách (PC, Mac…).

iOS oproti počítačovým platformám nepovoluje koexistenci několika oken. Každá aplikace má přiděleno pouze jedno okno a navíc velikost okna je fixní a je omezena rozměry displeje. Úhlopříčka displeje u iPhonu 4 má velikost 3,2 palce, což je dnes průměr. Oproti konkurenci však disponuje velikým rozlišením, a to 640 x 960 dpi, které zaručuje vysokou kvalitu zobrazení. Bohužel, tento displej má i velkou spotřebu, což se negativně projevuje na výdrži baterie.[15]

Problém s kapacitou baterie vývojáře nijak neomezuje, ale hardware ano. Platí to především o pamětích – uživatelské a operační. iPhone nepodporuje paměťové karty a proto je paměť omezena buď na 16 nebo 32 GB. V dnešní době je tato kapacita paměti sice dostačující, ale v budoucnu pravděpodobně nebude. Větší problém je s operační

[14] NOVÁK A., *Opravdu první česká recenze iPhone 4, i přes vysokou cenu ho chce skoro každý*
[15] MARK D., NUTTING J., LaMARCHE J., *Beginning iPhone 4 Development*, str. 5

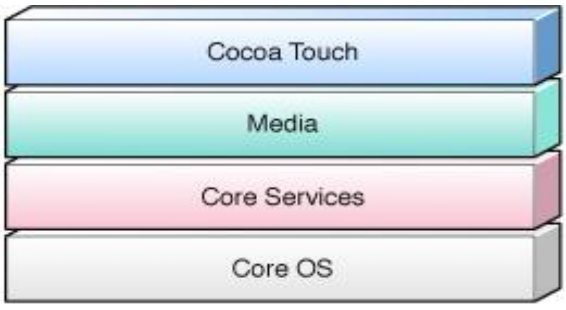
pamětí, která je 512 MB (iPhone 4) a v případě složitějších aplikací nemusí dostačovat. iPhone však používá techniku swapování. V případě náročné aplikace, která se nevejde do operační paměti, systém aplikaci rozdělí na několik částí. Části, které nejsou momentálně využívány, uloží na disk a aktuálně potřebné části uloží do operační paměti. Jednotlivé části z disku pak postupně nahrává do operační paměti podle potřeby.

Mezi další limity patří omezený přístup k datům. Aplikace třetích stran jsou nativní a mohou přistupovat jen do oblasti, která se nazývá sandbox.[16] V tomto směru je politika Applu velmi restriktivní, podobně je na tom operační systém bada od společnosti Samsung. Oproti tomu Androidu umožňuje aplikacím třetích stran přístup kamkoliv včetně soukromých nebo systémových souborů. To je velmi příjemné pro vývojáře, protože tento přístup přináší nové možnosti a umožňuje vytvářet mnohem sofistikovanější aplikace. Z pohledu bezpečnosti však tento přístup přináší velké riziko. V poslední době se stal Android terčem mnoha útoků a mnoho aplikací dostupných z Android Marketu obsahuje viry nebo jiný škodlivý software. Není překvapením, že pro Android existuje už několik antivirových programů, například AVG Mobile nebo ESET Mobile Security.[17]

3.4 Vývoj aplikací na platformu iPhone

Rok po vydání iPhonu (březen 2008) uvolnila společnost Apple Inc. na nátlak nezávislých vývojářů první verzi iOS SDK a umožnila jim tak tvořit nativní aplikace. Vývoj aplikací se stal velmi populární. Napomohla tomu také dobře fungující distribuce aplikací pomocí App Storu.

Jak už bylo řečeno v kapitole 3.3, iPhone využívá operační systém iOS. Ten je postaven na upravené verzi jádra *Mach*, které využívá operační systém Mac OS X.



Obrázek 4 - Vrstvy systému iOS[18]

[16] MARK D., NUTTING J., LaMARCHE J., *Beginning iPhone 4 Development*, str. 5
[17] MIKUDÍK R., *Google Android ohrožují trojany se zákeřnými algoritmy*
[18] Apple Inc., *iOS Overview*

Operační systém tvoří čtyři vrstvy (viz Obrázek 4), které slouží k implementaci aplikací. Nejvýznamnější vrstvou je *Cocoa Touch*. Jedná se o API a je obdobou vrstvy *Cocoa*, která je přítomná v systému Mac OS X. Vrstva *Cocoa Touch* je přímo vytvořená pro mobilní platformu a obsahuje specifické třídy, například rozhraní podporující dotykové ovládání, Wi-Fi nebo třídu pro triangulaci umístění pomocí signálu GPS. Oproti vrstvě *Cocoa* neobsahuje nepotřebné třídy.[19]

Vrstva *Media* zahrnuje podporu mediálních funkcí jako je video a zvuk, ale obsahuje například i podporu tisku. Obsahuje technologie Quartz, OpenGL a Quicktime.

Vrstva *Core Services* zajišťuje základní funkce, například práci v síti, správa paměti, napájení, správa souborů a složek a další.

Samostatné jádro (*Core OS*) pak zajišťuje nízko úrovňovou komunikaci mezi hardwarem a softwarem pomocí ovladačů.[20]

Dnešní verze iOS SDK umožňuje vývoj i na ostatní mobilní platformy společnosti Apple Inc. jako jsou iPod Touch nebo iPad a to právě díky společnému operačnímu systému iOS.

3.4.1 Objective-C

Programovací jazyk Objective-C byl navržený na počátku 80. let 20. století a vycházel z jazyka SmallTalk-80. Vznikl rozšířením jazyka C o další vrstvu, což znamená, že do jazyka C byly doplněny různé prvky a tím vznikl nový programovací jazyk umožňující tvorbu objektů a manipulaci s nimi. V roce 1988 si společnost NeXT Software Objective-C licencovala a vyvinula jeho knihovny a vývojové prostředí označované NextStep. V roce 1994 uvolnily společnosti NeXT Computer a Sun Microsystems standardizovanou specifikaci systému NextStep označovanou jako OpenStep. V prosinci 1996 byla společnost NeXT převzata společností Apple Computer.[21]

Software a prostředí NextStep/OpenStep se stalo základem nové verze operačního systému Mac OS společnosti Apple Inc. Vývojové prostředí se přejmenovalo na Cocoa. Cocoa obsahuje podporu jazyka Objective-C doplněnou různými vývojovými nástroji, například Xcode a Interface Builder a stala se velmi výkonným a efektivním prostředím

[19] KOCHAN S. G., Objective-C 2.0, kap. 19
[20] Apple Inc., *iOS Overview*
[21] TECHOTOPIA, *The History of Objective-C*

pro vývoj aplikací na Mac OS. V roce 2007 společnost Apple Inc. uvolnila aktualizaci jazyka Objective-C, kterou nazvala Objective-C 2.0.[22]

3.4.2 Frameworky

Framework je kolekce tříd, metod, funkcí a podpůrných zdrojů potřebných pro tvorbu aplikací. Přehled všech frameworků, které lze využít na platformě iPhone jsou uvedeny v následující tabulce (Tabulka 1).

Framework	Purpose
AddressBook	Accessing user's contacts
AddressBookUI	Displaying Addressbook
AudioToolbox	Audio data streams; playing and recording audio
AudioUnit	Audio units
CFNetwork	WiFi and cellular networking
CoreAudio	Core audio classes
CoreFoundation	Similar to Foundation framework, but lower level (don't use unless you absolutely must)
CoreGraphics	Quartz 2D
CoreLocation	User's location/GPS
Foundation	Cocoa foundation layer
MediaPlayer	Video playback
OpenAL	Positional audio library
OpenGLES	Embedded OpenGL (2-D and 3-D graphics rendering)
QuartzCore	Core animation
Security	Certificates, keys, and trust policies
SystemConfiguration	Network configuration
UIKit	iPhone user interface layer

Tabulka 1 – Dostupné frameworky pro iPhone[23]

Vrstva Cocoa Touch obsahuje dva základní frameworky, které jsou použity v každé aplikaci. Jsou to Fundation a UIKit framework.

Fundation framework obsahuje kořenovou třídu NSObject. Další třídy reprezentují základní datové typy jako je string (třída NSString), number (třída NSNumber), arrays (třída NSArray) a mnoho dalších.

[22] Techotopia, *The History of Objective-C*
[23] BRANNAN A. J., *iPhone SDK Programming: A Begginner's Guide*, str. 7

UIKit framework obsahuje třídy potřebné ke konstrukci uživatelského prostředí. Obsahuje mnoho ovládacích prvků, tlačítek, posuvníků, oken, ukazatelů (progres barů) a jiných vizuálních prvků, které umožňují širokou škálu úprav vzhledu a jsou speciálně navrženy pro dotykový displej a prostředí iPhone.[24]

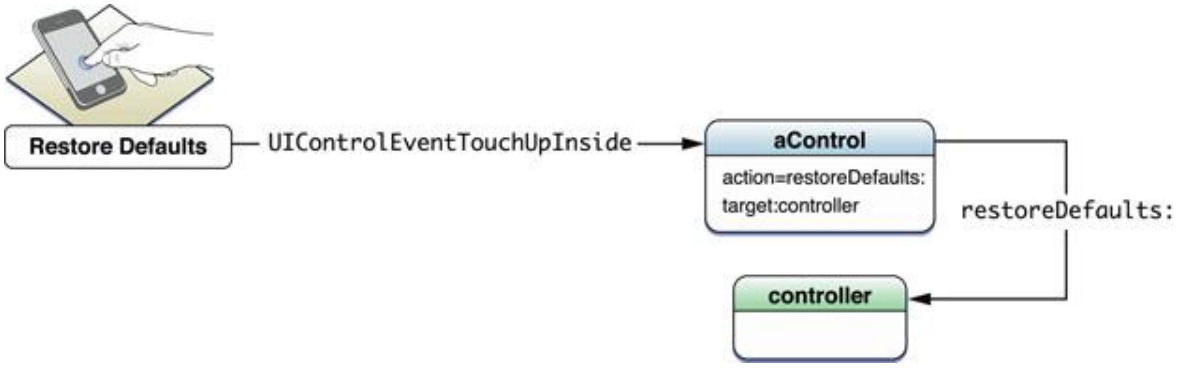
3.4.3 Návrhové vzory

Návrhové vzory jsou při vývoji aplikací a tvorby software velmi užitečné. Jedná se o řešení různých dílčích problémů, které se často v průběhu tvorby aplikace vyskytují. Vzhledem k jejich častému výskytu byla většina těchto problémů vyřešena a navržené vzory je vhodné využívat. Díky tomu lze ušetřit mnoho času a navíc jsou tyto postupy jednoduché, efektivní a především vyzkoušené.[25]

3.4.3.1 Target-Action

Návrhový vzor Target-Action (viz Obrázek 5) je při vývoji na platformu iPhone velmi často aplikován. Hlavním důvodem je jeho využití ve vztahu k dotykovému ovládání.

Objekt obsahuje informaci, která je nezbytná k odeslání zprávy jinému objektu. Informace se skládá ze dvou částí. První částí je *selector*, který identifikuje metodu, která bude použita. Druhá část se nazývá *target* a označuje objekt, který přijme zprávu, tzv. *action message*. Metody využívající tento návrhový vzor obsahují tag *IBAction* podle kterého je může Interface Builder jednoduše identifikovat. Stejným způsobem Interface Builder identifikuje proměnné pomocí tagu *IBOutlet* (viz kapitola 3.4.4.2).[26]



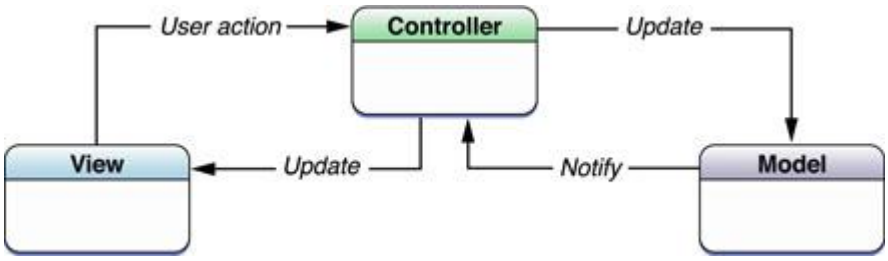
Obrázek 5 – Schéma modelu Target-Action[27]

[24] PIPER I., *Learn Xcode Tools for Mac OS X and iPhone Development*, kap. 5
[25] MERUNKA V., *Objektové modelování*, str. 99
[26] Apple Inc., *Cocoa Application Competencies for iOS*
[27] Apple Inc., *Cocoa Application Competencies for iOS*

3.4.3.2 MVC

MVC je návrhový vzor Model-View-Controller. Je to základní vzor používaný v objektově orientovaném programování a zejména ve většině aplikací pro platformu iPhone. Jeho implementace v Objective-C je trochu odlišná od původní implementace ve Smalltalku. Základní myšlenka je ale stejná. Objekty lze pomocí modelu MVC roztřídit do tří základních rolí (viz Obrázek 6). Role definují, jak bude objekt komunikovat s ostatními objekty.[28]

- *Model* - obsahuje logickou strukturu aplikace, zdrojová data a metody. Měl by být znovu použitelný v jiné aplikaci.
- *View* – uživatelské prostředí aplikace, neobsahuje žádná zdrojová data. Měl by být také znovu použitelný v jiné aplikaci.
- *Controller* – zprostředkovává komunikaci.



Obrázek 6 - Schéma modelu MVC[29]

Uživatel provede akci ve *View*, prostřednictvím *Controlleru* se změna dostane k *Modelu*, který je aktualizován změnou. Jakmile je změna provedena, *Model* informuje *Controller* a ten změní *View*.

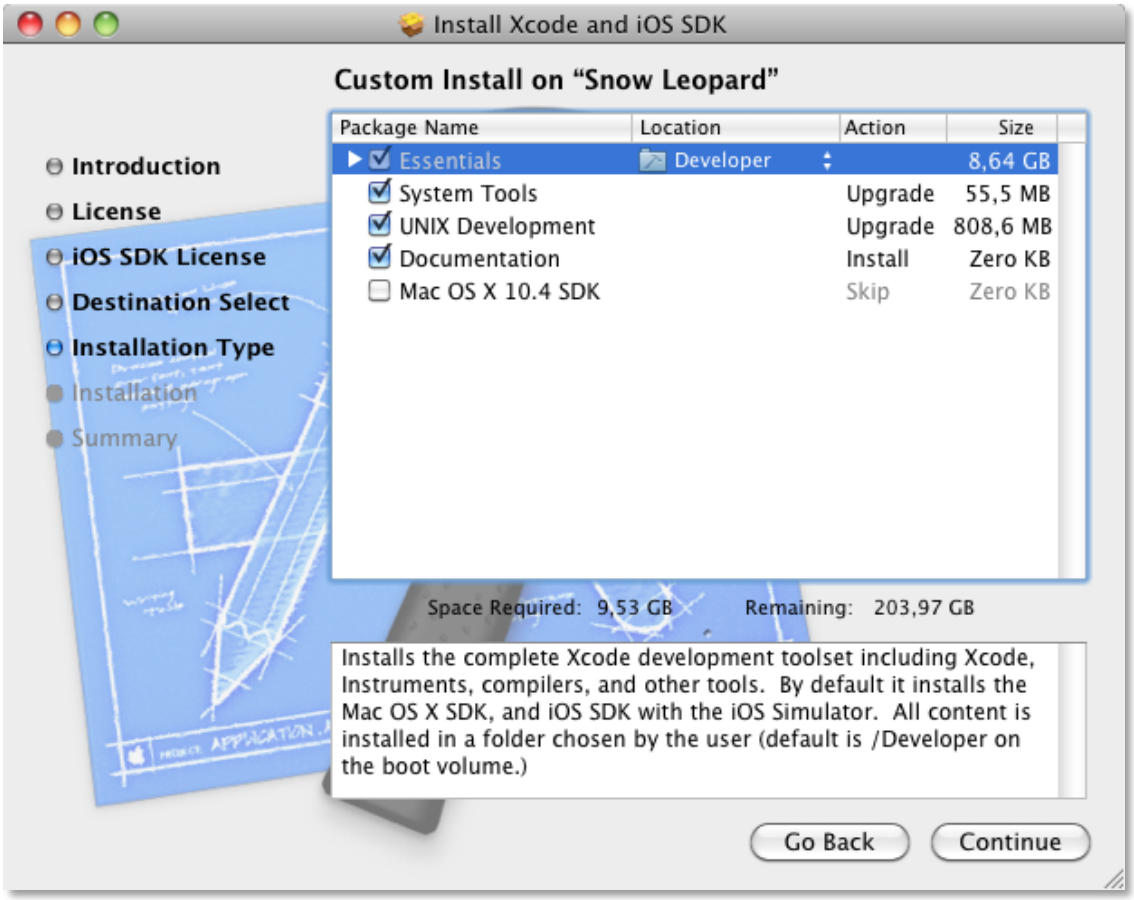
Cílem modelu MVC je rozdělit objekty do těchto tří výše uvedených kategorií. Každý nově vytvořený objekt by měl patřit do jedné z kategorií a jeho funkčnost by měla být zajištěna pomocí jiné kategorie. Například objekt z kategorie *View*, obsahující nějaký grafický prvek, by neměl obsahovat zdrojový kód (metodu). Objekt s grafickým prvkem by měl být použitelný i v jiné aplikaci. Zdrojový kód (metoda) tohoto objektu, vyjadřující jeho chování, by měl obsahovat jiný objekt, který bude v kategorii *Model*.[30]

[28] Apple Inc., *Cocoa Core Competencies*
[29] Apple Inc., *Cocoa Core Competencies*
[30] MARK D., NUTTING J., LaMARCHE J., *Beginning iPhone 4 Development*, str. 35

3.4.4 iOS SDK

iOS SDK je integrované vývojové prostředí (IDE) určené pro iOS. Jedná se o skupinu programů a nástrojů, pomocí kterých lze vytvářet aplikace nejen pro iPhone, ale i pro všechny ostatní platformy společnosti Apple Inc., které tento operační systém využívají. Prostředí je postaveno na jazyku Objective-C (viz kapitola 3.4.1).

Toto vývojové prostředí lze získat po registraci na stránkách společnosti Apple Inc., které jsou určeny pro vývojáře. Registrace jako *Apple Developer* je zdarma, pokud však dojde k testování aplikací na zařízení a popřípadě distribuci aplikace, je nutné zaplatit členský poplatek ve výši 99 $. Při distribuci lze zvolit, zda bude aplikace placená nebo zdarma. Třicet procent z ceny každé prodané aplikace připadá společnosti Apple Inc. a zbylých sedmdesát procent vývojáři aplikace.[31]



Obrázek 7 - Instalace SDK

[31] BRANNAN A. J., *iPhone SDK Programming: A Begginner's Guide*, str. 4

Instalační balíček je kompletní a obsahuje veškeré potřebné programy a nástroje pro vývoj aplikací včetně dokumentace (viz Obrázek 7).

3.4.4.1 Xcode

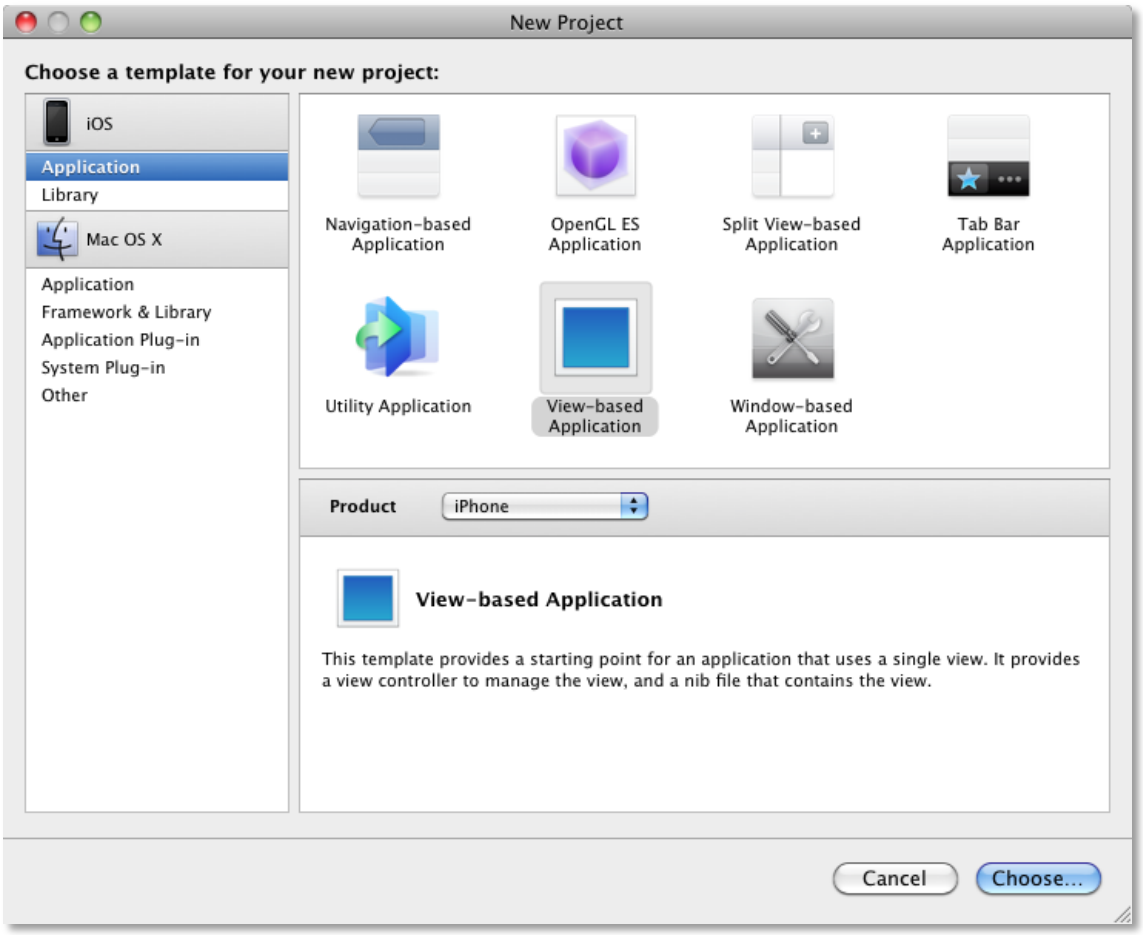
Xcode je základní aplikací iOS SDK a lze ho spustit, stejně jako všechny ostatní aplikace iOS SDK, ze složky /Developer/Applications. Po spuštění Xcode se objeví úvodní obrazovka (viz Obrázek 8).

Obrázek 8 - Úvodní obrazovka Xcode

Úvodní obrazovka slouží jako rozcestník, nabízí vytvoření nového projektu, rychlý přístup k již rozpracovaným projektům nebo přístup k návodům a dalším materiálům pro začínající vývojáře. Kliknutím na *Create a new project* začíná tvorba nové aplikace.

iOS SDK je určené pro všechny platformy s tímto operačním systémem a vzhledem k tomu je nutné nejprve zvolit zařízení, pro které bude aplikace určena. Po zvolení zařízení lze vybrat typ aplikace (viz Obrázek 9). iOS SDK podle zvoleného typu aplikace připraví šablonu, která obsahuje frameworky a základní třídy. Aplikace je pak v tomto stavu spustitelná. Například při výběru *View-based Application* vznikne aplikace, která nebude

nic umět, bude pouze zobrazovat prázdné pozadí. Pro vývojáře je to velké usnadnění, nemusí se totiž definováním typu projektu zabývat a mohou se plně věnovat obsahu. Stávající aplikaci tedy stačí upravit a rozšířit o vlastní třídy a obsah.[32]



Obrázek 9 - Výběr platformy a typu aplikace

Prostředí Xcodu je tvořeno oknem projektu. To obsahuje všechny nástroje, které se používají k tvorbě aplikací. Jedná se v podstatě o velmi pokročilý textový editor. Skládá se ze čtyř hlavních částí: *Groups & File lists*, *Detail View*, *Editor* a *Toolbar* (viz Obrázek 10).

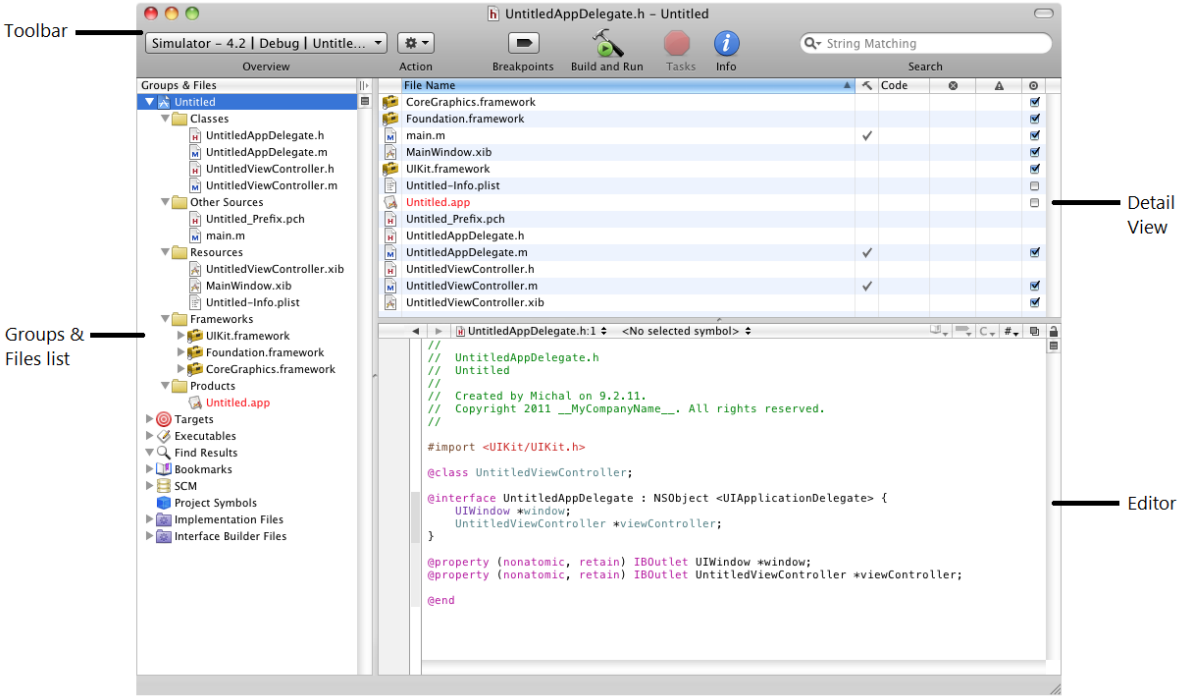
Toolbar obsahuje hlavní ovládací prvky, které jsou nezbytně nutné k práci s projektem, například výběr verze SDK a kompilátor. Toto je důležité při optimalizaci chování aplikací pro starší verze iPhonu, protože nové verze iOS SDK se přizpůsobují poslednímu modelu iPhonu. Některé funkce nemusí být ve starších verzích iPhonu dostupné a může tak dojít k nekompatibilitě aplikace. Tlačítko *Build and Run* slouží ke kompilaci a spuštění aplikace na iOS Simulatoru (viz kapitola 3.4.4.4). Tlačítko

[32] PIPER I., *Learn Xcode Tools for Mac OS X and iPhone Development*, kap. 2

Breakpoints aktivuje breakpointy v projektu a změní tlačítko *Build and Run* na *Build and Debug* a Xcode se přepne do módu Debugger (viz kapitola 3.4.4.3). Stiskem tlačítka *Task* se zastaví všechny aktuálně probíhající operace.[33]

Groups & Files list seskupuje všechny komponenty projektu. Mezi nejdůležitější skupiny patří *Resources* seskupující zdroje (části uživatelského prostředí), *Classes* obsahující zdrojové soubory **.m* a hlavičkové soubory **.h* a skupina *Frameworks* seskupující frameworky použité v projektu.

Podrobný obsah složek se zobrazí v části *Detail View* a zdrojové soubory pak lze snadno upravovat pomocí *Editoru*. *Editor* podporuje prediktivní psaní zdrojového kódu a jeho jednotlivé části je schopen barevně odlišit. Díky tomu je velmi přehledný.[34]



Obrázek 10 - Okno projektu

3.4.4.2 Interface Builder

Interface Builder je vizuální aplikace pro tvorbu uživatelského prostředí pro systémy iOS a Mac OS X. Obsahuje knihovny s jednotlivými grafickými prvky, které lze jednoduše poskládat, uspořádat a vytvořit tak uživatelské prostředí aplikace. Dále lze prvky propojovat a nastavovat jejich vlastnosti. Výsledkem je soubor s příponou **.xib*, který je

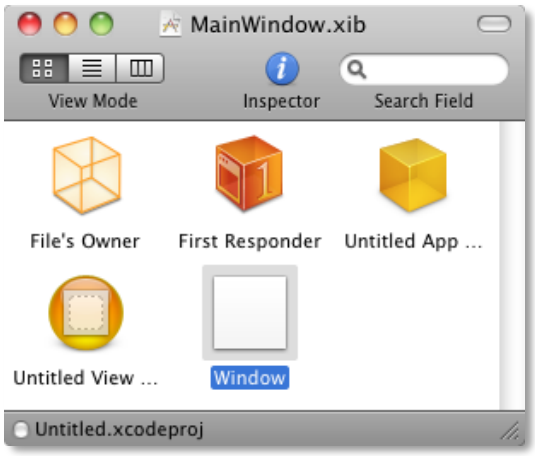
[33] Apple Inc., *Tour of Xcode*
[34] Apple Inc., *Tour of Xcode*

součástí každého projektu. Lze ho najít ve složce *Resources* v části *Groups & Files list* (viz kapitola 3.4.4.1).

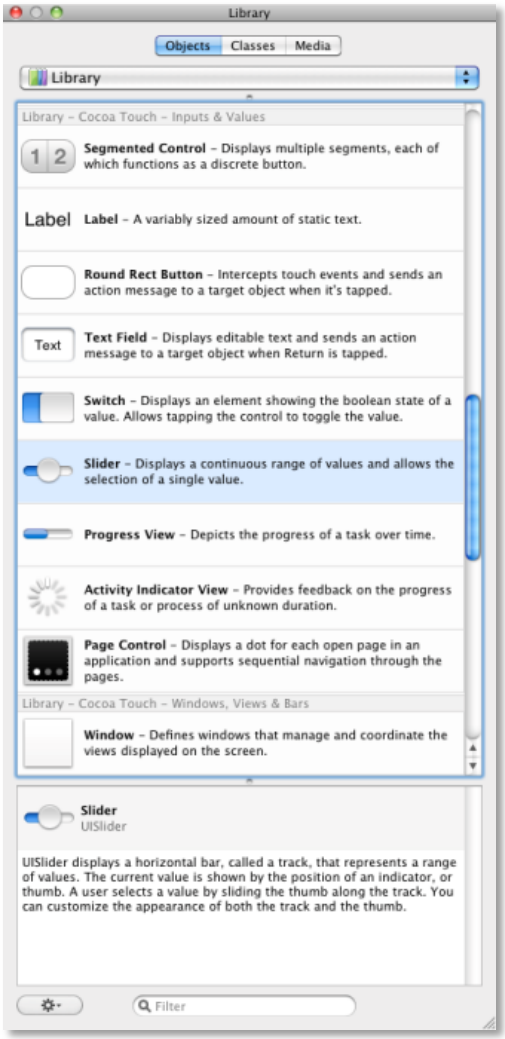
Interface Builder se skládá ze tří hlavních částí (oken). Jsou to *Document Window*, *Library* a *Inspector*.[35]

Document Window (viz Obrázek 11) shromažďuje objekty, které reprezentují jednotlivé části projektu a lze je rozdělit na dvě skupiny. První skupinou jsou *Interface objects.* Tyto objekty obsahují jednotlivé grafické prvky, například okna, ovládací prvky, menu a další. Druhou skupinou jsou *Placeholder objects.* Ty reprezentují již napsanou část zdrojového kódu a s pomocí *Connection Panel* (viz níže) umožňují propojovat jednotlivé grafické části se zdrojovým kódem projektu.

Library (viz Obrázek 12) obsahuje samostatné ovládací prvky a další zdroje, pomocí nichž se tvoří uživatelské prostředí. Do této knihovny lze přidávat i vlastní grafické prvky. Vybrané prvky lze do projektu přidat jednoduše přetažením do okna projektu.



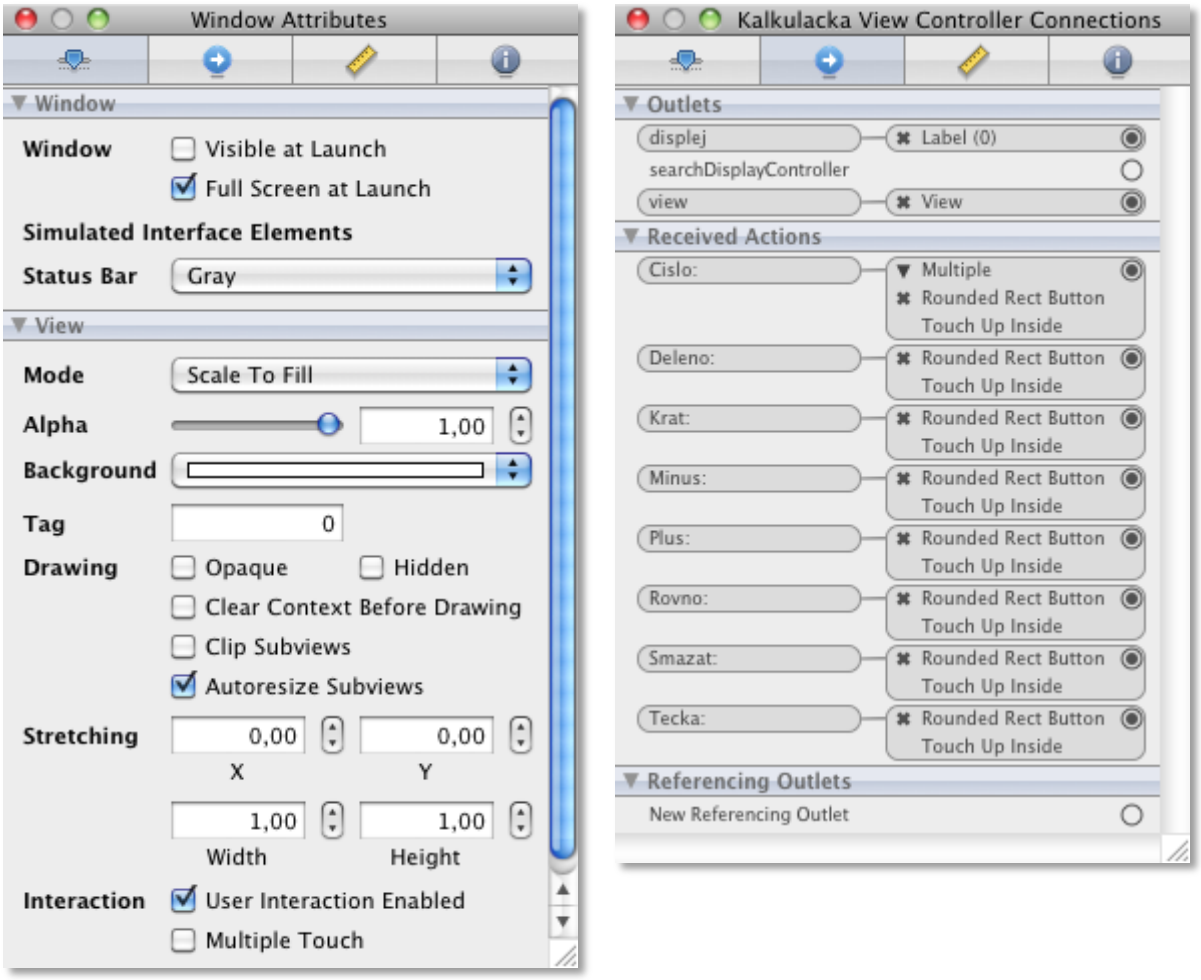
Obrázek 11 - Document Window



Obrázek 12 - Library

[35] Apple Inc., *Interface Builder User Guide*

S pomocí nástroje Inspector (viz Obrázek 13) lze nastavit různé vlastnosti, chování a efekty vybraného grafického prvku nebo části uživatelského prostředí.



Obrázek 13 – Inspector a Connection panel

Connection panel (viz Obrázek 13) je součástí nástroje *Inspector*. Slouží k vytvoření vazeb mezi jednotlivými ovládacími prvky a zdrojovým kódem. Okno *Connection panel* obsahuje *Outlets*, což je součást zdrojového kódu a slouží k identifikaci proměnných, které bude Interface Builder využívat. Pro identifikaci ve zdrojovém kódu se používá *IBOutlet*. Dále obsahuje *Received Actions* a *Sent Actions*. *Received Actions* je zpráva, kterou objekt může vykonávat, protože vlastní příslušnou metodu. *Sent Actions* je zpráva, kterou objekt pošle cílovému objektu, který danou zprávu vykoná.[36]

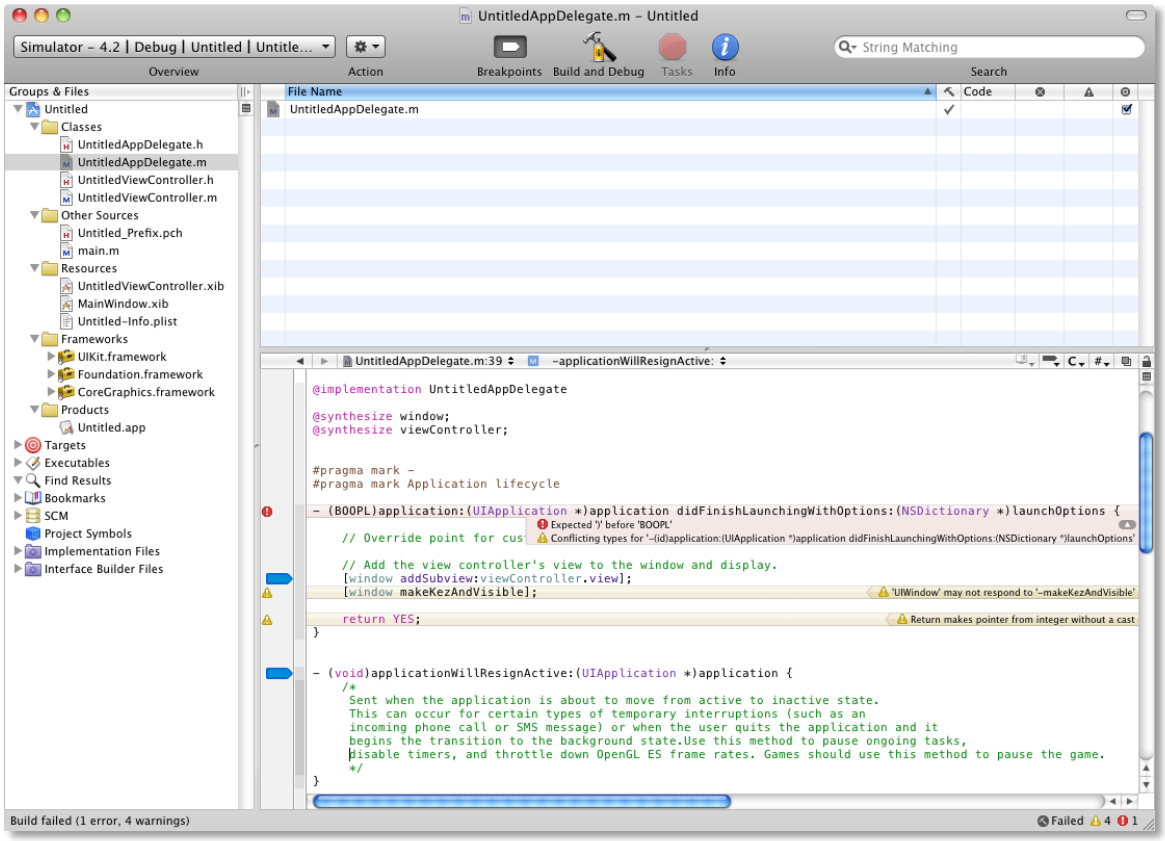
[36] Apple Inc., *Interface Builder User Guide*

3.4.4.3 Debugger

Jedná se o mód Xcodu, který monitoruje aplikaci při chodu a vyhledává chyby. Následně reportuje informaci o chybách a nabízí možná řešení. Debugger vyhledává především chyby v syntaxi, ale i chyby ostatní, např. pointer ukazující na neexistující proměnnou nebo situaci, která by v budoucnu mohla k chybě vést. Chyby syntaxe jsou označeny červeným vykřičníkem. Ostatní chyby a upozornění jsou označeny žlutým vykřičníkem.

Důležitou součástí Debuggeru jsou *Breakpointy*. Pomocí nich lze aplikaci zastavovat při chodu a analyzovat aktuální stav, například sledovat, jakých hodnot nabývá určitá proměnná. *Breakpointy* reprezentují modré šipky (viz Obrázek 14).

Debugger je součástí Xcodu, lze ho ale spustit i v samostatném okně, kde je navíc k dispozici *Console* (log) a podrobnější informace o jednotlivých částech kódu.[37]



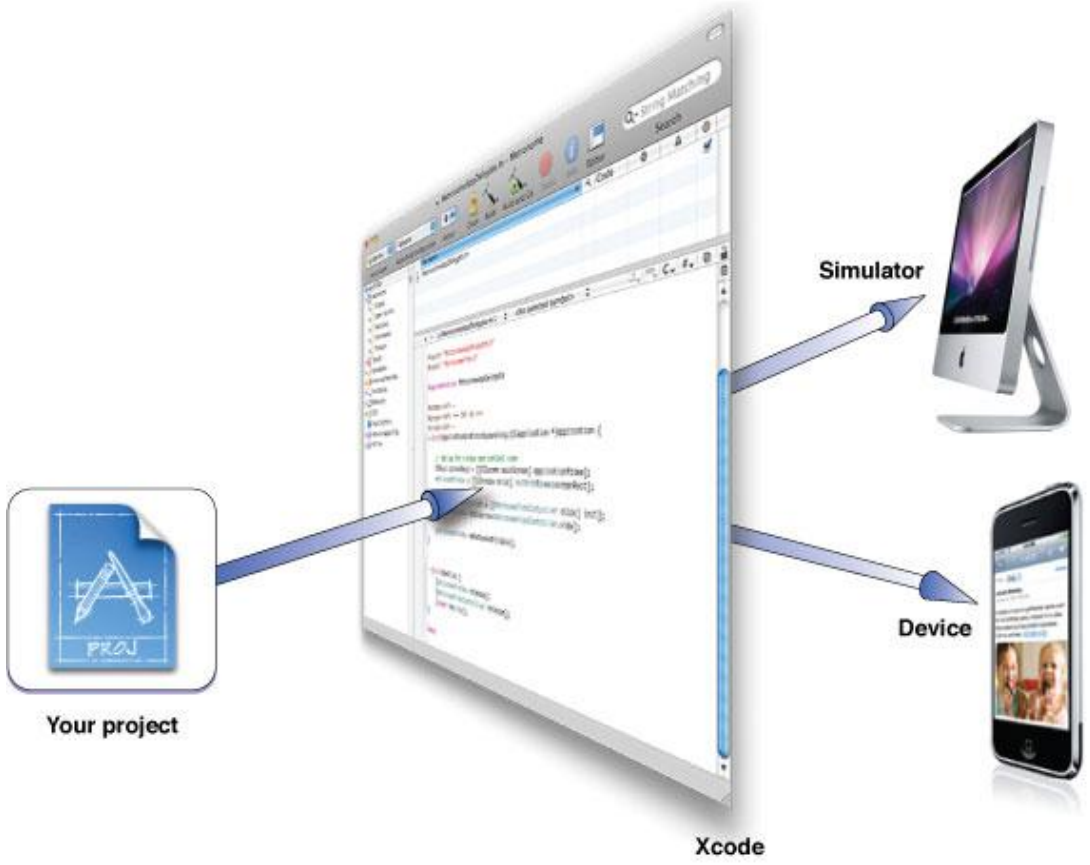
Obrázek 14 - Xcode v módu Debugger

[37] PIPER I., *Learn Xcode Tools for Mac OS X and iPhone Development*, kap. 7

3.4.4.4 iOS Simulator

Aplikace lze po dokončení, ale i během vývoje, testovat (viz Obrázek 15). K testování je možné použít iOS Simulator nebo přímo zařízení, na které je aplikace vyvíjena.

Simulátor emuluje prostředí systému iOS a umožňuje tak zjistit, zda se aplikace chová korektně. Plnohodnotné testování lze provést po připojení zařízení, pro které je aplikace určena a simulovat její běh v reálných podmínkách.[38]



Obrázek 15 - Průběh testování[39]

[38] Apple Inc., *Tools for iOS Development*
[39] Apple Inc., *Tools for iOS Development*

4 Vlastní práce

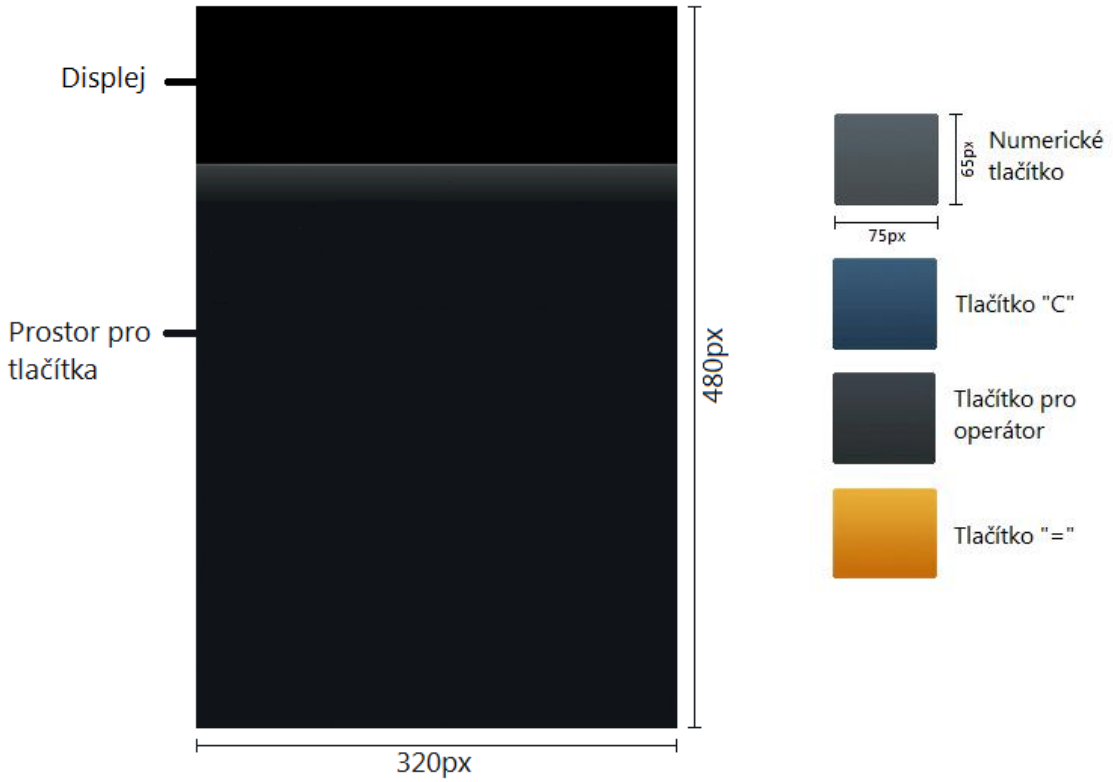
4.1 Analýza aplikace

Součástí této práce je také vytvoření aplikace pro platformu iPhone a ukázka použití jednotlivých nástrojů iOS SDK pro tuto platformu. Předmětem vývoje je aplikace s názvem Kalkulačka.

Jedná se o jednoduchou matematickou kalkulačku, umožňující provádět základní matematické operace – sčítání, odčítání, dělení a násobení. Čísla je možné zadávat s desetinnou čárkou a výsledek je zaokrouhlován s přesností na jedno desetinné místo.

4.1.1 Uživatelské prostředí

Tvorba uživatelského prostředí spočívá ve vytvoření šablony vzhledu kalkulačky. Šablona se skládá z pozadí a tlačítek. Pozadí je rozděleno na dvě části (viz Obrázek 16) - prostor pro tlačítka a část, kde se zobrazují výsledky (displej).



Obrázek 16 - Návrh uživatelského prostředí

4.1.2 Zdrojový kód

Aplikace se skládá ze tří částí. Jednotlivé části jsou tvořeny třídami, které zastupují role modelu MVC (viz kapitola 3.4.3.2).

Roli *View* zastupuje soubor *KalkulackaViewController.xib*, který shromažďuje jednotlivé grafické prvky a vytváří tak vzhled aplikace. Roli *Modelu* zastupuje třída *Kalkulacka*, která reprezentuje jednotlivé matematické operace – sčítání, odčítání, dělení a násobení. A nakonec *KalkulackaViewController*, která zastupuje roli *Controlleru*. Podle návrhového vzoru Target-Action (viz kapitola 3.4.3.1) komunikuje pomocí *IBOutlet* a *IBAction* s *View* a transformuje vložená data, která následně předá ke zpracování *Modelu*. *Model* pak vrátí výsledek zpět *Controlleru* a ten zajistí jeho zobrazení na displeji. Ke třídě *KalkulackaViewController* patří ještě pomocná třída *KalkulackaAppDelegate*, která je zodpovědná za spouštění a ukončování aplikace.

4.2 Vývoj aplikace

V předchozí kapitole byla provedena analýza aplikace a předběžný návrh uživatelského prostředí. V této kapitole jsou popsány základní kroky vývoje aplikace.

4.2.1 Vytvoření nového projektu

Vývoj aplikace byl proveden pomocí nástroje Xcode (viz kapitola 3.4.4.1). Vybráním *Create a new Xcode project* (viz Obrázek 8) na Úvodní obrazovce Xcodu byl zahájen vývoj nové aplikace s názvem Kalkulačka. Tato aplikace je určená pro platformu Apple iPhone a vzhledem k tomu, že využívá jeden statický pohled (view), lze použít šablonu (template) *View-based Application* (viz Obrázek 9). Xcode na základě tohoto výběru vygeneruje zdrojový kód pro tuto šablonu.

4.2.1.1 Úprava třídy KalkulackaViewController

Úprava třídy *KalkulackaViewController* spočívala v doplnění o vlastní kód. Tato třída má roli *Controlleru* podle modelu MVC (viz kapitola 3.4.3.2). Zajišťuje tedy komunikaci mezi *View* a *Modelem*. Musí proto obsahovat metody, které zpracovávají uživatelem vložená data. Konkrétně se jedná o číslice, operátory a speciální symboly, jako je „=“ a desetinná čárka. Třída *KalkulackaViewController* se skládá ze dvou souborů, které jsou součástí skupiny *Classes* v Xcodu v části *Groups & File lists*. Prvním souborem je hlavičkový

soubor *KalkulackaViewController.h*, ve kterém byly deklarovány jednotlivé proměnné a metody. Druhým souborem je implementační soubor *KalkulackaViewController.m*, ve kterém byly implementovány jednotlivé metody, deklarované v hlavičkovém souboru. Oba soubory lze upravovat pomocí *Editoru*.

Jednotlivé metody aplikace Kalkulačka byly navrženy podle teoretického postupu uživatele:

1. Vložení prvního čísla
2. Zmáčknutí operátoru
3. Vložení druhého čísla
4. Zmáčknutí „=“
5. Zobrazení výsledku

Třída *KalkulackaViewController* tedy obsahuje metodu c*islo*:

```
- (IBAction) cislo:(id) sender {
        int cislo = [sender tag];
        [self zpracovatCislo: cislo];}
```

Tato metoda uloží po stisknutí patřičného numerického tlačítka jeho hodnotu do proměnné *cislo* a zavolá metodu *zpracovatCislo*:

```
- (void) zpracovatCislo: (int) cislo {
        [textDispleje appendString: [NSString stringWithFormat: @"%i", cislo]];
        [displej setText: textDispleje];}
```

Vložené číslo se z proměnné *cislo* uloží do proměnné *textDispleje* a zobrazí se na displeji kalkulačky. Tímto způsobem na displeji přibývá každé další vložené číslo, dokud nedojde ke zmáčknutí tlačítka operátoru. V tomto okamžiku je zavolána jedna z metod *plus*, *minus*, *krat* nebo *deleno*:

```
- (IBAction) plus: (id) sender {
   [self zpracovatOperator: '+'];}
- (IBAction) minus: (id) sender {
   [self zpracovatOperator: '-'];}
- (IBAction) krat: (id) sender {
   [self zpracovatOperator: '*'];}
- (IBAction) deleno: (id) sender {
   [self zpracovatOperator: '/'];}
```

Zvolený operátor se uloží do proměnné *novyOperator* a dojde k zavolání metody *zpracovatOperator*:

```
- (void) zpracovatOperator: (char) novyOperator {
        [self ulozitCislo];
        mojeKalkulacka.operator=novyOperator;}
```

Tato metoda zavolá metodu *ulozitCislo* a uloží nový operátor do proměnné *operator*. Metoda *ulozitCislo* je velmi důležitá:

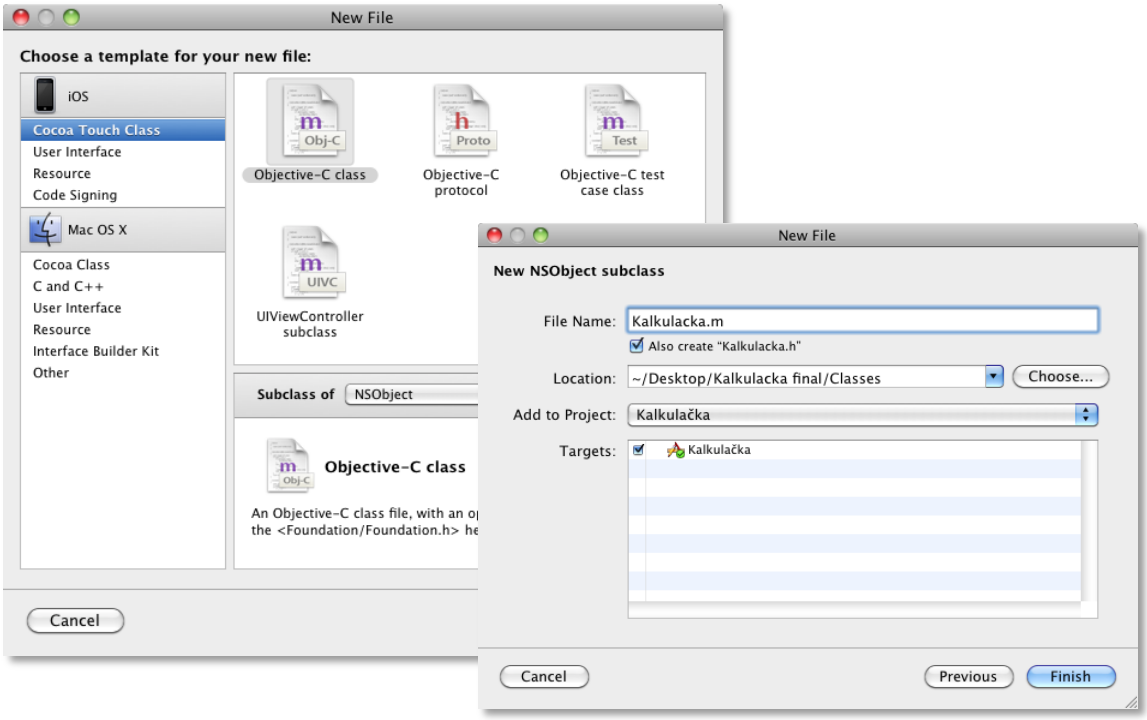
```
-(void) ulozitCislo {
        if (prvniCislo) {
          mojeKalkulacka.cislo1=[textDispleje floatValue];
          [textDispleje setString:[NSString string]];
          prvniCislo=NO;
          dalsiCislo=YES;
        }
        else if (dalsiCislo) {
          mojeKalkulacka.cislo2=[textDispleje floatValue];
          if (mojeKalkulacka.cislo2==0 && mojeKalkulacka.operator=='/') {
                  [displej setText: @"Chyba! Nulou nelze dělit, zmáčkni C"];
          }
          else {
          [textDispleje setString:[NSString string]];
          [mojeKalkulacka provestOperaci];
          [self zobrazitVysledek];
          }
        }
        else if (rovnaSe) {
          rovnaSe=NO;
          dalsiCislo=YES;
        }}
```

Jejím úkolem je uložit čísla z displeje do proměnné. Musí však rozpoznat, zda číslo na displeji je první vložené nebo druhé, a to z důvodu, že se uživatel může rozhodnout pokračovat ve výpočtu a namísto „=“ zmáčkne znovu operátor. Kalkulačka využívá k ukládání čísel dvě proměnné: *cislo1* a *cislo2*. Pokud se tedy uživatel rozhodne pokračovat ve výpočtu, musí se provést výpočet a mezivýsledek uložit do proměnné *cislo1*. Při dalším vložení čísla se postupuje jako při vkládání čísla druhého.

Metoda *ulozitCislo* používá k identifikaci prvního a druhého čísla booleovské proměnné *prvniCislo*, *dalsiCislo* a *rovnaSe*.

4.2.1.2 Vytvoření třídy Kalkulacka

Po vložení druhého čísla a zmáčknutí „=“ nebo operátoru dojde k zavolání metody *provestOperaci*. Tato metoda však není součástí třídy *KalkulackaViewController*, protože



Obrázek 17 - Vytvoření nové třídy

obsahuje interpretaci jednotlivých matematických funkcí a naplňuje tak podstatu role *Modelu*. Nový projekt však neobsahuje třídu, která by roli *Modelu* zastupovala, a proto bylo nutné ji vytvořit. Třída byla nazvána *Kalkulacka*.

Vytvořit novou třídu lze jednoduše pomocí kontextové nabídky *File>New File* v Xcodu (viz Obrázek 17). Nově přidaná třída se objeví ve skupině *Classes* v *Groups & File lists* a bude obsahovat metodu *provestOperaci*:

```
- (float) provestOperaci {
   float vysledek;
        switch (operator) {
          case '+':
                vysledek = cislo1 + cislo2;
                break;
          case '-':
                vysledek = cislo1 - cislo2;
                break;
          case '*':
                vysledek = cislo1 * cislo2;
                break;
          case '/':
                vysledek = cislo1 / cislo2;
                break;
        }
        temp = vysledek;
        return temp;}
```

Tato metoda provádí základní matematické operace – sčítání, odčítání, násobení a dělení. Výsledek operace ukládá do proměnné *temp* a tu vrací zpět metodě *ulozitCislo*. Metoda *ulozitCislo* následně zavolá metodu *zobrazitVysledek*:

```
-(void) zobrazitVysledek {
        [displej setText:[self prevestNaText]];
        mojeKalkulacka.cislo1=mojeKalkulacka.temp;}

- (NSString *) prevestNaText {
          return [NSString stringWithFormat:@"%.1f", mojeKalkulacka.temp];}
```

Zobrazení výsledku probíhá s pomocí metody *prevestNaText*, která převede proměnnou *temp* na text a aktualizuje displej. Následně do proměnné *cislo1* uloží výsledek z proměnné *temp*.

Třída *KalkulackaViewController* také obsahuje metodu *tecka*, která umožňuje vložení desetinné čárky a metodu *Smazat*:

```
- (IBAction) tecka:(id) sender {
        [textDispleje appendString:@"."];
        [displej setText: textDispleje];}

- (IBAction) smazat: (id) sender {
        [mojeKalkulacka vynulovat];
        [textDispleje setString:@""];
        [displej setText: textDispleje];
        prvniCislo=YES;
        dalsiCislo=NO;
        rovnaSe=NO;}
```

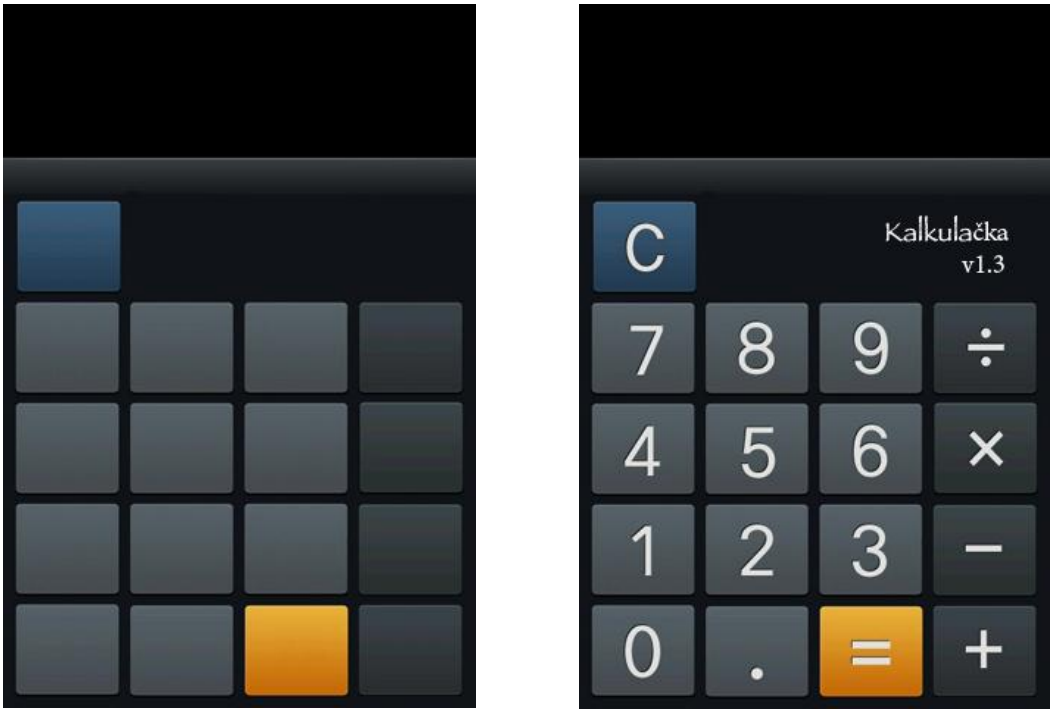
Metoda *smazat* zavolá metodu *vynulovat*, vymaže obsah displeje a nastaví booleovské proměnné do původního stavu.

Metoda *vynulovat* se nachází v třídě *Kalkulacka* a jejím úkolem je vynulovat obsah proměnných *cislo1*, *cislo2* a *temp*.

```
- (void) vynulovat {
        cislo1=0;
        cislo2=0;
        temp=0;}
```

4.2.1.3 Uživatelské prostředí

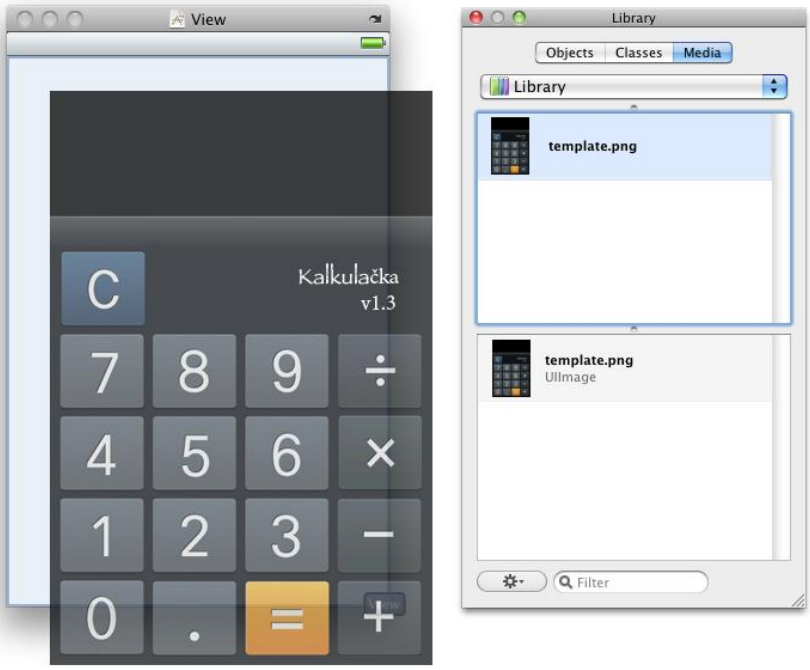
Tvorba uživatelského prostředí byla provedena pomocí nástroje Interface Builder (viz kapitola 3.4.4.2). Samostatný vzhled aplikace lze vytvořit v jakémkoliv grafickém editoru. Vzhledem k tomu, že aplikace Kalkulačka je statická, bylo nejjednodušší vytvořit šablonu, na kterou byla následně umístěna jednotlivá tlačítka. Vzhled vychází z návrhu uvedeném v kapitole 4.1.1. Rozložení tlačítek a následná finální podoba šablony je uvedena na následujícím obrázku (Obrázek 18).



Obrázek 18 - Šablona vzhledu

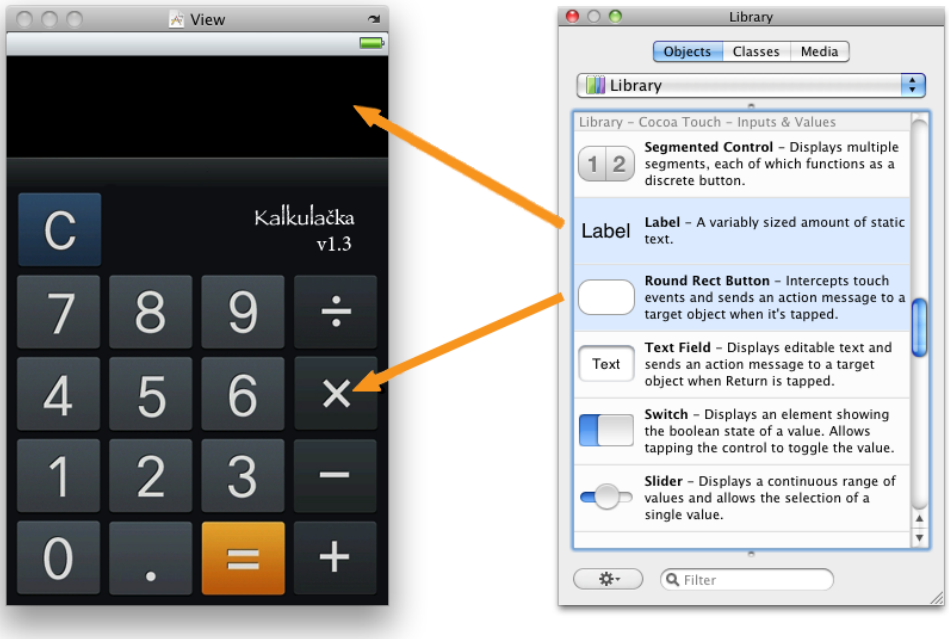
Připravenou šablonu bylo nutné vložit do projektu. Šablonu lze přidat zkopírováním do složky *Resources* v *Groups & File lists* v Xcodu. Po zkopírování je šablona přístupná prostřednictvím *Library* v Interface Builderu.

Dokončením tohoto kroku bylo vše připraveno pro tvorbu uživatelského prostředí. Po spuštění Interface Builderu bylo již k dispozici okno aplikace (view) a to díky vybrané šabloně (template) *View-based Application* při zakládání nového projektu v Xcodu (viz kapitola 4.2.1). Na okno aplikace byla přetažením umístěna vytvořená šablona z *Library*. Stejný postup byl použit i u tlačítek a displeje (viz Obrázek 19 a Obrázek 20).



Obrázek 19 - Tvorba vzhledu I

Pro tlačítka byl zvolen objekt *Round Rect Button* a pro displej objekt typu *Label*. Tlačítka a displej byly jednoduše umístěny na patřičné místo na šabloně. U tlačítek pak bylo nutné nastavit transparentnost, aby nezakryla šablonu a přidat efekt stisknutí.



Obrázek 20 - Tvorba vzhledu II

Výsledkem práce v Interface Builderu byl soubor *KalkulackaViewController.xib*, který zastupuje roli *View* v modelu MVC.

4.2.2 Propojení grafické části a zdrojového kódu

Posledním krokem ve vývoji aplikace je propojení grafické části se zdrojovým kódem. Propojení bylo realizováno v Interface Builderu pomocí *Connection Panelu*, který je součástí nástroje *Inspector* (viz kapitola 3.4.4.2).

Zdrojový kód obsahuje metody, jejichž součástí je tag *IBAction* a proměnné s tagem *IBOutlet*. Pomocí těchto tagů Interface Builder identifikuje části kódu, které mají být propojeny s grafickou částí.

Obrázek 21 - Propojení

Propojení lze vytvořit výběrem objektu, například tlačítka a následným zvolením akce, v tomto případě *Touch Up Inside* (viz Obrázek 21). Propojení bylo provedeno tažením myši na *Placeholder object (File's Owner)* v *Document Window*, který reprezentuje zdrojový kód. Po uvolnění tlačítka myši byla zobrazena nabídka s metodami, které byly identifikovány tagem *IBAction* a vybrána metoda reprezentující zvolené tlačítko. Stejným způsobem bylo vytvořeno propojení u ostatních tlačítek. Výjimkou byl pouze displej, který je identifikován pomocí tagu *IBOutlet*, postup propojení byl však identický jako u tlačítek.

4.2.3 Testování

Po dokončení aplikace bylo provedeno její testování. Pokud by zdrojový kód obsahoval chybu, Debugger by na ni při kompilaci upozornil (viz kapitola 3.4.4.3). Aplikace byla testována pomocí iOS Simulatoru (viz kapitola 3.4.4.4) na aktuální verzi operačního systému iOS (v4.2) (viz Obrázek 22). Simulator byl spuštěn pomocí tlačítka *Build and Run* v Xcodu (viz kapitola 3.4.4.1).

Obrázek 22 - Aplikace Kalkulačka spuštěná v iOS Simulatoru

Po provedeném testování lze konstatovat, že aplikace Kalkulačka fungovala podle předpokladů. Výpočty prováděla korektně a její chování nevykazovalo žádné anomálie.

5 Závěr

Možnost vývoje aplikací je dnes běžnou součástí všech mobilních operačních systémů. Uživatel tak může přizpůsobovat svůj smartphone velmi specifickým potřebám a to především díky společnosti Apple Inc., která s touto možností přišla jako první a odstartovala novou etapu ve využívání smartphonů. Díky tomu došlo k radikální změně ve světě mobilních telefonů a společnost Apple Inc. dodnes drží přední postavení v určování trendů v oblasti mobilních zařízení.

Cílem této práce bylo představení platformy iPhone a možnosti vývoje aplikací pro tuto platformu, která je hlavní součástí výše zmíněné mobilní „revoluce“. Platforma iPhone byla podrobně představena a popsána v první části této práce spolu s vývojovým prostředím Cocoa Touch a nástroji iOS SDK (Xcode, Interface Builder, Debugger a iOS Simulator). Dalším cílem této práce bylo vytvoření ukázkové aplikace. Jednotlivé kroky jejího vývoje byly popsány v druhé části práce. Finálním výsledkem bylo vytvoření plně funkční ukázkové aplikace s názvem Kalkulačka.

6 Seznam použitých zdrojů

6.1 Tištěné dokumenty

BRANNAN, James A. *IPhone SDK Programming : A Beginner's Guide*. New York : McGraw Hill, 2009. 480 s. ISBN 978-0-07-162650-7.

KOCHAN, Stephen G. *Objective-C 2.0 : Výukový kurz programování pro Mac OS X a iPhone*. 1. vydání. Brno : CPRESS, 2010. 550 s. ISBN 978-80-251-2654-7.

MARK, Dave; NUTTING, Jack; LAMARCHE, Jeff. *Beginning iPhone 4 Development : Exploring the iOS SDK*. New York : APRESS, 2011. 657 s. ISBN 978-1-4302-3025-0.

MERUNKA, Vojtěch. *Objektové modelování*. 1. vydání. Praha : Alfa Nakladatelství, 2008. 184 s. ISBN 978-80-87197-04-2.

PIPER, Ian. *Learn Xcode Tools for Mac OS X and iPhone Development*. New York : APRESS, 2010. 450 s. ISBN 978-1-4302-7221-2.

6.2 Elektronické dokumenty

Apple : iPhone [online]. Apple Inc., 2011 [cit. 2011-03-07]. iOS 4 is the world's most advanced mobile operating system. Dostupné z WWW: <http://www.apple.com/iphone/ios4/>.

bada : All about bada [online]. Samsung Electronics Co., 2011 [cit. 2011-03-07]. What is bada. Dostupné z WWW: <http://www.bada.com/>.

bada Developers [online]. Samsung Electronics Co., 2011 [cit. 2011-03-07]. Getting Started with the SDK. Dostupné z WWW: <http://developer.bada.com/apis/docs/commonpage.do?menu=MC01040000&mtb1=&mtb2=>.

HAVRYLUK, Michal. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2011-09-03 [cit. 2011-03-08]. Mobilní OS pro experty: Vzestup a strmý pád Windows Mobile. Dostupné z WWW: <http://mobil.idnes.cz/aplikace.asp?c=A100811_180849_chytre-telefony_ham>.

HAVRYLUK, Michal. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2010-11-01 [cit. 2011-03-07]. Mobilní OS pro experty: Skvělý nástup a slibná budoucnost Android OS. Dostupné z WWW: <http://mobil.idnes.cz/aplikace.asp?c=A100910_202521_pda_ham>.

HAVRYLUK, Michal. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2010-06-25 [cit. 2011-03-07]. Mobilní OS pro experty: nahlédněte pod pokličku úspěšnému Symbianu. Dostupné z WWW: <http://mobil.idnes.cz/mobilni-os-pro-experty-nahlednete-pod-poklicku uspesnemu-symbianu-1fo-/telefony.asp?c=A100624_142926_chytre-telefony_ham>.

HAVRYLUK, Michal. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2010-07-23 [cit. 2011-03-07]. Mobilní OS pro experty: zatracovaný a revoluční iOS pro iPhone. Dostupné z WWW: <http://mobil.idnes.cz/aplikace.asp?c=A100720_203231_chytre-telefony_ham>.

iOS *Reference Library* [online]. Apple Inc., 2010, 2010-07-08 [cit. 2011-03-07]. iOS Overview. Dostupné z WWW: <http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/URL_iPhone_OS_Overview/index.html>.

iOS Reference Library : Cocoa Application Competencies for iOS [online]. Apple Inc., 2010, 2010-07-07 [cit. 2011-03-07]. Target-Action. Dostupné z WWW: <http://developer.apple.com/ library/ios/#documentation/general/conceptual/Devpedia-CocoaApp/TargetAction.html>.

iOS Reference Library : Cocoa Core Competencies [online]. Apple Inc., 2010, 2010-08-03 [cit. 2011-03-07]. Model-View-Controller. Dostupné z WWW: <http://developer.apple.com/library/ios/#documentation/general/conceptual/DevPedia-CocoaCore/MVC.html>.

iOS Reference Library [online]. Apple Inc., 2009, 2009-07-22 [cit. 2011-03-07]. A Tour of Xcode. Dostupné z WWW: <http://developer.apple.com/library/ios/#documentation/DeveloperTools/Conceptual/A_Tour_of_Xcode/000-Introduction/qt_intro.html>.

iOS Reference Library [online]. Apple Inc., 2010, 2010-07-08 [cit. 2011-03-07]. Tools for iOS Development. Dostupné z WWW: <http://developer.apple.com/library/ios/#reference library/GettingStarted/URL_Tools_for_iPhone_OS_Development/index.html>.

LITCHFIELD, Steve. *All About Symbian* [online]. 2010-07-16 [cit. 2011-03-07]. Definiting the Smartphone. Dostupné z WWW: <http://www.allaboutsymbian.com/features/item/Defining_the_Smartphone.php>.

Mac OS X Reference Library [online]. Apple Inc., 2010, 2010-07-12 [cit. 2011-03-07]. Interface Builder User Guide. Dostupné z WWW: <http://developer.apple.com/library/mac/#documentation/DeveloperTools/Conceptual/IB_UserGuide/Introduction/Introduction.html>.

Market share for browsers, operating systems and search engines [online]. 2011 [cit. 2011-03-07]. Operating System Market Share. Dostupné z WWW: <http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8>.

MIKUDÍK, Radek. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2011-03-02 [cit. 2011-03-07]. Google Android čelí dalším útokům špiónských programů. Dostupné z WWW: <http://mobil.idnes.cz/google-android-celi-dalsim-utokum-spionskych-programu-p16-/mob_tech.asp?c=A110301_ 210003_mob_tech_ram>.

About Motorola : History [online]. USA : Motorola Mobility, Inc., 2011 [cit. 2011-03-07]. Making History: Developing the Portable Cellular System. Dostupné z WWW: <http://www.motorola.com/Consumers/US-EN/About_Motorola/History/Explore_Motorola_Heritage/Cell_Phone_Development >.

NOVÁK, Adam. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2007-07-24 [cit. 2011-03-07]. Pozor, revoluce právě začala - velký test Apple iPhone. Dostupné z WWW: <http://mobil.idnes.cz/pozor-revoluce-prave-zacala-velky-test-apple-iphone-fy6-/telefony.asp?c=A070720_225811_ telefony_ada>.

NOVÁK, Adam. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2008-08-23 [cit. 2011-03-07]. Exkluzivně: první česká recenze Apple iPhone 3G. Dostupné z WWW: <http://mobil.idnes.cz/exkluzivne-prvni-ceska-recenze-apple-iphone-3g-fq1-/iphone.asp?c=A080722_141704_iphone_ada>.

NOVÁK, Adam. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2009-06-22 [cit. 2011-03-07]. Recenze nového iPhone 3GS: koupi si pořádně rozmyslete. Dostupné z WWW: <http://mobil.idnes.cz/recenze-noveho-iphone-3gs-koupi-si-poradne-rozmyslete-pf4-/iphone.asp?c=A090621_123025_ iphone_ada>.

NOVÁK, Adam. *iDnes : mobil.cz* [online]. Praha : Mafra a.s., 2010-07-01 [cit. 2011-03-07]. Opravdu první česká recenze iPhone 4, i přes vysokou cenu ho chce skoro každý. Dostupné z WWW: <http://mobil.idnes.cz/opravdu-prvni-ceska-recenze-iphone-4-i-pres-vysokou-cenu-ho-chce-skoro-kazdy-1b0-/iphone.asp?c=A100630_144518_iphone_ada>.

Techtopia [online]. Techopedia.com, 2010, 2010-05-27 [cit. 2011-03-07]. The History of Objective-C. Dostupné z WWW: <http://www.techotopia.com/index.php/The_History_of_Objective-C>.

7 Přílohy

7.1 Seznam obrázků

7.2 Seznam tabulek

7.3 Zdrojový kód

KalkulackaAppDelegate.h

```
#import <UIKit/UIKit.h>
@class KalkulackaViewController;
@interface KalkulackaAppDelegate : NSObject <UIApplicationDelegate> {
    IBOutlet UIWindow *window;
    IBOutlet KalkulackaViewController *viewController;}
@property (nonatomic, retain) IBOutlet UIWindow *window;
@property (nonatomic, retain) IBOutlet KalkulackaViewController *viewController;
@end
```

KalkulackaAppDelegate.m

```
#import "KalkulackaAppDelegate.h"
#import "KalkulackaViewController.h"
@implementation KalkulackaAppDelegate
@synthesize window;
@synthesize viewController;
- (void)applicationDidFinishLaunching:(UIApplication *)application {
    [window addSubview:viewController.view];
    [window makeKeyAndVisible];}
- (void)dealloc {
    [viewController release];
    [window release];
    [super dealloc];}
@end
```

KalkulackaViewController.h

```
#import <UIKit/UIKit.h>
#import "Kalkulacka.h"
@interface KalkulackaViewController : UIViewController {
    UILabel             *displej;
    NSMutableString     *textDispleje;
    Kalkulacka          *mojeKalkulacka;
    BOOL                prvniCislo, dalsiCislo, rovnaSe;}
@property (nonatomic, retain) IBOutlet UILabel *displej;
@property (nonatomic, retain) NSMutableString *textDispleje;
- (IBAction) cislo: (id) sender;
- (IBAction) plus: (id) sender;
- (IBAction) minus: (id) sender;
- (IBAction) krat: (id) sender;
- (IBAction) deleno: (id) sender;
- (IBAction) rovno: (id) sender;
- (IBAction) smazat: (id) sender;
- (IBAction) tecka:(id) sender;
- (void) zobrazitVysledek;
- (void) ulozitCislo;
- (void) zpracovatCislo: (int) cislo;
- (void) zpracovatOperator: (char) novyOperator;
@end
```

KalkulackaViewController.m

```
#import "KalkulackaViewController.h"
@implementation KalkulackaViewController
@synthesize displej, textDispleje;
- (void) viewDidLoad {
        prvniCislo=YES;
        dalsiCislo=NO;
        rovnaSe=NO;
        self.textDispleje = [NSMutableString stringWithCapacity:40];
        mojeKalkulacka = [[Kalkulacka alloc] init];}
- (void) zpracovatCislo: (int) cislo {
        [textDispleje appendString: [NSString stringWithFormat: @"%i", cislo]];
        [displej setText: textDispleje];}
- (void) zpracovatOperator: (char) novyOperator {
        [self ulozitCislo];
        mojeKalkulacka.operator=novyOperator;}
- (NSString *) prevestNaText {
        return [NSString stringWithFormat:@"%.1f", mojeKalkulacka.temp];}
- (IBAction) smazat: (id) sender {
    [mojeKalkulacka vynulovat];
    [textDispleje setString:@""];
    [displej setText: textDispleje];
        prvniCislo=YES;
        dalsiCislo=NO;
        rovnaSe=NO;}
-(void) ulozitCislo {
        if (prvniCislo) {
          mojeKalkulacka.cislo1=[textDispleje floatValue];
          [textDispleje setString:[NSString string]];
          prvniCislo=NO;
          dalsiCislo=YES;}
        else if (dalsiCislo) {
          mojeKalkulacka.cislo2=[textDispleje floatValue];
          if (mojeKalkulacka.cislo2==0 && mojeKalkulacka.operator=='/') {
                  [displej setText: @"Chyba! Nulou nelze dělit, zmáčkni C"];}
          else {
          [textDispleje setString:[NSString string]];
          [mojeKalkulacka provestOperaci];
          [self zobrazitVysledek];}}
        else if (rovnaSe) {
          rovnaSe=NO;
          dalsiCislo=YES;}}
```

```
-(void) zobrazitVysledek {
        [displej setText:[self prevestNaText]];
        mojeKalkulacka.cislo1=mojeKalkulacka.temp;}
- (IBAction) tecka:(id) sender {
        [textDispleje appendString:@"."];
        [displej setText: textDispleje];}
- (IBAction) cislo:(id) sender {
    int cislo = [sender tag];
    [self zpracovatCislo:cislo];}
- (IBAction) plus: (id) sender {
    [self zpracovatOperator: '+'];}
- (IBAction) minus: (id) sender {
    [self zpracovatOperator: '-'];}
- (IBAction) krat: (id) sender {
    [self zpracovatOperator: '*'];}
- (IBAction) deleno: (id) sender {
    [self zpracovatOperator: '/'];}
- (IBAction) rovno: (id) sender {
        [self ulozitCislo];
        dalsiCislo=NO;
        rovnaSe=YES;}
- (void)dealloc {
    [self release];
        [mojeKalkulacka release];
        [super dealloc];}
- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];}
- (void)viewDidUnload {}
@end
```

Kalkulacka.h

```
#import <UIKit/UIKit.h>
@interface Kalkulacka : NSObject {
        char      operator;
        float     temp, cislo1, cislo2;}
@property (nonatomic) float temp, cislo1, cislo2;
@property (nonatomic) char operator;
- (float) provestOperaci;
- (void) vynulovat;
@end
```

Kalkulacka.m

```
#import "Kalkulacka.h"
@implementation Kalkulacka
@synthesize temp, cislo1, cislo2, operator;
- (void) vynulovat {
        cislo1=0;
        cislo2=0;
        temp=0;}
- (float) provestOperaci {
    float vysledek;
        switch (operator) {
          case '+':
              vysledek = cislo1 + cislo2;
              break;
          case '-':
              vysledek = cislo1 - cislo2;
              break;
          case '*':
              vysledek = cislo1 * cislo2;
              break;
          case '/':
              vysledek = cislo1 / cislo2;
              break;}
        temp = vysledek;
        return temp;}
@end
```