Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

BAKALÁŘSKÁ PRÁCE

Tvorba mobilních aplikací v Objective-C pro iOS

Mobile application development for iOS in Objective-C

Vypracoval: Jan Štefančik

Vedoucí bakalářské práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2015

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 16. dubna 2015

Poděkování

Rád bych poděkoval panu PaedDr. Petru Pexovi, Ph.D. za ochotu, vstřícnost a spolupráci při vedení mé bakalářské práce, za odborné rady a připomínky. Dále bych rád poděkoval za čas, který mi věnoval a kvalitní komunikaci.

Abstrakt

Práce se zabývá vývojem nativních aplikací pro platformu iOS. Cílem bude představit systém iOS vývojářům, kteří chtějí zjistit specifika a možnosti tvorby nativních aplikací pro tento mobilní operační systém firmy Apple a prověřit tak možnosti vývoje pro mobilní zařízení. V práci bude popsána architektura operačního systému iOS a na několika dílčích příkladech otestovány některé z dostupných frameworků. V bakalářské práci bude dále podrobně zpracován samotný vývoj aplikací, bude popsáno vše od získání a instalace potřebného programového vybavení pro práci v Objective-C, přes orientaci ve vývojovém prostředí Xcode až po publikování hotové aplikace na App Store.

Klíčová slova

iOS, Xcode, App Store, Objective-C, mobilní aplikace

Abstract

The thesis deals with the development of native applications for iOS platform. The aim is to introduce iOS to developers who want to find out the specifics and possibilities of creating native applications for the mobile operating system from Apple and examine the possibilities of development for mobile devices. This thesis will describe architecture of the operating system iOS and some of the available frameworks will be tested on some easy examples. In the thesis will be explained everything from how to obtain and install the necessary software. Orientation in Xcode, application development in Objective-C and publishing final application on App Store.

Keywords

iOS, Xcode, App Store, Objective-C, mobile application

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan ŠTEFANČIK**
Osobní číslo:	**P11071**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Tvorba mobilních aplikací v Objective-C pro iOS**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce bude představit systém iOS vývojářům, kteří chtějí zjistit specifika a možnosti tvorby nativních aplikací pro tento mobilní operační systém firmy Apple a prověřit tak možnosti vývoje pro mobilní zařízení typu iPhone, iPad. V práci bude popsána architektura operačního systému iOS a na několika dílčích příkladech otestovány některé z dostupných frameworků. V bakalářské práci bude dále podrobně zpracován samotný vývoj aplikací, bude popsáno vše od získání a instalace potřebného programového vybavení pro práci v Objective-C, přes orientaci ve vývojovém prostředí Xcode až po publikování hotové aplikace na AppStore. Hlavním výsledkem bakalářské práce bude plnohodnotná nativní aplikace pro katedru informatiky PF JU v Českých Budějovicích, která bude v práci také podrobně popsána. Aplikace bude dostupná zdarma na aplikačním portálu App Store a data budou automaticky aktualizována z webového portálu KIN PF JU.

Rozsah grafických prací:	**CD ROM**
Rozsah pracovní zprávy:	**40**
Forma zpracování bakalářské práce:	**tištěná**
Seznam odborné literatury:	**viz příloha**

Vedoucí bakalářské práce:	**PaedDr. Petr Pexa, Ph.D.** Katedra informatiky
Datum zadání bakalářské práce:	**9. prosince 2013**
Termín odevzdání bakalářské práce:	**30. dubna 2014**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 25. listopadu 2013

Příloha zadání bakalářské práce

Seznam odborné literatury:

1. MARK, Dave a Jeff LAMARCHE. IPhone SDK: průvodce vývojem aplikací pro iPhone a iPod touch. Vyd. 1. Brno: Computer Press, 2010, 480 s. ISBN 978-80-251-2820-6.
2. KOCHAN, Stephen G. Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone. Vyd. 1. Brno: Computer Press, 2010, 550 s. ISBN 978-80-251-2654-7.
3. Nahavandipoor, Vandad. IOS 7 Programming Cookbook. 2nd. ed. Sebastopol: O'Reilly Media, Inc, USA, 2013. Print.
4. Kochan, Stephen G. Programming in Objective-C. S.l.: Addison-Wesley, 2013. Print.
5. Daniel, Steven F.. Xcode 4 iOS development beginner's guide : use the powerful Xcode 4 suite of tools to build applications for the iPhone and iPad from scratch. Birmingham, U.K.: Packt Pub., 2011. Print.
6. IOS 7 SDK: Working with Background Fetch. [online]. [cit. 2013-11-27]. Dostupné z: http://mobile.tutsplus.com/tutorials/iphone/ios-7-sdk-working-with-background-fetch/
7. Programování pro iOS - 45. Práce v Xcode 4. [online]. [cit. 2013-11-27]. Dostupné z: http://www.mujmac.cz/rubriky/informace/programovani-pro-ios-45-prace-v-xcode-4-58964cz
8. Getting Started with iPhone and iOS Development. [online]. [cit. 2013-11-27]. Dostupné z: http://
9. www.codeproject.com/Articles/88929/Getting-Started-with-iPhone-and-iOS-Development
10. Local and Push Notification Programming Guide. [online]. [cit. 2013-11-27]. Dostupné z: https: developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificat ionsPG/Introduction.html

Obsah

1 Úvod

1.1 Cíle práce

Cílem bakalářské práce bude představit systém iOS vývojářům, kteří chtějí zjistit specifika a možnosti tvorby nativních aplikací pro tento mobilní operační systém firmy Apple a prověřit tak možnosti vývoje pro mobilní zařízení s operačním systémem iOS.

V práci bude popsána architektura operačního systému iOS a na několika dílčích příkladech otestovány některé z dostupných frameworků. V práci bude dále podrobně zpracován samotný vývoj aplikací, bude popsáno vše od získání a instalace potřebného programového vybavení pro práci v Objective-C, přes orientaci ve vývojovém prostředí Xcode až po publikování hotové aplikace na AppStore.

Hlavním cílem bakalářské práce bude plnohodnotná nativní aplikace pro katedru informatiky PF JU v Českých Budějovicích, která bude v práci také podrobně popsána. Aplikace bude dostupná ke stažení zdarma na aplikačním portálu App Store a data budou automaticky aktualizována z webového portálu KIN PF JU.

1.2 Východiska práce

Podíl chytrých telefonů na celkovém počtu mobilních telefonů v populaci narůstá. Uživatelé chytrých telefonů si vedle dosavadního ovládání různých internetových aplikací skrze internetový prohlížeč zvykli používat nativní mobilní aplikace, které jsou pro ně mnohem pohodlnější, než k nim přistupovat přes webový prohlížeč. Tyto aplikace jsou rychlé, spolehlivé a umí využívat hardwarových schopností telefonu.
V práci naleznete aplikaci pro Katedru informatiky, která bude sloužit jako rozcestník pro pracovníky katedry, stávající studenty a nové uchazeče o studium. Aplikace bude nativní, takže uživatelé již nebudou muset přistupovat na stránky katedry jenom přes webové stránky, ale budou mít možnost přistupovat přímo přes nativní aplikaci, kterou si mohou zdarma stáhnout do svého telefonu.

1.3 Metodika práce

V úvodu čtenáře seznámím s operačním systémem iOS a jeho historií. Poté představím architekturu iOS, s ní spojené vrstvy v systému a frameworky, které nám

usnadní vývoj aplikací. Čtenář se dále také dozví, jaké požadavky je nutné splnit v případě, že bude pro tuto platformu vyvíjet. V práci ukáži, co vše je k programování a publikování aplikací potřebné. Dále sestavím několik jednoduchých příkladů, kde budou využity již zmiňované frameworky. V neposlední řadě vytvořím nativní aplikaci Katedry informatiky, která bude publikovaná na aplikační portál App Store.

2 Platforma iOS

iOS je mobilním operačním systémem vytvořeným společností Apple, který byl původně určen pouze pro mobilní telefony iPhone, ale příchodem dalších mobilních zařízení, konkrétně iPodem Touch, iPadem a chytrými hodinkami Apple Watch, Apple použil tento mobilní operační systém i pro tato zařízení. V současné době můžeme iOS nalézt ještě v jednom zařízení a tím je Apple TV.

2.1 Verze systému iOS

2.1.1 iPhone OS

První verze tohoto operačního systému byla zároveň představená s prvním mobilním telefonem této společnosti, a to 29. června 2007. V té době Apple neměl pro tento operační systém žádné oficiální označení. První iPhone s tímto operačním systémem obsahoval základní aplikace a nepodporoval aplikace třetích stran. Jediná možnost, jak vyvíjet pro tuto platformu, byla tvořit webové aplikace, které se chovaly jako nativní, a spouštět je v internetovém prohlížeči Safari. To se však změnilo v březnu 2008, kdy Apple uvolnil beta verzi svého vývojářského prostředí pro tvorbu aplikací s názvem iOS SDK, a tím umožnil vývojářům vyvíjet nativní aplikace pro tuto platformu. [1][2]

2.1.2 iOS 2

11. července 2008 vydala společnost Apple svůj druhý operační systém pro mobilní zařízení, tentokrát už s označením iOS 2, jehož hlavní novinkou byla aplikace App Store, která umožňovala uživatelům stahovat tisíce různých aplikací, které vývojáři stihli od vydání iOS SDK připravit. Mezi dalšími novinkami byla možnost zobrazení dokumentů vytvořené v MS Office, možnost pořízení snímku obrazovky, ukládání obrázků z webového prohlížeče do aplikace fotky, rodičovský zámek a nově přibyla možnost více ploch, na kterých si uživatelé mohli organizovat aplikace stažené z App Storu. [1][2]

2.1.3 iOS 3

17.června 2009 Apple představil iOS 3, který na rozdíl o předchozí verze nebyl tak převratný, ale přinesl požadované funkce a vylepšení, na která byli uživatelé zvyklí z konkurenčních platforem. Mezi nová vylepšení patřila možnost kopírovat, vyjmout a vložit text. Dále Apple přidal podporu MMS zpráv, možnost natáčet

video pomocí aplikace fotoaparát, funkci osobní hotspot pro sdílení internetového připojení z telefonu a do aplikace Mapy integroval magnetický kompas, aby uživatelé měli ještě lepší přehled, kde se nacházejí a k jaké světové straně jsou se svým zařízením natočeni. [1][2]

2.1.4 iOS 4

21. června 2010 uvedl Apple v pořadí již čtvrtou verzi mobilního operačního systému s označením iOS. Tato aktualizace posunula mobilní zařízení od této společnosti zase o kus dále. Hlavní novinkou byl multitasking, který uživatelům umožňoval přepínat mezi jednotlivými aplikacemi bez toho, aby přišli o svojí práci v jiné aplikaci. Byla přidána nová aplikace FaceTime, která uživatelům této platformy dovolila komunikovat prostřednictvím videohovorů. Dále byla přiidána i služba GameCenter, která hráčům umožnila porovnávat své výsledky v různých hrách a aplikace iBooks, která umožnila čtení PDF souborů a nákup elektronických knih. [1][2]

2.1.5 iOS 5

Rok poté, 6. června 2011, vydal Apple iOS 5, jehož hlavní novinkou byla hlasová asistentka SIRI, která uměla zkontrolovat, přečíst nové emaily, přidat novou událost do kalendáře, převézt nadiktovanou zprávu a následně odeslat, a to vše bez toho, aby se uživatel musel dotknout obrazovky zařízení. Velmi vítanou funkcí byly iMessages, které dodnes umožňují posílat zprávy ať už textové, nebo obrázkové dalším uživatelům této platformy,a to pomocí internetového připojení, bez dalších poplatků. Novinkou byla i bezdrátová synchronizace a záloha dat zařízeni s programem iTunes, která udělala z iPhonu téměř nezávislé zařízení na počítači. Poslední důležitou novinkou byla cloudová služba iCloud, která měla za úkol synchronizovat data uživatele napříč všemi jeho zařízeními. [1][2]

2.1.6 iOS 6

Další verze tohoto operačního systému, byla představena 11. června 2012 a nesla označení iOS6. I v této verzi přibyla spousta novinek. Apple se rozhodl ukončit spolupráci se společností Google a aplikace Mapy dostala nové mapové podklady. Byla představená nová aplikace Passbook, do které si uživatelé mohli začít ukládat letenky, různé slevové kupóny a lístky do kina. Od této verze uživatelé již mohli provozovat videohovory přes FaceTime i prostřednictvím mobilní datové sítě, v předchozích verzích byla tato funkce dostupná pouze s WiFi připojením. V této verzi Apple myslel i na motoricky handicapované uživatele a implemento-

val funkci Assistive Touch, díky které si uživatele mohli nastavit ovládací gesta dle svých vlastních potřeb. Aktualizaci dostala i hlasová asistentka SIRI, která od této verze vyhledává restaurace v okolí a zjišťovat výsledky sportovních klání. [1][2]

2.1.7 iOS 7

Předposlední verze operačního systému iOS 7 byla představená 10. června 2013 a prošla největší změnou od jejího vzniku. Design iOS převzal po Scottu Forstallovi vrchní šéfdesigner Applu Jony Ive a designově přepracoval systém od základu. Změny nezůstaly pouze u designu. Přibyla funkce ControlCenter, která umožnila rychlý přístup k zapnutí WiFi, Bluetooth, leteckého režimu, funkci nerušit, ztlumení hlasitosti, fotoaparátu a kalkulačce . Naprostou novinkou systému iOS 7 byla funkce AirDrop, díky níž mohli uživatelé pohodlně sdílet videa a fotografie s ostatními uživateli, kteří jsou v jeho dosahu. Další příjemnou novinkou byla automatická aktualizace aplikací na pozadí zařízení. V případě, že byla k dispozici v App Storu nová aktualizace aplikace a uživatel byl připojen se svým zařízeníkm internetu prostřednictvím WiFi, tak se systém postaral sám o stažení na pozadí. [1][2]

2.1.8 iOS 8

Poslední verzí operačního systému, kterou Apple zatím představil, je verze iOS 8, která byla představena 2. června 2014. Design v této verzi byl zachován z předchozí verze iOS7, co však Apple do této verze přidal je funkce handoff, díky které si uživatelé například mohou rozepsat email v iPhonu, a poté na Macu pokračovat přesně tam, kde skončili. Apple podporuje tuto funkci u většiny svých aplikací a umožnil jí implementovat i vývojářům do svých aplikací. Mezi další novinky patří aplikace Zdraví, která sjednocuje všechny zdravotní data z různých zařízení na jednom místě. Dále iCloud Drive, kde je nyní možné přistupovat k dokumentům a datům ostatních aplikací, a tím zamezit vzniku duplicitních dokumentů v případě, že jej uživatel upravil ve více aplikacích. [1][2]

3 Požadavky pro vývoj Aplikace

Pro vývoj a publikování iOS aplikací je potřeba vlastnit počítač od společnosti Apple, který je osazen procesorem od Intelu, a na kterém běží minimálně operační systém OS X Snow Leopard nebo novější. Dále je zapotřebí vývojové prostředí Xcode a v případě, že budeme chtít aplikaci publikovat, je nutné mít také platný vývojářský účet. [3]

3.1 Hardwarové požadavky

Jak už jsem zmínil v úvodu této kapitoly, pro vývoj aplikací je nutné mít nainstalovaný alespoň operační systém OS X ve verzi Snow Leopard. Tento systém nám dovolí nainstalovat vývojové prostředí Xcode nanejvýše ve verzi 4.3.3. Abychom mohli využívat nejnovější funkce, které Apple do tohoto vývojového prostředí implementoval, je potřeba nainstalovat aktuální verzi tohoto vývojového prostředí, kterou je aktuálně verze 6.2. Tato verze vyžaduje novější operační systém OS X Mavericks, který lze nainstalovat na zařízeních uvedených v tomto seznamu:

- iMac (Mid-2007 nebo novější)
- MacBook (13-inch Aluminum, Late 2008)
- MacBook (13-inch, Early 2009 nebo novější)
- MacBook Pro (13-inch, Mid-2009 nebo novější),
- MacBook Pro (15-inch or 17-inch, Mid/Late 2007 nebo novější)
- MacBook Air (Late 2008 nebo novější)
- Mac mini (Early 2009 nebo novější)
- Mac Pro (Early 2008 nebo novější)
- Xserve (Early 2009)

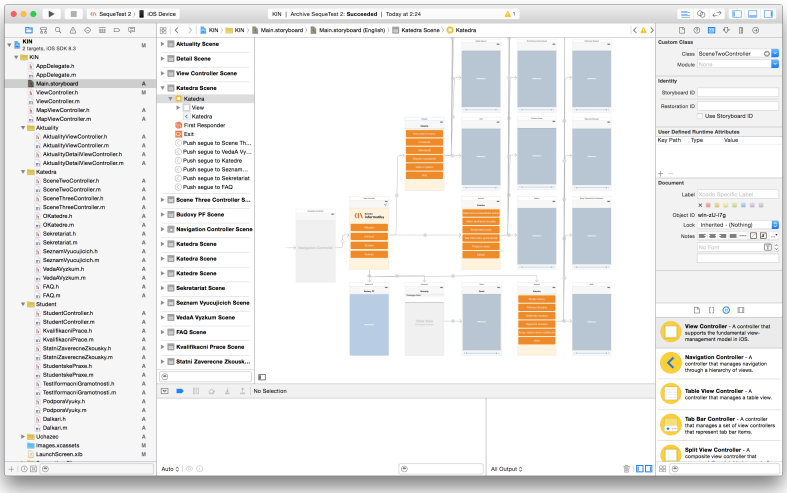
Systém dále vyžaduje minimálně 2 GB operační paměti, a při aktualizaci ze starší verze je vyžadován minimálně OS X Snow Leopard, nebo novější. [4]

3.2 Objective-C

Objective-C je programovací jazyk, který byl navržen počátkem 80. let 20. století. Tento jazyk vychází z jednoho z prvních objektově orientovaných programovacích jazyků SmallTalk-80 a z programovacího jazyka C. Objective-C je další vrstvou jazyka C, což znamená, že do tohoto jazyka byly přidány různé prvky, které umožňují vytvářet objekty a pracovat s nimi. V roce 2007 byla představená aktualizace tohoto jazyka s názvem Objective-C 2.0. [5]

3.3 Xcode

Xcode je integrované vývojové prostředí, které poskytuje vývojářům nástroje potřebné pro vývoj aplikací pro iOS a OS X. Xcode lze získat dvěma způsoby. Prvním z nich je bezplatné stažení přes App Store[1], kde tuto aplikaci stačí vyhledat pod klíčovým slovem Xcode a následně kliknout na volbu stáhnout. V App Storu, nalezne vývojář vždy poslední vydanou verzi tohoto vývojové prostředí. Prostřednictvím App Storu lze v případě, že aplikace dostane aktualizaci, zároveň snadno tuto aplikaci aktualizovat. Druhou variantou, jak Xcode získat, je možnost stažení z portálu pro vývojáře developer.apple.com. V tom případě je nutné mít vývojářskou licenci. Výhodou stažení z tohoto portálu je to, že lze stáhnout i nadcházející verzi této aplikace, která je označená jako beta, a také beta verzi systému iOS pro všechna podporovaná iOS zařízení. Tím pádem lze vyvíjenou aplikaci s předstihem otestovat a upravit tak, aby byla připravená, až Apple tuto beta verzi iOS oficiálně vydá ke stažením všem uživatelům. [6][7]



Obrázek 3.1: Snímek vývojového prostředí Xcode

[1] App Store - platforma pro digitální distribuci aplikací pro iOS a OS X

Project Editor

Naprostým základem Xcode je textový editor. Editor doplňuje za vývojáře zdrojový kód, umožňuje složit část kódu pro lepší přehled ve zdrojovém kódu, upozorňuje na syntaktické a logické chyby. Nabízí také mnoho nastavení, jak si prostředí přizpůsobit pro vlastní potřeby. Vývojáři mají v samostatném okně také přístup k dokumentaci pro iOS a dalších podkladů. [6][7]

Interface Builder

Interface je dalším nástrojem Xcode, který se využívá k prototypování uživatelského rozhraní. S tímto nástrojem lze v jednom souboru navrhnout pomocí připravených komponent všechna okna aplikace. Mezi komponenty patří například: tlačítka, textová pole a přepínače. Po umístění vhodných komponent do jednotlivých oken, je možné mezi nimi definovat vazby, a tak navrhnout, jak se aplikace bude chovat po stisknutí na určitou komponentu. [6][7]

Simulator

Simulátor je součástí Xcode, který lze spustit na OS X a umožňuje vývojářům otestovat svojí aplikaci bez nutnosti připojovat fyzické zařízení k počítači. Simulátor poskytuje stejné uživatelské rozhraní, které běží na skutečných fyzických zařízeních. Vývojář tak může rychle zjistit, jak jeho aplikace bude vypadat na různých zařízeních v případě, že je nemá fyzicky k dispozici. Simulátor má ale určitá omezení. V případě, že budete chtít pořídit do své aplikace fotografii, tak to nedovolí kvůli absenci kamery. Nelze ani otestovat aplikace, které jsou postavené na spolupráci s akcelerometrem. [6][7]

Performance Tools

Dalším nástrojem, který Xcode vývojářům nabízí, je Performance Tools. Tento nástroj umožňuje sbírat data z vyvíjené aplikace a graficky tato data zobrazit v časové ose. Nástroj dokáže zobrazit využití operační paměti, využití procesoru, aktivitě na disku a síťové aktivitě. Tento nástroj je velmi důležitý při ladění, aby se případně mohl zajistit rychlejší běh aplikace. [6][7]

Debugger

Debugger je nástroj pro odladění chyb v programu. Debugger nám umožňuje rychle nalézt a opravit chyby ve zdrojovém kódu. V debbugeru lze procházet zdrojový kód řádek po řádku a zkoumat hodnoty instančních proměnných, dále nastavit breakpointy, popřípadě vypisovat zprávy v konzoli. [6][7]

3.4 Vývojářská licence

V případě, že budete mít funkční aplikaci, kterou byste rádi vyzkoušeli i jinde než v simulátoru, který je integrovaný ve vývojovém prostředí Xcode, třeba ve vašem iPhonu, případně jí nahrát na App Store, nebo získat přístup k více studijním materiálům týkající se vývoje aplikací pro iOS, je nutné si zaregistrovat jednu z níže uvedených vývojářských licencí. [6][8]

iOS Developer Program

Nejčastější volbou pro vývojáře je klasická licence, která je zpoplatněna částkou 99 eur ročně. Po zaregistrování a zaplacení požadované částky můžete zkoušet vytvořené aplikace na fyzickém zařízení. Apple také poskytuje přístup k portálu pro vývojáře, kde naleznete poslední beta verze Xcode a iOS, dále studijní materiály a hlavně možnost publikovat aplikace na App Store. [6][8]

Developer Enterprise Program

Vývojářský účet pro velké vývojářské týmy stojí 299 eur ročně. Tento program je určen pro distribuci in-house aplikací a cílí na velké firmy, které vyvíjí aplikace pro firemní účely. Tento program neumožňuje nahrávat vytvořené aplikace na App Store, kde by si je mohli běžní uživatelé stáhnout do svých zařízení. [6][8]

Developer University Program

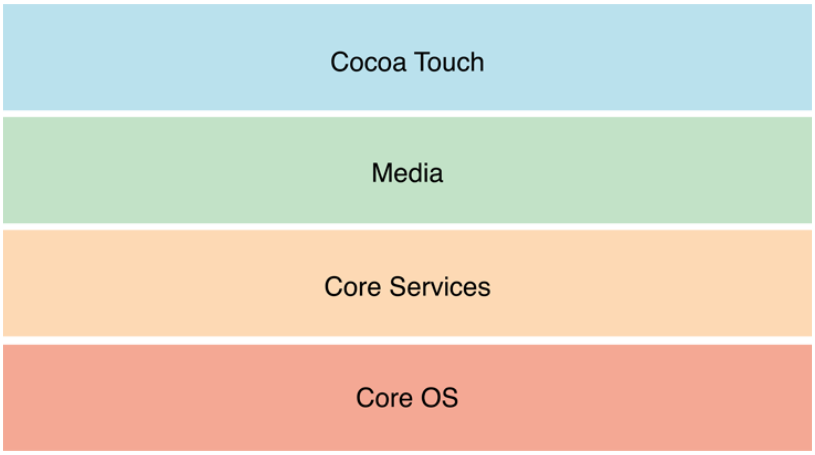
Poslední možností, jak získat vývojářskou licenci, je prostřednictvím univerzitního programu, který je dostupný pouze pro vysoké školy. Tento program je zdarma a má za cíl podpořit univerzity v tom, aby zařadily výuku tvorby mobilních aplikací pro iOS do svých studijních plánů. Program umožňuje vyučujícím vytvořit tým čítající až 200 studentů. Zároveň jim poskytuje softwarové vybavení v podobě vývojového prostředí Xcode, přístup k materiálům na portálu pro vývojáře a možnost své aplikace vyzkoušet na jejich fyzických zařízeních. Studenti mohou své vytvořené aplikace sdílet s ostatními studenty případně vyučujícími. [6][8]

Registrace

Vývojářský účet lze registrovat na stránce `https://developer.apple.com/register/`, kde stačí zvolit jeden z vybraných vývojářských programů a zaplatit prostřednictvím platební karty. V případě, že registrujete klasickou licenci pro jednotlivce, tak registrace zabere 24 - 72 hodin. V mém případě, to bylo necelých 24 hodin a měl jsem k dispozici veškeré výhody, které daná vývojářská licence nabízí. [6][8]

4 Architektura iOS

Systém iOS se skládá ze čtyř základních vrstev, jak je možné vidět na obrázku 4.1, které zajišťují základní funkčnost a poskytují vývojářům vysokoúrovňové služby a frameworky potřebné k vývoji aplikací. [9]



Obrázek 4.2: Vrstvy systému iOS[2]

4.1 Cocoa Touch

Vrstva Cocoa Touch obsahuje klíčové frameworky pro tvorbu iOS aplikací. Tyto frameworky definují vzhled aplikace a poskytují také vysokoúrovňové služby, které lze při tvorbě aplikací využít. Při vývoji aplikací se doporučuje začínat touto vrstvou a nižší používat v případě potřeby.

4.1.1 Vysokoúrovňové služby Cocoa Touch

AirDrop

AirDrop umožňuje uživatelům sdílet fotografie, dokumenty, URL[3] adresy a další druhy dat s jinými zařízením v blízkosti. Tuto funkci je možné snadno implementovat při tvorbě aplikace, a tak umožnit uživatelům sdílet data. [10]

[2]Dostupné z: `https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview`

[3]URL - je zkratka z anglického Uniform Resource Locator a používá se pro přesnou identifikaci dokumentů na internetu

Storyboards

Storyboard je soubor, který se používá pro návrh uživatelského rozhraní na jednom místě. Důležitou součástí storyboardů je možnost definování přechodů z jednoho rámce vzhledu na druhý. Díky těmto přechodům je možné přesně definovat tok a uživatelské rozhraní aplikace. Storyboard soubor je možné editovat v Xcode pomocí nástroje Interface Builder. [10]

Multitasking

Výdrž baterie je důležitým aspektem pro uživatele iOS zařízení, a proto je multitasking navržen tak, aby výdrž co nejvíce prodloužil. Při vyvolání multitasking se aplikace, která právě běží, přesune do pozadí. V případě, že už nemusí nic vykonávat, se aplikace úplně pozastaví. Aplikace zůstává stále v paměti zařízení, ale už nic nevykonává. Aplikace, které potřebují vykonávat určité věci na pozadí, mohou o to systém zažádat. [10]

- Aplikace může zažádat o rozšiřující čas pro vykonání určitého procesu
- Aplikace, která přehrává audio soubory, může například zažádat systém o určitý čas pro poskytování této služby
- Aplikace může stahovat v časových intervalech nový obsah ze sítě

UI State Preservation

Tato vysokoúrovňová služba umožňuje v případě, že je v zařízení zapnuto příliš aplikací a systém nemá dostatek paměti, ukončit některé aplikace běžící na pozadí. V případě, že se aplikace přesune z popředí do pozadí, tak se uloží aktuální stav. Při příštím zapnutí se aplikace načte přesně v tom stavu, v jakém byla při ukončení. A to i přesto, že jí systém sám na pozadí ukončil. [10]

Apple Push Notification Service

Tato služba umožňuje upozorňovat uživatele na nové informace, aniž by musela být daná aplikace právě spuštěna. Uživateli je možné zobrazit krátkou textovou informaci, přehrát zvuk či aktualizovat číselnou značku na ikoně aplikace. Iniciaci odeslání push notifikace musí server vývojáře aplikace, který následně komunikuje se servery společnosti Apple. Ty se pokusí o doručení na mobilní zařízení pomocí jeho unikátního identifikátoru. [10]

Local Notifications

Lokální notifikace doplňují mechanismus push notifikací. Ty nevyžadují žádné připojení k serveru a údaje o notifikacích ukládají lokálně. Aplikace běžící na pozadí, například navigace, může v případě potřeby upozornit uživatele na důležité události, například na blížící se zatáčku, a to ihned. Je také možné naplánovat notifikaci na určitý datum a čas. Taková notifikace je uložena v systému. Aplikace v požadovaný čas nemusí běžet, aby se uživateli dané upozornění zobrazilo. [10]

Gesture Recognizers

Vrstva Cocoa Touch obsahuje nástroj pro systémové rozpoznávání definovaných gest a přiřazení určité akce v aplikaci. Systém je schopný rozpoznat velké množství gest. Například tapnutí, vícenásobné tapnutí, přetahování objektů, švihnutí, sevření a rozevření prstů. Je také možné nadefinovat vlastní gesta, případně připravená gesta modifikovat pro vlastní potřeby. [10]

Standard System View Controllers

Mnoho frameworků použitých v systému používá standardizované komponenty. V rámci zachování konzistentního uživatelského zážitku je proto vhodné používat tyto standardizované komponenty i ve vyvíjených aplikacích. Mezi hlavní komponenty patří: [10]

- Zobrazení a editace kontaktních informací
- Vytváření a editace událostí v kalendáři
- Psaní e-mailu a SMS zpráv
- Otevření souboru
- Výběr obrázku z knihovny fotoaparátu
- Natočení videoklipu

4.1.2 Frameworky dostupné ve vrstvě Cocoa Touch

Adress Book UI

Address Book UI framework poskytuje standardní uživatelské a grafické rozhraní pro vytvoření, editaci a výběr existujícího kontaktu. Framework usnadňuje práci s kontakty a zajišťuje, že aplikace využívá stejné rozhraní, jako ostatní systémové aplikace, a tím zajišťuje konzistenci napříč iOS platformou. [10]

Event Kit UI

EventKit UI framework poskytuje standardní uživatelské a grafické rozhraní pro zobrazení a editaci událostí v kalendáři. [10]

GameKit

GameKit framework implementuje podporu služby Game Center, který užmoňuje uživatelům sdílet herní informace online. Game Center poskytuje podporu následujících funkcí: [10]

- Vytvoření online přezdívky. Hráči využívající Game Center, komunikují s ostatními hráči anonymně prostřednictvím vytvořené přezdívky. Pomocí přezdívky si mohou přidat do svého seznamu přátel nové hráče.
- Žebříčky, kde se zobrazuje dosažené skóre ve hře. Skóre lze porovnávat s hráči z celého světa, i s hráči ze seznamu přátel.
- Vytvoření hry pro více hráčů, kteří jsou přihlášení do služby Game Center. Hráči nemusí být připojení lokálně v jedné síti. Veškerou herní komunikaci zajišťuje Game Center.
- Úspěchy umožňují zaznamenenat výsledky a postup, který hráč v dané hře dosáhl. Je možné vyzvat kamaráda, aby se pokusil vaše dosažené výsledky překonat.

iAd Framework

Framework iAd zajišťuje snadnou implementaci reklamních bannerů do aplikace. Lze nastavit, kde a kdy se má reklama uživateli zobrazit. [10]

MapKit Framework

MapKit framework umožňuje zobrazení mapových podkladů, které lze integrovat do aplikace. Pomocí anotací lze na mapě označit například body zájmů. [10]

Message UI Framework

Message UI framework poskytuje podporu pro vytváření emailů a sms zpráv přímo z aplikace. Lze přednastavit příjemce, předmět, tělo a přílohu zprávy. Uživatel tyto údaje může před odesláním modifikovat. [10]

Notification Center Framework

Tento framework umožňuje tvorbu widgetů, které se zobrazují v notifikačním centru. [10]

PushKit Framework

PushKit framework poskytuje podporu pro aplikace využívácící VoIP technologii. Tento framework nahrazuje minulé API[4] , které udržovalo trvalé připojení a tím docházelo k nešetrnému zacházení s baterií. Tento framework funguje na principu Push notifikací. Notifikace dorazí při příchozím hovoru a k navázání připojení dochází v momentu, kdy uživatel hovor přijme. [10]

UI Kit Framework

UI Kit framework spravuje uživatelského rozhraní aplikace a umožňuje následující funkce: [10]

- Základní správa aplikace
- Podpora pro pohybové a dotykové události
- Multitasking
- Tisk
- Vlastní úprava standardních UIKit prvků
- Podpora pro text a webový obsah
- Podpora pro vyjmutí, kopírování a vložení
- Podpora animací uživatelského rozhraní
- Integrace s ostatními aplikacemi na základě URL schémat
- Podpora zpřístupnění pro zdravotně handicapované uživatele
- Podpora služby Apple Push Notification
- Tvorba PDF
- Podpora vlastního vstupního rozhraní, které se chová jako systémová klávesnice

[4]API - sbírka procedur, funkcí, tříd či protokolů nějaké knihovny

- Podpora sdílení obsahu prostřednictvím emailu, Twitteru, Facebooku a dalších sociálních služeb
- Přístup k vestavěné kameře pokud je fyzicky k dispozi
- Přístup ke knihovně obrázků
- Přístup k informacím o zařízení
- Přístup ke stavu baterie
- Přístup k proximity senzoru[4]
- Přístup k informacím z připojeného náhlavního zařízení.

[5]Proximity senzor - senzor schopný detekovat přítomnost blízkých objektů bez fyzického kontaktu

4.2 Media

Vrstva Media obstarává grafické, audio a video technologie, které vývojáři implementují do svých aplikací. Technologie v této vrstvě umožnují programátorům vytvořit aplikace, které vypadají a zní skvěle.

4.2.1 Audio technologie

Audio technologie platformy iOS spolupracují s hardwarem zařízení, aby poskytly co nejlepší zvukový zážitek pro uživatele. Technologie umožňuje zvuk nahrávat a přehrávat v největší možné kvalitě, dále poskytuje podporu MIDI, a umožňuje tak připojit hudební nástroje, s kterými může aplikace spolupracovat.

V případě, že vyvíjíte aplikaci, která pracuje se zvukem, jsou vám k dispozici následující technologie uvedené v tabulce 4.1. [11]

Technologie	**Popis**
Media Player framework	poskytuje přístup k hudební knihovně uživatele a podporu přehrávání jeho skladeb a seznamů skladeb
AV Foundation	rozhraní pro nahrávání a přehrávání zvuku a videa
OpenAL	technologie, která se využívá při práci s prostorovým zvukem
Core Audio	obsahuje sadu frameworků, které poskytují jednoduché a sofistikované rozhraní pro záznam, přehrávání audia a práci s komunikačním protokolem MIDI

Tabulka 4.1: Audio technologie v iOS

Systém iOS podporuje mnoho standardních formátů. Dále podporuje specifické audio formáty vyvinuté přímo společností Apple. Mezi podporované audio formáty patří: [11]

- AAC
- Apple Lossless (ALAC)
- A-law
- IMA/ADPCM (IMA4)
- Linear PCM
- μ-law
- DVI/Intel IMA ADPCM
- Microsoft GSM 6.10

4.2.2 Grafické technologie

Vysoce kvalitní grafika je důležitou součástí systému iOS. Systém poskytuje celou řadu nástrojů, které umožňují vytvářet vlastní grafiku v aplikacích. Nejjednoduššího a nejefektivnějšího vytváření aplikací je možné dosáhnout používáním vykreslených obrázků, komponent a následně nechat samotné vykreslení na systému.

V některých situacích není toto řešení uskutečnitelné, a je možné použít technologie uvedené v tabulce 4.2. [11]

Technologie	**Popis**
Core Graphics	pracuje s 2D vektory a stará se o vykreslení obrázků
Core Animation	pokročilá podpora animací
Core Image	poskytuje sadu filtrů pro nedestruktivní úpravu obrázků
OpenGL ES and GLKit	hardwarově akcelerované vykreslování 2D a 3D grafiky
Metal	rozhraní pro vývoj graficky náročných aplikací
TextKit and Core Text	rozhraní pro náročnější práci s textem
Image I/O	čtení a zápis většiny obrázkových formátů
Photos Library	přístup k fotografiím a videím v uživatelské knihovně

Tabulka 4.2: Grafické technologie v iOS

4.2.3 Video technologie

Video technologie pro iOS poskytují podporu pro přehrávání, nebo streamování videa prostřednictvím internetu. Zařízením, které disponují kamerou, umožňuje video nahrávat a následně jej přehrát.

V tabulce 4.3 jsou uvedené technologie, které lze při práci s videem využít. [11]

Technologie	**Popis**
UIImagePicker-Controller	umožňuje výběr multimediálních souborů z knihovny uživatele, případně umožňuje nový obsah nahrát pomocí kamery, pokud se v zařízení fyzicky nachází
AVKit	poskytuje jednoduché rozhraní pro přehráváná videa, přehrávání na celou, nebo na část obrazovky a možnost implementovat ovládací prvky pro přehrávání videa
AV Foundation	rozhraní pro nahrávání a přehrávání zvuku a videa
Core Media	poskytuje vlastní, nízko úrovňový přístup pro práci s videem

Tabulka 4.3: Video technologie v iOS

Systém iOS podporuje přehrávání videí s koncovkami .m4v, .mp4, .3gp a .mov při použití následujících kodeků: [11]

- H.264 video, až do 1.5 Mbps, 640 x 480 pixels, 30 snímků za vteřinu, Low-Complexity verze H.264 Baseline Profile with AAC-LC audio až do 160 Kbps, 48 kHz, stereo audio ve formátech .m4v, .mp4, a .mov
- H.264 video, až do 768 Kbps, 320 x 240 pixels, 30 snímků za vteřinu, Baseline Profile až do úrovně 1.3 s AAC-LC audiem až do 160 Kbps, 48 kHz, stereo audio ve formátech .m4v, .mp4, a .mov
- MPEG-4 video, až do 2.5 Mbps, 640 x 480 pixels, 30 snímků za vteřinu, Simple Profile s AAC-LC audiem až do 160 Kbps, 48 kHz, stereo audio ve formátech .m4v, .mp4, a .mov

4.2.4 AirPlay

AirPlay umožňuje aplikacím streamovat audio a video obsah do Apple TV, případně streamovat hudební obsah do reproduktorů od výrobců třetích stran, kteří AirPlay podporují. AirPlay podpora je integrovaná ve frameworcích UIKit, Media Player, AV Foundation a Core Audio. Podporované video, nebo audio soubory přehrávané prostřednictvím jednoho ze zmiňovaných frameworků AirPlay podporují, a tak není třeba nic komplikovaného programovat, protože o funkčnost se stará sám systém. [11]

4.2.5 Frameworky dostupné ve vrstvě Media

Assets Library Framework

Poskytuje přístup k fotografiím a videím, uložených v knihovně aplikace Obrázky, která se nachází ve všech iOS zařízeních. [11]

AV Foundation Framework

Nabízí sadu Objective-C tříd, které umožňují přehrávání, nahrávání a správu video a audio souborů. [11]

Příklad funkcí které tento framework nabízí:

- Správa mediální souborů.
- Editace mediálního obsahu.
- Schopnost nahrávat audio a video.
- Schopnost přehrávat audio a video.

- Správa metadat pro multimediální soubory.
- Přesnou synchronizaci zvukových stop.
- Podpora streamování přes službu AirPlay.

AVKit Framework

Využívá existující objekty frameworku AV Foundation a poskytuje jednoduché rozhraní pro přehrávání videa. Dále umožňuje implementovat ovládací prvky pro přehrávání videa. [11]

Core Audio

Poskytuje nativní podporu pro správu audio souborů. Tento framework umožňuje vytváření, nahrávání, míchání a přehrávání audio souborů. Také nabízí rozhraní pro práci s komunikačním protokolem MIDI. [11]

Core Graphics Framework

Core Graphics framework je určený k vykreslování dvourozměrných grafických prvků. Podporuje vektorové vykreslování, antialiasované renderování, přechody, obrázky, barvy, prostorovou transformaci, vytváření a zobrazení PDF souborů. [11]

Core Image Framework

Tento framework poskytuje velkou sadu filtrů, které lze aplikovat na obrázky, popřípadě videa. Výhodou je, že všechny úpravy provedené tímto frameworkem jsou nedestruktivní, a tak původní zdroj zůstane nepozměněn. Vzhledem k tomu, že jsou filtry optimalizované pro iOS zařízení, tak je aplikace jednotlivých filtrů rychlá a efektivní. [11]

Core Text Framework

Nabízí jednoduché rozhraní pro práci s textem a manipulací s fonty. Tento framework se používá v aplikacích, kde se nevyužívá TextKit, ale přesto aplikace vyžaduje i pokročilejší práci s textem. Framework umožňuje pokročilé funkce, jako je například obtékání textu kolem objektu a použití více druhů fontu. [11]

Core Video Framework

Framework poskytuje vyrovnávací paměť pro Core Media framework. Většina aplikací, nepotřebuje přistupovat k tomuto frameworku přímou cestou. [11]

Game Controller Framework

Game Controller Framework umožňuje vyhledat a konfigurovat MFi[6] herní ovladače. Ovladače mohou být s iOS zařízením spojené fyzicky, nebo bezdrátově přes Bluetooth. Framework aplikaci informuje o tom, že je ovladač připojený a umožňuje určit, které ovládací vstupy jsou relevantní k dané aplikaci. [11]

GLKit Framework

GLKit framework obsahuje třídy, které zjednodušují práci při tvorbě aplikací využívající OpenGL ES. [11]

Image I/O Framework

Tento framework poskytuje rozhraní pro import a export obrázkových dat a metadat. Framework podporuje všechny standardní formáty obrázků dostupné na iOS, a lze jej také využít k přístupu k Exifu[7] a IPTC[8] metadatům. [11]

Media Player Framework

Media Player framework umožňuje přehrávat video a audio soubory z aplikace a umožňuje implementovat následující funkce: [11]

- Přehrávání videa na jiné obrazovce přes AirPlay
- Přistupovat k hudební knihovně v zařízení
- Vyhledat a následně přehrávat skladby a seznamy skladeb
- Konfigurace přehrávání filmů
- Detekovat, kdy je video přehráváno přes AirPlay
- Zobrazit právě přehrávanou skladu na uzamčené obrazovce zařízení

Metal Framework

Framework Metal je určený pro tvorbu 3D her, díky kterému je možné vytěžit z her neuvěřitelně vypadající grafické zpracování. Prekompilací největšího možného množství shader kódu, je procesor zbaven nutnosti tento kód během hry kompilovat. Tím se šetří drahocenný čas, který je zapotřebí pro zpracování, a tím pádem lze výpočetní výkon využít k jiným účelům. [11]

[6]MFi- příslušenství vyrobené pro iOS zařízení (Made For iPhone)
[7]Exif - specifikace pro formát metadat, vkládaných do souborů digitálními fotoaparáty
[8]IPTC - slouží k ukládání textových informací do obrazových souborů

OpenAL Framework

Open Audio Library je multiplatformní rozhraní, které umožňuje implementovat prostorový zvuk do aplikací. Využívá se hlavně u her. Vzhledem k tomu, že jde o otevřený standard, tak lze jednoduše exportovat vytvořené moduly i na jiné platformy. [11]

OpenGL ES Framework

OpenGL ES framework je část OpenGL rozhraní pro počítačové vykreslování 2D a 3D grafiky pro použití ve video hrách, akcelerovaných za pomoci grafického procesoru. [11]

Photos UI Framework

Tento framework umožňuje vývojářům použít rozšíření k aplikacím na úpravu fotografií. Samotné úpravy fotografií lze pomocí rozšíření provádět v aplikaci Obrázky. Odpadá tím nutnost obrázek otevřít v konkrétní aplikaci na úpravu fotografií. Díky tomu lze velmi jednoduše nástroje dané aplikace použít právě v nativní aplikaci Obrázky. [11]

Quartz Core Framework

Quartz Core framework obstarává rozhraní pro Core Animation. Core Animation je technologie, která umožňuje vytvářet jednoduše animace v aplikacích, které jsou rychlé a nenáročné na výpočetní výkon. [11]

SceneKit Framework

Je framework, který je určený k vytváření jednoduchých 3D her. Obsahuje také fyzikální engine[9], v kterém je možné využít například gravitaci, kolizi objektů, elektromagnetismus a turbulenci. [11]

SpriteKit Framework

SpriteKit je framework určený k tvorbě 2D her. SpriteKit poskytuje potřebnou infrastrukturu, kterou většina her potřebuje. Stará se o vykreslování grafiky, animace, zvuky a podporuje také fyzikální engine. [11]

[9]Engine - jádro počítačové hry, databázového stroje nebo programu

4.3 Core Services

Vrstva Core Service obsahuje základní systémové služby a frameworky, které využívají veškeré iOS aplikace. Mezi klíčové frameworky této vrstvy patří Core Foundation a Foundation framework.

4.3.1 Vysokoúrovňové služby Core Services

Peer-to-Peer Services

Peer-to-peer spojení mezi více zařízeními pomocí technologie Bluetooth se využívá převážně u her pro více hráčů. Tato služba se dá využít i u aplikací, které si mezi sebou například dokáží posílat určité soubory. [12]

iCloud Storage

iCloud je služba, která umožňuje uchovávat a synchronizovat dokumenty uživatele napříč všemi zařízeními, která jsou přihlášená pod stejným účtem. Výhodou je, že v případě ztráty zařízení uživatel nepřijde o svá data, protože jsou bezpečně uložená na iCloudu. [12]

Block Objects

Blok je jazykový konstrukt jazyka C, který je možný používat ve stávajícím C nebo Objective-C kódu. Blok je v podstatě anonymní funkcí, která se v jiných programovacích jazycích nazývá closure nebo lambda. Bloky se velmi často používají jako návratové funkce. [12]

Data Protection

Aplikace, které ukládají citlivá data, mohou využívat vestavěnou podporu šifrování. V případě, že aplikace označí soubor jako chráněný, systém ho automaticky uloží do paměti v zašifrované podobě. Pokud je zařízení uzamčené, tak je obsah souboru nedostupný jak aplikaci, tak případnému útočníkovi. Ve chvíli, kdy dojde k odemčení zařízení, se vygeneruje dešifrovací klíč, který aplikaci umožní soubor přečíst. [12]

In-App Purchase

Umožňuje vývojářům ve svých aplikacích prodávat doplňkový obsah. Tato služba se implementuje pomocí frameworku StoreKit, který je popsán níže. [12]

File-Sharring Support

Tato služba umožňuje zobrazit soubory uživatele v aplikacích, a to v programu iTunes[10] ve verzi 9.1, nebo novější. V případě, že aplikace tuto službu podporuje, je možné prostřednictvím iTunes do konkrétní aplikace nahrávat soubory a naopak soubory z aplikace ukládat do počítače. [12]

Grand Central Dispatch

Grand Central umožňuje správu úloh v aplikaci. Tato služba kombinuje asynchronní model programování s vysoce optimalizovaným jádrem a poskytuje tak jednoduchou a zároveň efektivní alternativu k vláknovému programování. [12]

SQLite

SQLite knihovna umožňuje vytvořit odlehčenou databázi přímo v aplikaci, bez nutnosti se připojovat k vzdálenemu serveru. Přímo z aplikace lze vytvořit lokální databázi a spravovat v ní záznamy. Knihovna je určená k univerzálnímu použití a je neustále optimalizovaná tak, aby poskytla co nejrychlejší přístup k záznamům. [12]

XML Support

Framework foundation poskytuje třídu NSXML Parser, pro načítání elementů z XML dokumentů. Pro rozšířenou práci s XML dokumenty je tu libxml2 knihovna. Jedná se o open source knihovnu, která umožňuje parsování, zápis XML dat a transformaci XML obsahu do HTML. [12]

[10] iTunes - aplikace určená pro organizaci a přehrávání multimediálních souborů

4.3.2 Frameworky dostupné v Core Services

Accounts Framework

Framework Accounts nabízí jednotné přihlášení některých uživatelských účtů. Výhodou je jednotné přihlášení k účtu. Tím odpadá vyplňování přihlašovacích údajů v aplikacích, které daný uživatelský účet využívají. Tento framework se nejčastěji používá se Social frameworkem, který je popsaný v této kapitole. [12]

Address Book Framework

Adress Book framework umožňuje přistupovat k databázi kontaktů. Kontakty lze vytvořit, smazat a upravovat. [12]

CFNetwork Framework

Nabízí síťové rozhraní pro využití síťových protokolů. Tento framework usnadňuje práci se síťovým spojením přes FTP a HTTP. Framework umožňuje následující funkce: [12]

- Použití BSD soketů
- Vytvoření zabezpečeného připojení za použití SSL nebo TLS
- Rozpoznání DNS hosta
- Práce s HTTP servery, autentizace na HTTP a HTTPS servery
- Práce s FTP servery
- Publikování, rozpoznání a procházení služeb Bonjour[11]

CloudKit Framework

CloudKit nabízí rozhraní pro přenášení dat mezi aplikací a iCloudem. CloudKit dává vývojářům kontrolu nad tím, kdy k přenosům na iCloud dochází a umožňuje spravovat různé typy dat. [12]

[11]Bonjour - síťová technologie společnosti Apple

Core Data Framework

Core Data framework je technologie která se stará o uchovávání dat v aplikaci.

- Uchování dat v SQLite databázi pro optimální výkonnost
- Třída NSFetchedResultsController pro správu zobrazení table views
- Správa undo a redo operací
- Podpora validace atributů
- Podpora změn a zajištění relace mezi objekty tak aby zůstali konzistentními
- Podpora pro seskupování, filtrovaní a organizaci dat v paměti

Core Foundation Framework

Core Foundation framework je základem pro Foundation framework. Framework obsahuje nízko úrovňovou implementací stejných funkcí, jako nabízí framework Foundation. [12]

Core Location Framework

Tento framework dokáže určit aktuální geografickou polohou zařízení. K samotnému určení se využívá GPS, mobilní síť, popřípadě WIFI, aby se mohla co nejpřesněji určit zeměpisná šířka a délka zařízení. [12]

Core Media Framework

Core Media framework je nízko úrovňový framework, kterého využívá AV Foundation framework. Framework je určen programátorům, kteří potřebují vykonávat komplikovanější práci s audiem a videem. [12]

Core Telephony Framework

Tento framework nabízí přístup k informacím o mobilní sítí uživatele. Framework se využívá hlavně u VoIP aplikací. [12]

EventKit Framework

EventKit framework nabízí rozhraní pro přístup k událostem v kalendáři uživatele. Framework umožňuje vývojářům využít následující funkce: [12]

- Přístup k jednotlivým událostem

- Přidání nové události do kalendáře
- Vytvoření připomínek
- Nastavení upozornění u jednotlivých událostí

Foundation Framework

Tento framework umožňuje přístup k hlavním třídám v Objective-C a poskytuje podporu pro následující funkce: [12]

- Kolekce (pole, slovníky, atd)
- Práce s balíčky
- Práce s řetězci
- Nastavení data a času
- Nastavení aplikací
- URL a streamování
- Vlákna a smyčky
- Bonjour síťový protokol
- Nastavování komunikační portů
- Internacionalizaci
- Regulární výrazy
- Podporu cashování

HealthKit Framework

HealthKit je nový framework, který spravuje zdravodní data uživatele. S množstvím různých fitnes aplikací, které uživatelé používají, je možné sledovat zdravotní data a fitnes informace všech druhů. Vzhledem k tomu, že uživatelé používají různé aplikace na měření různých aktivit a zdravotních dat, tak ztrácejí přehled, jak na tom vlastně jsou. Pomocí HealthKit frameworku je možné číst nebo zapsat data do jedné centralizované aplikace Health. Výhodou této aplikace je, že si uživatelé mohou prohlédnou veškerá zdravotní data a fitnes informace na jednom místě. [12]

HomeKit Framework

HomeKit je aplikace, která nabízí standardizovaný způsob komunikace s chytrými zařízeními v domácnosti. Vývojáři mohou do své aplikace HomeKit implementovat, a tak uživateli umožnit ovládat domácnost z jedné centralizované aplikace místo toho, aby v jedné aplikaci rozsvítil světla, v další otevřel garážová vrata a v poslední nastavil například termostat. [12]

JavaScript Core Framework

Tento framework umožňuje evaulaci kódu napsaného v JavasScriptu a pársování JSON dat. [12]

NewsstandKit Framework

Aplikace Newsstand poskytuje centrální místo pro všechny časopisy a noviny, které uživatel elektronicky odebírá. Vydavatelé, kteří chtějí vytvořit aplikaci pro své čtenáře, kteří si v ní mohou nová vydání přečíst, tak využívají právě framework NewsstandKit. Ten jim po zapnutí aplikace umožňuje stahovat nová vydání na pozadí a v případě, že je obsah stažen a připraven, tak na to uživatele pomocí notifikace upozorní. [12]

PassKit Framework

Aplikace Passbook poskytuje uživatelům prostor v zařízení, kde mohou uchovávat jednoduše kupóny, vstupenky, letenky a slevové kartičky. Místo toho, aby museli mít uživatelé všechny tyto kartičky fyzicky u sebe, tak je nyní může mít v telefonu. Framework PassKit tedy poskytuje Objective-C rozhraní pro implementování této funkce do vyvíjených aplikací. [12]

Quick Look Framework

Quick Look framework nabízí příme rozhraní pro zobrazení souborů, které vyvíjená aplikace přímo nepodporuje. Tento framework je vhodný pro aplikace, které stahují soubory z internetu, nebo z neznámých zdrojů. Po stažení daného souboru, se jeho obsah zobrazí přímo v uživatelském rozhraní. [12]

Safari Services Framework

Safari Services Framework poskytuje podporu pro přidání URL adres do seznamu četby v aplikaci Safari. [12]

Social Framework

Tento framework poskytuje jednoduché rozhraní pro přístup k sociálním účtům uživatele. Framework podporuje sociální sítě jako Twitter, Facebook, Sina Weibo a další. Aplikace mohou tento framework využít ke sdílení fotografií a statusů do jednotlivých sociálníc sítí. Social framework spolupracuje s frameworkem Accounts, který mu umožňuje využít jednotného přihlášení. [12]

StoreKit Framework

StoreKit framework umožňuje vývojářům implementovat funkci nákupů za rozšířený obsah jako jsou virtuální peníze, odemčení nadstadardních funkcí, případně nové úrovně v hrách přímo do jejich aplikací. Framework se primárně zaměřuje na uskutečnění a bezpečnost transakce. [12]

System Configuration Framework

System Configuration framework poskytuje rozhraní pro zjištení síťového nastavení zařízení. Pomocí něj lze určit zdali je zařízení připojeno k Wi-Fi nebo pomocí mobilního připojení. [12]

WebKit Framework

Tento framework umožňuje zobrazit v aplikacích HTML obsah. Využívá se často při přihlášení k webovým službám. [12]

4.4 Core OS

Vrstva Core OS poskytuje nízkoúrovňové funkce ostatním technologiím, které jsou na ní postaveny. I když nejsou například v aplikacích využívány přímo, využívají je frameworky z ostatních vrstev. V případě, že vyvíjíte aplikaci, která se zabývá bezpečností, nebo externím hardwarovým příslušenstvím, budete využívat pravděpodobně některé frameworky z této vrstvy.

4.4.1 Frameworky dostupné v Core OS

Accelerate Framework

Tento framework poskytuje rozhraní pro zpracování obrazu, DSP[12] a lineární algebry. Výhoda tohoto frameworku je ta, že je optimalizován pro všechna iOS zařízení. Vývojáři mají jistotu, že jejich kód poběží efektivně na všech zařízeních. [13]

Core Bluetooth Framework

Core Bluetooth framework dovoluje vývojářům komunikovat se zařízeními prostřednictvím Bluetooth a umožňuje jím skenovat Bluetooth příslušenství, připojovat se k němu, odpojovat, upozornit na nedostupnost příslušenství. [13]

External Accessory Framework

Nabízí podporu pro komunikaci s externím zařízením připojeného prostřednictvím 30-pin konektoru, lightning konektoru a bezdrátově přes Bluetooth. Framework umožňuje také získávat informace o dostupném příslušenství, navázat s ním komunikaci a následně jej ovládat pomocí podporovaných příkazů. [13]

Generic Security Services Framework

Je framework poskytující standardní sadu služeb, které zastřešují zabezpečení v iOS aplikacích. [13]

Network Extension Framework

Poskytuje podporu pro konfiguraci a řízení VPN. Framework umožňuje vytvořit VPN[13] konfiguraci. Samotného spojení lze docílit manuálně, nebo v reakci na konkrétní událost. [13]

[12]DPS - digitální signálový procesor
[13]VPN - virtuální privátní síť

Security Framework

Zajišťuje bezpečnost citlivých dat v aplikacích. Framework obsahuje rozhraní pro certifikáty, soukromé a veřejné klíče a generování kryptografických pseudonáhodných čísel. [13]

System

Systém obsahuje jádro, ovladače a nízkoúrovňové rozhraní UNIX. Operační systém zajišťuje správu virtuální paměti, vlákna, souborový systém, síťovou a procesovou komunikaci. Ovladače této vrstvy poskytují také rozhraní mezi dostupným hardwarem a systémovými frameworky. Z bezpečnostních důvodů je přístup k jádru a ovladačům omezen a přístup k nim mají pouze vybrané systémové aplikace a frameworky.
iOS poskytuje sadu rozhraní pro přístup k mnoha nízko úrovňovým částím operačního systému a tím následující funkce: [13]

- Vlákna (POSIX vlákna)
- Sít (BSD sokety)
- Přístup k souborovému systému
- Standardní I/O
- Bonjour a DNS služba
- Alokování paměti
- Matematické výpočty

5 Aplikace využívající vybranné frameworky

5.1 Aplikace Core Location

Vaše iOS zařízení dokáže prostřednictvím frameworku Core Location lokalizovat aktuální polohu. Samotný framework Core Location umí určit polohou zařízení podle čtyř různých geolokačních technologií: GPS[14], triangulace založené na umístění BTS[15], WPS[16] a iBeacon.

Technologie GPS určuje aktuální zeměpisnou polohu s pomocí mikrovlnných signálů z více různých satelitů, které přijímá. Tato technologie je nejpřesnější, ale lze jí využít pouze v iOS zařízeních, která obsahují GPS modul.[14]

Při triangulaci založené na pozici BTS vysílačů se poloha telefonu určuje na základě polohy vysílačů, které jsou v dosahu zařízení. Tento způsob určení polohy může být velmi přesný ve městech a jiných oblastech s vysokou hustotou vysílačů, ale není již tak přesný v oblastech, kde jsou vysílače umístěné dále od sebe. Tuto technologii lze využívat pouze v zařízeních s GSM modulem. [15]

WPS, využívá k určeni polohy IP adresu Wi-Fi připojení telefonu, na jejímž základě následné za využití velké databáze známých providerů této služby odhaduje, kde se telefon nachází. Technologie WPS je velmi nepřesná a může se mýlit i o mnoho kilometrů. [16]

Poslední technologie iBeacon je založena na mikrolokalizaci pomocí takzvaných "majáků", neboli iBeaconu. Tyto majáky v sobě obsahují Bluetooth LE a jsou energeticky velmi úsporné. [17]

První tři způsoby lokalizace jsou energeticky velmi náročné a výrazně snižují výdrž zařízení. Je nutné pamatovat, že při používání tohoto frameworku je třeba dbát na to, aby se námi vytvořená aplikace nedotazovala na aktuální polohu, když to nebude nezbytně nutné. V tomto frameworku je možné specifikovat přesnost určení zeměpisné polohy, a tím snížit nároky na baterii zařízení. [18]

[14]GPS - Globální polohovací systém
[15]BTS - Systém základnových stanic
[16]WPS - Wi-Fi polohovací systém

5.1.1 Popis aplikace

Abych demonstroval práci s tímto frameworkem, rozhodl jsem se vytvořit aplikaci, která po stisknutí tlačítka ukáže zeměpisnou šířku, zeměpisnou délku, aktuální rychlost, nadmořskou výšku, vzdálenost aktuální polohy od hlavního města České republiky a dekódovanou adresu.
Abychom mohli s CoreLocation frameworkem vůbec pracovat, musíme tento framework nejdříve importovat do hlavičkového souboru naší třídy, jak můžeme vidět na obrázku 5.1, a to příkazem:

```
#import<CoreLocation/Corelocation.h>
```

```
#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>

@interface ViewController : UIViewController <CLLocationManagerDelegate>

{
    CLLocationManager *manager;
    CLGeocoder *geocoder;
    CLPlacemark *placemark;
}

@property (weak, nonatomic) IBOutlet UILabel *longitudeLabel;
@property (weak, nonatomic) IBOutlet UILabel *latitudeLabel;
@property (weak, nonatomic) IBOutlet UILabel *adressLabel;
@property (weak, nonatomic) IBOutlet UILabel *altitudeLabel;
@property (weak, nonatomic) IBOutlet UILabel *speedLabel;
@property (weak, nonatomic) IBOutlet UILabel *distanceFromPragueLabel;

- (IBAction)getLocationButton:(id)sender;
- (IBAction)stopUpdatingLocation:(id)sender;

@end
```

Obrázek 5.1: Hlavičkový soubor aplikace Core Location

Dále budeme potřebovat nastavit správce zeměpisné polohy. Je třeba tedy vytvořit ukazatel na správce zeměpisné polohy a to tak, že v hlavičkovém souboru deklarujeme:

```
CLLocationManager *manager;
```

Nyní je nutné správci přidělit ještě delegáta. Správce bude volat metody delegáta v případě, že budou informace o zeměpisné poloze k dispozici, nebo se změní. Delegát musí odpovídat protokolu CLLocationManager a přidáme ho následnovně:

```
@interface ViewController : UIViewController
<CLLocationManagerDelegate>
```

Do hlavičkového souboru zároveň ještě přidáme dvě deklarace, díky kterým budeme později moci dekódovat adresu ze zeměpisných souřadnic na lidsky čitelnou adresu a to:

```
CLLGeocoder *geocoder;
CLPlacemark *placemark;
```

Na závěr je třeba přidat ještě tlačítko, které po stisknutí začne zjišťovat polohou zařízení a outlety, v kterých se budou aktualizovat data v uživatelském rozhraní aplikace, v závislosti na aktuální poloze zařízení.

Nyní, když máme deklaraci za sebou, přejdeme k implementaci metody, která se zavolá po stisknutí tlačítka `getLocationButton`. Na obrázku 5.2 si můžeme všimnout, že nejdříve nastavíme požadovanou přesnost pomocí konstanty:

```
manager.desiredAccuracy = kCLLocationAccuracyBest;
```

```
- (IBAction)getLocationButton:(id)sender {

    manager.delegate = self;
    manager.desiredAccuracy = kCLLocationAccuracyBest;
    manager.distanceFilter = 100.0f;


    [manager startUpdatingLocation];
    [manager requestWhenInUseAuthorization];

}
```

Obrázek 5.2: Metoda getLocationButton

Přesnost se nastavuje v metrech a je třeba dobře promyslet, jak velkou přesnost bude tato aplikace vyžadovat. V tomto případě jsem zvolil nejvyšší možnou přesnost, aby aplikace byla při dekódování adresy co možná nejpřesnější. V případě, že aplikace nevyžaduje tak přesné určení polohy, je vhodnější využít konstanty `kCLLocationAccuracyNearestTenMeters` s přesností na 10 metrů, `kCLLocationAccuracyHundredMeters` s přesností na 100 metrů, `kCLLocationAccuracyKilometer` s přesností na 1 kilometr a poslední možností je `kCLLocationAccuracyThreeKilometers` s přesností na 3 kilometry.

Dále nastavujeme filtr vzdálenosti pomocí:

```
manager.distanceFilter = 100.0 f;
```

Tímto říkáme správci zeměpisné polohy, aby nás upozornil na změnu polohy pouze tehdy, pokud zařízení překročí námi stanovenou hodnotu. V tomto případě je filtr nastaven na hodnotu 100 metrů.

Na závěr se uživatele tážeme, zdali umožní aplikaci využívat polohové služby zařízení a spouštíme správce zeměpisné polohy, který zavolá metodu delegáta pokaždé, když zařízení překročí stanovený filtr vzdálenosti.

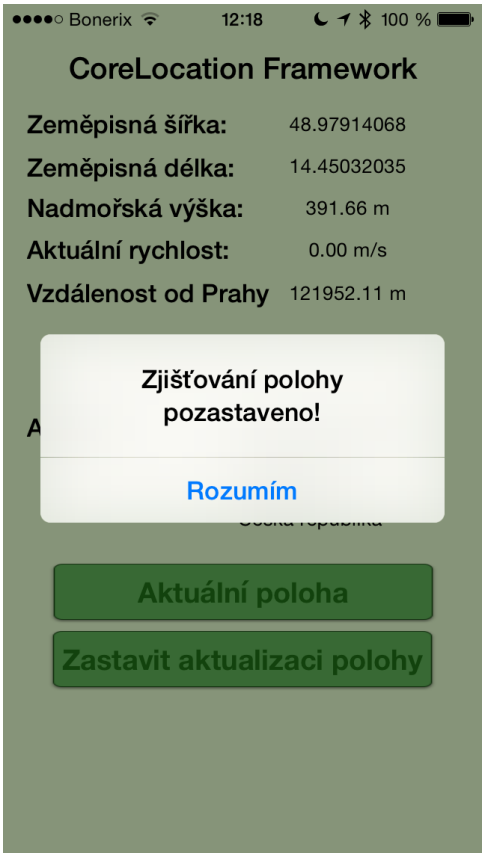
V první části metody `didUpdateLocation`, která je na obrázku 5.3, se přiřazují hodnoty získané správcem zeměpisné polohy jednotlivým outletům v uživatelském rozhraní. Konkrétně zeměpisná šířka, délka, nadmořská výška, aktuální rychlost a vzdálenost od určitého místa.

V druhé části metody se volá další metoda `reverseGeocodeLocation`, pomocí které se převádí hodnoty zeměpisné šířky a délky na čitelnou adresu. V tomto případě metoda obsahuje atributy pro určení popisného čísla, ulice, poštovního směrovacího čísla, města, kraje a státu, v kterém se zařízení aktuálně nachází.

```
-(void)locationManager:(CLLocationManager *)manager didUpdateToLocation:
    (CLLocation *)newLocation fromLocation:(CLLocation *)oldLocation
{
    CLLocation *currentLocation = newLocation;

    if (currentLocation != nil) {
        self.latitudeLabel.text  = [NSString stringWithFormat:@"%.8f",
            currentLocation.coordinate.latitude];
        self.longitudeLabel.text = [NSString stringWithFormat:@"%.8f",
            currentLocation.coordinate.longitude];
        self.altitudeLabel.text  = [NSString stringWithFormat:@"%.2f",
            currentLocation.altitude];
        self.speedLabel.text     = [NSString stringWithFormat:@"%.2f m/s",
            currentLocation.speed];

        CLLocation *pragueLocation = [[CLLocation alloc]initWithLatitude:
            50.0755380 longitude:14.4378000];
        CLLocationDistance distanceFromPrague = [pragueLocation
            getDistanceFrom:oldLocation];
        NSString *distanceString = [[NSString alloc]initWithFormat:@"%.2f m",
            distanceFromPrague];
        self.distanceFromPragueLabel.text = distanceString;
    }

    [geocoder reverseGeocodeLocation:currentLocation completionHandler:^
        (NSArray *placemarks, NSError *error) {

        if (error == nil && [placemarks count] > 0) {

            placemark = [placemarks lastObject];

            self.adressLabel.text = [NSString stringWithFormat:@"%@ %@\n%@ %@\n
                %@\n%@",
            placemark.subThoroughfare,
            placemark.thoroughfare,
            placemark.postalCode,
            placemark.locality,
            placemark.administrativeArea,
            placemark.country];
        } else {

        }NSLog(@"%@", error.description);
    } ];
}
```

Obrázek 5.3: Metoda didUpdateToLocation

Obrázek 5.4: Aplikace Core Location



Obrázek 5.5: Upozornění na zastavení zjišťování polohy

Druhé tlačítko této aplikace volá metodu `stopUpdatingLocation`, kterou můžeme vidět na obrázku 5.6. V této metodě voláme další metodu `[manager stopUpdatingLocation]`, která říká správci zeměpisné polohy, aby přestal posílat delegátu aktualizované hodnoty. Dále voláme metodu `[alert show]`, která uživatele upozorní na to, že aplikace pozastavila aktualizace současné polohy zařízení. Toto upozornění je možné vidět na obrázku 5.5.

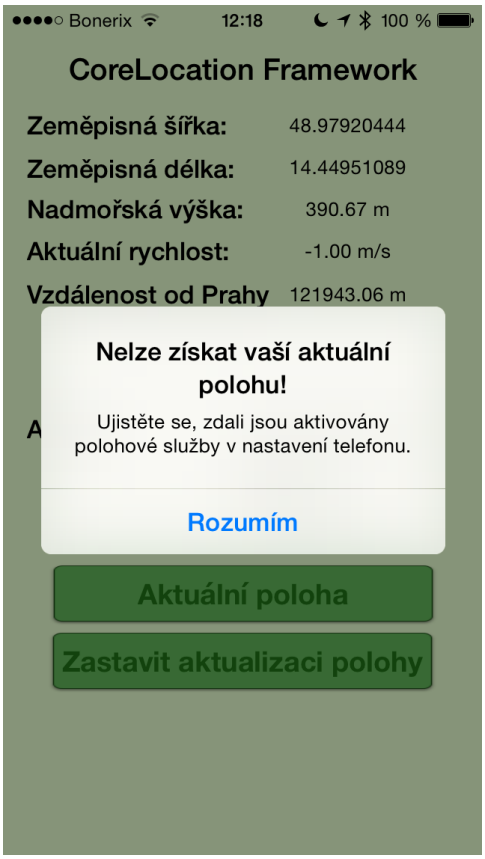
```
- (IBAction)stopUpdatingLocation:(id)sender {

    [manager stopUpdatingLocation];

    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"Zjišťování polohy pozastaveno!"
                          message:nil
                          delegate:nil
                          cancelButtonTitle:@"Rozumím"
                          otherButtonTitles:nil, nil];
    [alert show];

}
```

Obrázek 5.6: Metoda stopUpdatingLocation

V aplikaci je dále použita metoda `didFailWithError`, která se zavolá v případě, že uživatel nemá povoloné využití polohových služeb pro tuto aplikaci. Aplikace na to uživatele upozorní zprávou, kterou můžete vidět na obrázku 5.8.

```
-(void)locationManager:(CLLocationManager *)manager didFailWithError:(NSError *
    )error{

    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"Nelze získat vaší aktuální polohu!"
                          message:@"Ujistěte se, zdali jsou aktivovány polohové
                              služby v nastavení telefonu."
                          delegate:nil
                          cancelButtonTitle:@"Rozumím"
                          otherButtonTitles:nil, nil];
    [alert show];
}
```

Obrázek 5.7: Metoda didFailWithError



Obrázek 5.8: Upozornění na zakázaný přístup k poloze

5.2 Aplikace MapKit

Další aplikace, kterou jsem vytvořil, využívá frameworku MapKit. Tento framework umožňuje zobrazovat satelitní, hybridní a vektorové mapové podklady a anotovat vlastní body zájmů. Rozhodl jsem se na mapě anotovat všechny budovy Pedagogické fakulty Jihočeské univerzity v Českých Budějovicích.

5.2.1 Popis aplikace

Do hlavičkového souboru třídy, který je na obrázku 5.9, musíme stejně jako u první aplikace importovat framework, se kterým chceme pracovat pomocí příkazu `#import<MapKit/MapKit.h>`. Dále deklarujeme metody delegáta:

```
@interface ViewController : UIViewController
<MKMapViewDelegate>
```

V neposlední řadě přidáme outlet typu MKMapView, v kterém se bude mapa načítat.

```
#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>

@interface ViewController : UIViewController <MKMapViewDelegate>
@property (strong, nonatomic) IBOutlet MKMapView *mapView;
@end
```

Obrázek 5.9: Hlavičkový soubor aplikace MapKit

První věcí, která byla třeba v této aplikaci nastavit, bylo území, které se má zobrazit v okamžiku, kdy se aplikaci spustí. Ve výchozím stavu se načte mapa zeměkoule, a anotace je pak třeba na mapě hledat ručně. Na obrázku 5.10 můžete vidět část kódu, který tento problém řeší. Nejdříve se do proměnné `pocatecniPozice` uloží hodnota zeměpisné šířky a délky. Poté je třeba nastavit šířku a délku území, kerá se má na mapě zobrazit. V proměnných `sirkaUzemi vyskaUzemi` se nastaví hodnota na 2000 metrů. Do proměnné `uzemi` se uloží zmíněné proměnné a tato proměnná se následně bude načítat v `mapView`.

```
CLLocationCoordinate2D pocatecniPozice = CLLocationCoordinate2DMake(48.9730664,
    14.4698695);
CLLocationDistance sirkaUzemi = 2000;
CLLocationDistance vyskaUzemi = 2000;

MKCoordinateRegion uzemi = MKCoordinateRegionMakeWithDistance(pocatecniPozice,
    sirkaUzemi, vyskaUzemi);
```

Obrázek 5.10: Nastavení zobrazovaného území na mapě

Nyní, když je připravený kód pro zobrazení specifického území, je třeba mapu nastavit a načíst v `mapView`. První věcí, kterou nastavujeme, je `mapType`, který může mít tři možnosti. První z nich je `MKMapTypeStandard` pro standardní zobrazení, kterou jsem zvolil i pro tuto ukázkovou aplikaci. Další možností je `MKMapTypeSatelite` pro zobrazení satelitních mapových podkladů a poslední možnostní je `MKMapyTypeHybrid` pro hybridní zobrazení.

```
self.mapView.mapType = MKMapTypeStandard;
```

Obrázek 5.11: Nastavení typu mapy

V mapě nastavíme zobrazení bodů zájmů pomocí `showsPointsOfInterest`, jako jsou například: obchody, restaurace, lékárny, památky a další.

```
self.mapView.showsPointsOfInterest = YES;
```

Obrázek 5.12: Nastavení bodů zájmů

Pomocí `showUserLocation` nastavujeme zobrazení aktuální polohy zařízení na mapě.

```
self.mapView.showsUserLocation = YES;
```

Obrázek 5.13: Nastavení zobrazení polohy uživatele

Anotování jedotlivých bodů zájmů je možné vidět na obrázku 5.14. Anotace je typu `MKPointAnnotation`. Anotace má paramatery `title` pro hlavní titulek `subtitle` pro podtitulek a `coordinate` pro určení zeměpisné šířky a délky.

```
MKPointAnnotation *annotationJeronymova = [[MKPointAnnotation alloc]init];
annotationJeronymova.title = @"Pedagogická fakulta";
annotationJeronymova.subtitle = @"Jeronýmova 10, České Budějovice, 371 15";
annotationJeronymova.coordinate = CLLocationCoordinate2DMake(48.9738030, 14.4823450);

MKPointAnnotation *annotationDukelska = [[MKPointAnnotation alloc]init];
annotationDukelska.title = @"Pedagogická fakulta";
annotationDukelska.subtitle = @"Dukelská 9, České Budějovice, 371 15";
annotationDukelska.coordinate = CLLocationCoordinate2DMake(48.9714240, 14.4773000);

MKPointAnnotation *annotationUTriLvu = [[MKPointAnnotation alloc]init];
annotationUTriLvu.title = @"Pedagogická fakulta";
annotationUTriLvu.subtitle = @"U Tří Lvů, České Budějovice, 371 15";
annotationUTriLvu.coordinate = CLLocationCoordinate2DMake(48.9709459, 14.4776309);

MKPointAnnotation *annotationSadky = [[MKPointAnnotation alloc]init];
annotationSadky.title = @"Pedagogická fakulta";
annotationSadky.subtitle = @"Sádky, České Budějovice, 371 15";
annotationSadky.coordinate = CLLocationCoordinate2DMake(48.9763003, 14.4598597);
```

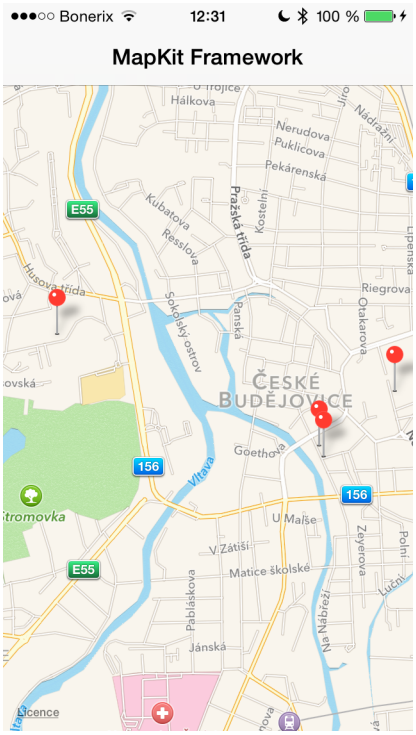
Obrázek 5.14: Anotování jednotlivých bodů zájmů

Načtení jednotlivých anotací v mapě se provádí pomocí `addAnnotation`.

```
[self.mapView addAnnotation:annotationJeronymova];
[self.mapView addAnnotation:annotationDukelska];
[self.mapView addAnnotation:annotationUTriLvu];
[self.mapView addAnnotation:annotationSadky];
```

Obrázek 5.15: Načtení bodů zájmů do mapView

Po spuštění aplikace se načtou mapové podklady Českých Budějovic a konkrétní území, které je specifikováno výše. Území bylo záměrně zvoleno tak, aby byly viditelné všechny čtyři budovy fakulty. Na obrázku 5.16 je možné vidět, co se do aplikace načítá. Po tapnutí na konkrétní anotaci, se zobrazí bližší informace o místě, jak je to možné vidět na obrázku 5.17.



Obrázek 5.16: Aplikace MapKit

Obrázek 5.17: Detail anotace

5.3 Aplikace Core Image

Poslední dílčí aplikace, kterou jsem se rozhodl vytvořit, je k vidění na obrázku 5.23 a 5.24. Aplikace využívá framework Core Image a umožňuje pořídit fotografii, případně jí vybrat z galerie uživatele a následně na ní aplikovat jeden z devíti předdefinovaných filtrů.

5.3.1 Popis aplikace

Do hlavičkového souboru třídy, který je na obrázku 5.18, musíme stejně jako u první aplikace importovat framework, se kterým chceme pracovat pomocí příkazu `#import<CoreImage/CoreImage.h>`. Dále deklarujeme metody delegáta:

```
@interface ViewController : UIViewController
<UINavigationControllerDelegate,
UIImagePickerControllerDelegate>
```

Poté přidáme outlet typu `UIImageView`, v kterém budeme fotografie zobrazovat. Je také potřebova deklarovat tlačítka pro výběr, vyfocení a uložení fotografie. V neposlední řadě deklarujeme tlačítka pro devět fitrů, které budeme na obrázky aplikovat.

```
#import <UIKit/UIKit.h>
#import <CoreImage/CoreImage.h>

@interface ViewController : UIViewController <UINavigationControllerDelegate,
    UIImagePickerControllerDelegate>

@property (strong, nonatomic) IBOutlet UIImageView *imageView;

- (IBAction)pickImageButton:(id)sender;
- (IBAction)captureImageButton:(id)sender;
- (IBAction)saveImageButton:(id)sender;

- (IBAction)blackAndWhiteFilterButton:(id)sender;
- (IBAction)photoEffectInstantFilterButton:(id)sender;
- (IBAction)photoEffectTransgerFilterButton:(id)sender;

- (IBAction)photoEffectSepiaToneFilterButton:(id)sender;
- (IBAction)photoEffectProcessFilterButton:(id)sender;
- (IBAction)photoEffectNoirFilterButton:(id)sender;

- (IBAction)photoEffectColorMapFilterButton:(id)sender;
- (IBAction)photoEffectColorInvertFilterButton:(id)sender;
- (IBAction)photoEffectColoroPosterizeFilterButton:(id)sender;
```

Obrázek 5.18: Hlavičkový soubor aplikace Core Image

Při stisknutí tlačítka vybrat, se zavolá metoda `pickImageButton` a umožní uživateli vybrat jakýkoliv obrázek, který má v galerii obrázků, a ten se následně načte do `UIImageView`.

```
- (IBAction)pickImageButton:(id)sender {

    UIImagePickerController *photoPicker = [[UIImagePickerController alloc]init]
        ;

    photoPicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    photoPicker.delegate = self;
    [self presentViewController:photoPicker animated:YES completion:^{

    }];

}
```

Obrázek 5.19: Metoda pickImageButton

Metoda `captureImageButton` se zavolá ve chvíli, kdy uživatel stiskne tlačítko vyfotit. Aplikace se přepne do režimu fotoaparát a umožní vyfotit fotografi, která se pak načte do `UIImageView`.

```
- (IBAction)captureImageButton:(id)sender {

    UIImagePickerController *photoPicker = [[UIImagePickerController alloc]init]
        ;

    photoPicker.sourceType = UIImagePickerControllerSourceTypeCamera;
    photoPicker.delegate = self;
    [self presentViewController:photoPicker animated:YES completion:^{

    }];

}
```

Obrázek 5.20: Metoda captureImageButton

Další dostupnou metodou v této aplikaci je metoda `saveImageButton`a slouží k uložení obrázků do knihovny obrázků uživatele.

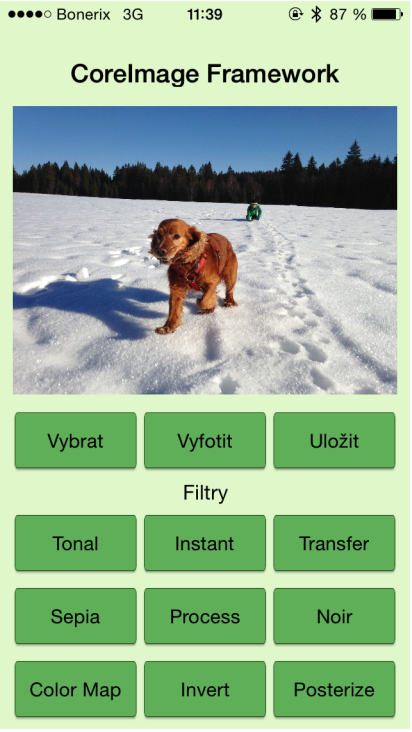
```
- (IBAction)saveImageButton:(id)sender {
    UIImageWriteToSavedPhotosAlbum(self.imageView.image, nil, nil, nil);
    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"Uloženo!"
                          message:@"Obrázek byl úspěšně uložen do alba
                              fotoaparát"
                          delegate:nil
                          cancelButtonTitle:@"Rozumím"
                          otherButtonTitles:nil, nil];

    [alert show];
}
```

Obrázek 5.21: Metoda saveImageButton

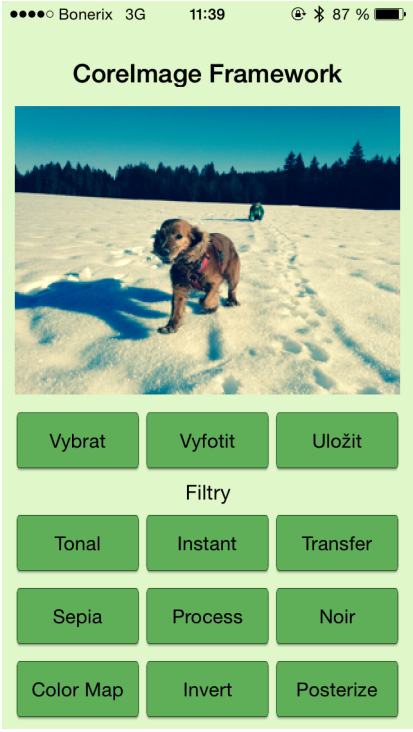
Metoda `photoEffectColorInvertFilterButton`, která se nachází na obrázku 5.22, aplikuje na původní obrázek typu `CIImage` pomocí další metody `photoEffectInstantFilter` filtr a výstupem této metody je proměnná `filteredImageData` , která se následně zobrazí v `imageView`. Touto cestou je řešen i zbytek filtrů dostupných v této aplikaci. U některých z nich je například ještě jeden řádek kódu, v kterém se nastavují parametry daného filtru.

```
- (IBAction)photoEffectColorInvertFilterButton:(id)sender {

    CIImage *originalImage = [[CIImage alloc] initWithImage:self.imageView.image
        ];

    CIFilter *photoEffectInstantFilter = [CIFilter
        filterWithName:@"CIColorInvert"];

    [photoEffectInstantFilter setDefaults];
    [photoEffectInstantFilter setValue:originalImage forKey:@"inputImage"];

    CIImage *filteredImageData = [photoEffectInstantFilter
        valueForKey:@"outputImage"];

    UIImage *filteredImage = [UIImage imageWithCIImage:filteredImageData];
    self.imageView.image = filteredImage;

}
```

Obrázek 5.22: Metoda photoEffectColorInvertFilterButton



Obrázek 5.23: Aplikace Core Image



Obrázek 5.24: Obrázek s filtrem Process

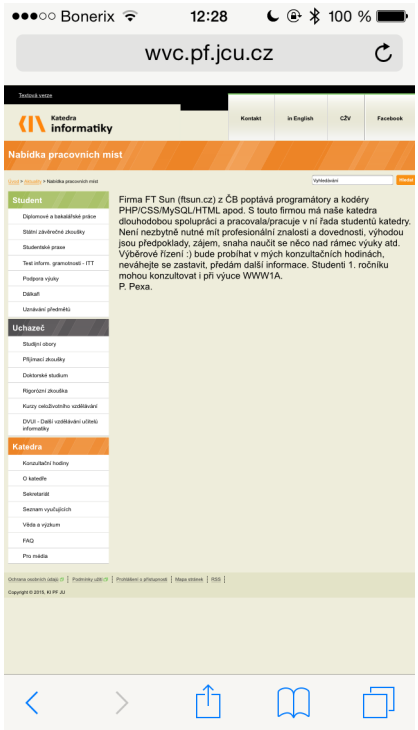
6 Aplikace KIN

Hlavním cílem mé bakalářské práce bylo vytvořit nativní aplikaci pro Katedru informatiky Pedagogické fakulty Jihočeské univerzity v Českých Budějovicích. Aplikace by měla sloužit jako rozcestník uchazečům, studentům a členům katedry, kteří chtějí mít dostupné informace o katedře přímo ve svých chytrých telefonech.

6.1 Možnosti přístupu k informacím o katedře

V současné chvíli mají studenti s platformou iOS přístup k informacím o aktuálním dění pouze přes webovou stránku katedry, která není v současnoti optimalizovaná pro mobilní zařízení. Na obrázku 6.1 je možné vidět, co se uživateli po načtení webových stránek zobrazí. Na obrázku 6.2 pak můžete vidět, co se zobrazí v případě, že si uživatel otevře konkrétní aktualitu. Hledání a případné přečtení požadovaných informací na stránkách je při chůzi s telefonem v jedné ruce prakticky nemožné.

Obrázek 6.1: Web Katedry informatiky

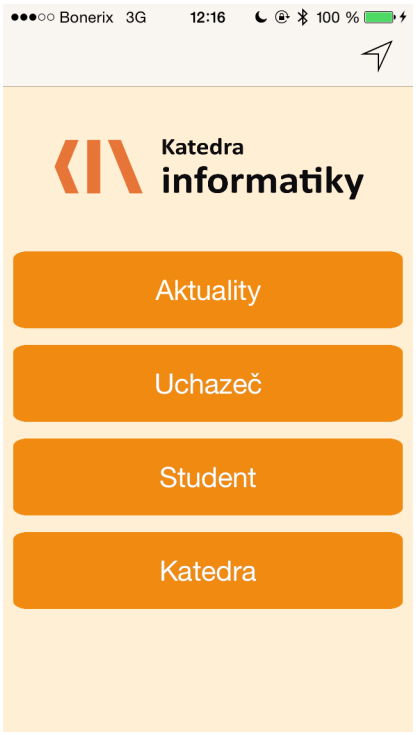


Obrázek 6.2: Detail aktuality

6.2 Návrh

Aplikace má za cíl poskytnout uživatelům co nejrychlejší a nejpohodlnější přístup k požadovaným informacím. Z tohoto důvodu bylo rozhraní aplikace navrženo takovým způsobem, aby uživatel mohl tuto aplikaci snadno obsluhovat jednou rukou. Mým cílem bylo vytvořit takovou aplikaci, která je v souladu s designem operačního systému iOS7 a iOS8, a proto jsem volil minimalistický design, který má napomoci rychlejší orientaci a co nejrychlejšímu přístupu k požadovaným informacím. Velký důraz byl kladen i na to, aby se aplikace mohla ovládat pohodlně jednou rukou, a tím usnadnit a zrychlit její používání.

Po spuštění se v aplikaci zobrazí čtyři hlavní sekce, jak je možné vidět na obrázku 6.3, a to: Aktuality, Uchazeč, Student a Katedra. Každá tato sekce má další podsekce, jak je možné vidět na obrázku 6.4. V pravém horním rohu je dále umístěna ikona symbolizující polohu, pomocí které si uživatel může zobrazit všechny budovy Pedagogické fakulty v Českých Budějovicích.

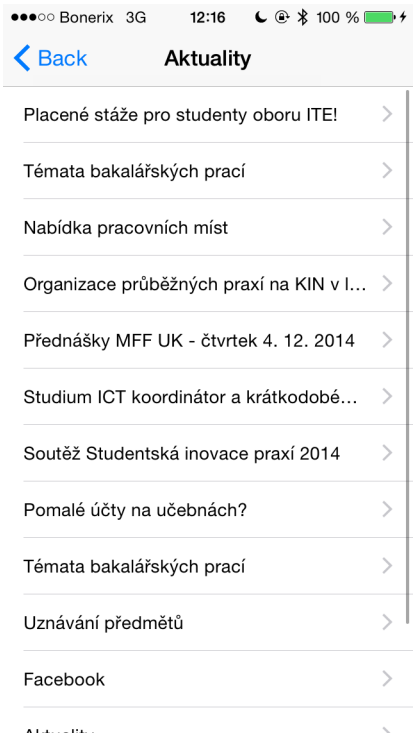


Obrázek 6.3: Aplikace po spuštění

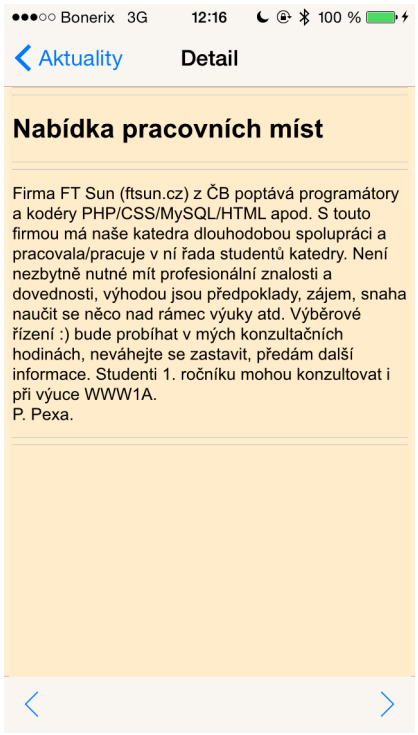
Obrázek 6.4: Sekce Katedra

6.3 Aktuality

Aktuality jsou část aplikace, v které se načítají nejnovější zprávy z katedry. Po stisktnutí tlačítka Aktuality se uživateli načte seznam aktualit, jak je možné vidět na obrázku 6.5, a to chronologicky seřazených od nejnovější po nejstarší. V případě, že uživatel zvolí určitou aktualitu, tak se zobrazí její detail, jak možné vidět na obrázku 6.6.



Obrázek 6.5: Načítání aktualit



Obrázek 6.6: Detail aktuality

Aktuality se načítají z RSS zdroje katedry informatiky a jsou dostupné na adrese `http://wvc.pf.jcu.cz/ki/?article=/rss`. Počet sekcí, které se zobrazí v `tableView` zajišťuje metoda `numberOFSectionsInTableView`. Počet řádků v `tableView` řeší metoda `numberOfRowsInSection`. Počet řádků odpovídá počtu aktualit.

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:
    (NSInteger)section {
    return feeds.count;
}
```

Obrázek 6.7: Metody numberOfSectionInTableView a numberOfRowsInSection

Samotné pársování aktualit z RSS zdroje katedry informatiky zajišťují metody delegáta `NSXMLParserDelegate`. Jedná se o metody na obrázku 6.8, 6.9, 6.10. Parser typu `NSXMLParser` postupně prochází RSS zdroj a do pole `feeds` typu `NSMutableArray` ukládá jednotlivé prvky `item` typu `NSMutableDictionary`. Do proměnné `item` se ukládá titulek `title` a odkaz `link` ke konkrétní aktualitě. Oba dva jsou typu `NSMutableString`.

```
- (void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName
    namespaceURI:(NSString *)namespaceURI
        qualifiedName:(NSString *)qName attributes:(NSDictionary *)
            attributeDict {

    element = elementName;

    if ([element isEqualToString:@"item"]) {

        item    = [[NSMutableDictionary alloc] init];
        title   = [[NSMutableString alloc] init];
        link    = [[NSMutableString alloc] init];
    }
}
```

Obrázek 6.8: Metoda didStartElement

```
- (void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName
    namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName {

    if ([elementName isEqualToString:@"item"]) {

        [item setObject:title forKey:@"title"];
        [item setObject:link forKey:@"link"];
        [feeds addObject:[item copy]];
    }
}
```

Obrázek 6.9: Metoda didEndElement

```
- (void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string {

    if ([element isEqualToString:@"title"]) {
        [title appendString:string];
    } else if ([element isEqualToString:@"link"]) {
        [link appendString:string];
    }
}
```

Obrázek 6.10: Metoda foundCharacters

Poté, co `parser` projde celý RSS zdroj, delegát `NSXMLParserDelegate` zavolá další metodu `parserDidEndDocument`, a tím se `tableView` naplní aktualitami.

```
- (void)parserDidEndDocument:(NSXMLParser *)parser {

    [self.tableView reloadData];
}
```

Obrázek 6.11: Metoda parserDidEndDocument

Jak jsem již zmínil výše, pro zobrazení obsahu konkrétní aktuality stačí stisknout na její titulek v `tableView`, kam se všechny aktuality načítají. O tuto funkci se stará metoda `prepareForSegue`, která zajišťuje předání URL adresy dané aktuality k detailnímu zobrazení ve `webView`. URL adresa aktuality je uložená do proměnné `string`.

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if ([[segue identifier] isEqualToString:@"showDetail"]) {

        NSIndexPath *indexPath = [self.tableView indexPathForSelectedRow];
        NSString *string = [feeds[indexPath.row] objectForKey: @"link"];
        [[segue destinationViewController] setUrl:string];
    }
}
```

Obrázek 6.12: Metoda prepareForSegue

Po vykonání `prepareForSegue` se zavolá metoda `viewDidLoad`, která k URL adrese konkrétní aktuality přidává řetězec `&v=text` a nový řetězec uloží do proměnné `completeLink`. Proměnná `completeLink` je následně převedena na URL adresu a uložena do proměnné `myURL` typu `NSURL`. Tato proměnná je poté převedena pomocí metody `requestWithURL` a následně uložena do nové proměnné `request` typu `NSURLRequest`. Tento `request` se pak načítá ve `webView`, kde dojde k zobrazení aktuality.

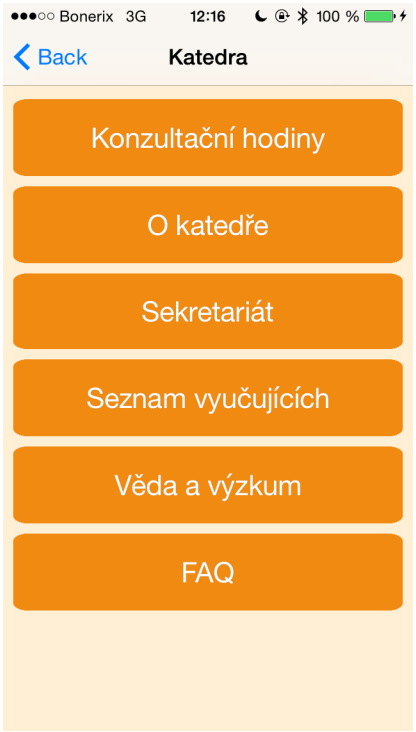
Důvodem, proč používám koncovku `&v=text` je ten, že přes URL s touto koncovkou se načítá textová verze webových stránek katedry. Díky tomu, že přistupuji ke stránkám právě přes textovou verzi webu, nedochází ke zbytečnému stahování souborů, které se jinak stahují při zobrazení grafické verze webu. Právě kvůli tomuto kroku se načítání znatelně urychlilo.

```
- (void)viewDidLoad
    {
        [super viewDidLoad];

        NSString *parsedLink = self.url;
        NSString *extension = @"&v=text";
        NSString *completeLink = [NSString stringWithFormat:@"%@%@",
            parsedLink, extension];

        NSURL *myURL = [NSURL URLWithString: completeLink];

        NSURLRequest *request = [NSURLRequest requestWithURL:myURL];

        self.webView.delegate = nil;
        self.webView.delegate = self;
        self.webView.scalesPageToFit = YES;
        [self.webView loadRequest:request];
    }
```

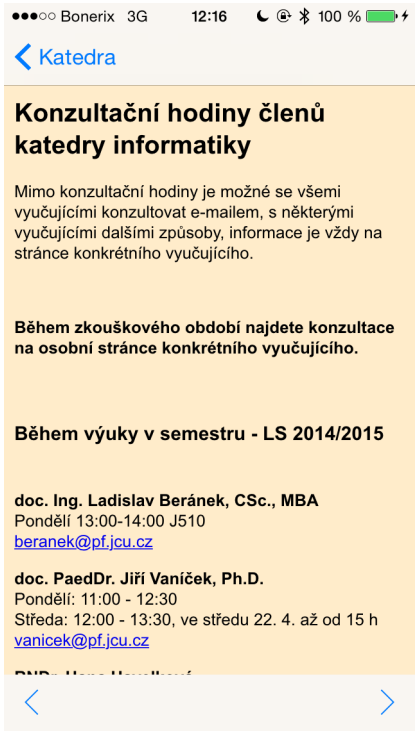
Obrázek 6.13: Metoda pro načtení detailu aktualit viewDidLoad

6.4 Student, Uchazeč, Katedra

Aplikace obsahuje kromě sekce Aktuality tři další sekce a těmi jsou Uchazeč, Student a Katedra. Každá tato sekce obsahuje další podsekci, které již uživateli zobrazí požadovanou informaci. Podsekce má svojí jedinečnou URL adresu, která je možná vidět na obrázku 6.16, a po stisknutí se uživateli načte obsah zvolené podsekce, jak je možné vidět na obrázku 6.15.



Obrázek 6.14: Sekce Katedra



Obrázek 6.15: Konzultační hodiny

Po stisknutí určité sekce se načte metoda `viewDidLoad`, která volá další metodu `loadRequestFromString`, jejíž parametrem je řetězec s URL adresou.

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    self.webView.delegate = nil;
    self.webView.delegate = self;
    [self loadRequestFromString:@"http://wvc.pf.jcu.cz/ki/?article=/katedra/
        konzultacni-hodiny.html&v=text"];
}
```

Obrázek 6.16: Metoda viewDidLoad

V metodě `loadRequestFromString` se URL adresa uloží pomocí metody `requestWithURL` do proměnné `urlRequest`, která je následně načtena pomocí metody `loadRequest`. Po načtení této metody se zavolá metoda delegáta `webViewDidStartLoad` a příslušný obsah se začne načítat.

```
- (void)loadRequestFromString:(NSString*)urlString
{
    NSURL *url = [NSURL URLWithString:urlString];
    NSURLRequest *urlRequest = [NSURLRequest requestWithURL:url];
    [self.webView loadRequest:urlRequest];
}
```

Obrázek 6.17: Metoda loadRequestFromString

V těle metody delegáta `webViewDidStartLoad` je nastaven příkaz, který indikuje načítání obsahu ve stavovém řádku zařízení.

```
- (void)webViewDidStartLoad:(UIWebView *)webView
{
    [UIApplication sharedApplication].networkActivityIndicatorVisible = YES;

    [self updateButtons];
}
```

Obrázek 6.18: Metoda webViewDidStartLoad

Metoda `webViewDidFinishLoad` ukončí indikaci ve stavovém řádku a následně se postará o načtení požadovaných elementů.

```
- (void)webViewDidFinishLoad:(UIWebView *)webView
{
    [UIApplication sharedApplication].networkActivityIndicatorVisible = NO
        ;
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.getElementById('box-right').style.display = 'show'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.getElementById('line-footer').style.display = 'none'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.getElementById('line-banner').style.display = 'none'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.getElementById('anotaceorange').style.display = 'none'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.getElementById('left-menu').style.display = 'none'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.getElementById('search-form').style.display = 'none'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.getElementById('box-left').style.display = 'none'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.getElementById('parse-link').style.display = 'none'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.body.style.backgroundColor = 'rgb(255, 236, 202)'"];
    [self.webView stringByEvaluatingJavaScriptFromString:
     @"document.body.style.fontFamily = 'Arial, Helvetica, sans-serif'"];

    [self updateButtons];
}
```

Obrázek 6.19: Metoda webViewDidFinishLoad

V případě, že uživatel není připojen k internetu, se zavolá další metoda delegáta `didFailLoadWithError`. Uživateli se při zavolání této metody zobrazí upozornění na problémy s připojením a nabádá ho k tomu, aby si v nastavení zkontroloval připojení k internetu.

```
- (void)webView:(UIWebView *)webView didFailLoadWithError:(NSError *)error
    {
    UIAlertView *alert = [[UIAlertView alloc]initWithTitle:
        NSLocalizedString(@"Chyba připojení", @"") message:
        NSLocalizedString(@"Nelze načíst obsah. Zkontroluje zdali jste
        připojení k internetu prostřednictvím WIFI nebo mobilního
        připojení.", @"") delegate:self cancelButtonTitle:@"OK"
        otherButtonTitles:nil];
    [alert show];
    alert.delegate = nil;
}
```

Obrázek 6.20: Metoda didFailLoadWithError

6.5 Budovy Pedagogické fakulty

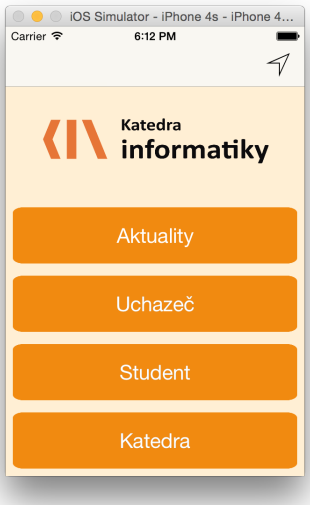
Do aplikace byla implementována možnost zobrazení budov Pedagogické fakulty. Tato funkce byla převzatá z dílčí aplikace využívající framework MapKit, která je podrobněji rozebrána v kapitole 5.2. Pro uchazeče o studium to může být přínosná funkce, díky které budou vědět, v jaké části města se jednotlivé budovy nachází.

6.6 Testování aplikace

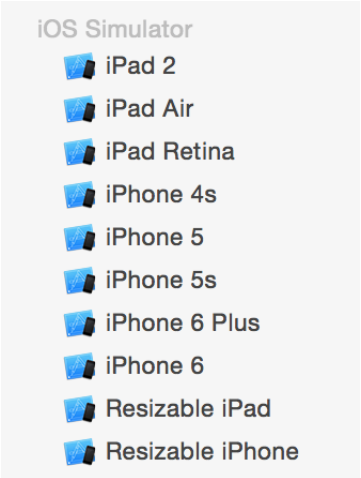
Ve chvíli, kdy je aplikace hotová, je potřeba ji před publikováním na App Store řádně otestovat za všech možných podmínek a na všech dostupných zařízeních.

6.6.1 Simulátor

První cestou, jak aplikaci vyzkoušet, je přímo ve vývojovém prostředí Xcode pomocí nástroje Simulátor. Ten dovoluje testovat aplikaci i v případě, že vývojář nemá zaplacenou licenci. Simulátor dovoluje otestovat aplikaci na všech podporovaných zařízeních a na obrázku 6.21 můžete vidět, jak v simulátoru vypadá spuštěná aplikace pro iPhone 4S. Při spuštění aplikace v simulátoru je možné zvolit, na jakém zařízení se má aplikace spustit, přesně jak můžete vidět na obrázku 6.22. Simulátor má ovšem určitá omezení. V simulátoru nelze například použít kameru zařízení a v případě, že budete chtít z fotogalerie nahrát do aplikace obrázek, tak se budete muset spokojit s předdefinovanými obrázky. Setkal jsem se i s tím, že se mi v aplikaci využívající frameworku Core Location nezobrazovala hodnota nadmořské výšky, která na fyzickém zařízení fungovala korektně.



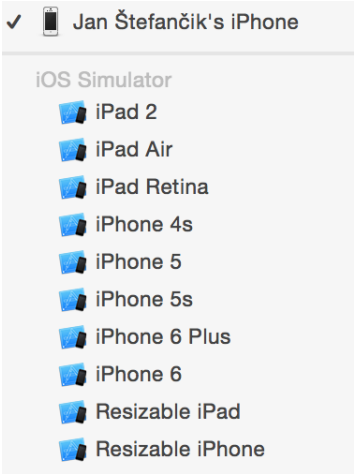
Obrázek 6.21: Testování v simulátoru



Obrázek 6.22: Zařízení v simulátoru

6.6.2 Fyzické zařízení

Simulátor je skvělým nástrojem pro testování zařízení, protože si málokdo může dovolit fyzicky vlastnit všechna iOS zařízení a testovat aplikaci na nich. Nicméně je nutné otestovat aplikaci i na reálném zařízení a tím vyzkoušet její funkčnost ve chvíli, kdy je například zařízení připojené k internetu pomocí WiFi, případně mobilní sítě. K tomu, aby se dala aplikace vyzkoušet na fyzickém zařízení, je třeba být ve vývojovém prostředí přihlášen pomocí vývojářského účtu a aplikaci nahrát do zařízení. K nahrání stačí iOS zařízení připojit k počítači a pří spuštění aplikace zvolit fyzické zařízení, jako je mozné vidět na obrázku 6.23.



Obrázek 6.23: Volba fyzického zařízení

6.6.3 Testování aplikace Katedry informatiky

Aplikace Katedry informatiky byla testována na všech zařízeních, které simulátor nabízí. Aplikaci jsem dále zkoušel na iPhonu 6, iPhonu 5, iPhonu 4s a iPadu mini 2. Aplikace se chovala jak v Simulátoru, tak na fyzických zařízeních stabilně a spolehlivě.

6.7 Publikování aplikace na App Store

Aby si mohl uživatel aplikaci stáhnout a využívat ji ve svém mobilním telefonu, musí být nejprve nahrána na App Store. K tomu, abychom nahráli aplikaci na App Store je nutné je nejdříve zaregistrovat ID aplikace, vytvořit popis aplikace, vygenerovat distribuční certifikát a následně vytvořit build, který se nahraje na App Store. [19]

6.7.1 App ID

Na stránkách `https://developer.apple.com` je nejdříve třeba přejít k položce Certifikáty, Identifikátory a Profily. Zde je třeba vybrat Identifikátory a tam vytořit id aplikace. Při vytvoření id aplikace je třeba vyplnit název aplikace a bundle id, který se musí shodovat s bundle id, který nalezneme v Xcode. [19]

6.7.2 iTunes Connect

Ve chvíli, kdy máme vytvořené id aplikace, přejdeme na další stránku určenou vývojářům `https://itunesconnect.apple.com/`, kde vytvoříme novou aplikaci. Zde kromě názvu aplikace a primárního jazyku vyplníme bundle id. Políčko pro bundle id nám jej samo nabídne v případě, že už je vytvořené. Poté přejdeme na další stránku, kde vyplňujeme další podrobnosti o aplikaci jako například: kategorie, klíčová slova pro vyhledávání, web aplikace a kontaktní informace. Je nutné vložit také snímky z aplikace, a to pro iPhone s 3.5, 4.0, 4.7 a 5.5 palcovým displejem. V neposlední řadě je třeba nahrát ikonu aplikace s rozlišením 1024 na 1024 bodů. [19]

6.7.3 Distribuční certifikát

Když máme hotový popis aplikace, musíme přejít k vytvoření distribučního certifikátu. Na stránkách `https://developer.apple.com` přejdeme stejně jako v prvním případě k položce Certifikáty, Identifikátory a Profily a zde v sekci profily vytvoříme distribuční certifikát. Při tvorbě distribučního certifikátu zvolíme id aplikace, které jsme předtím vytvořili. Následně vygenerujeme certifikát a na závěr ho stáhneme do počítače. Certifikát otevřeme a přejdeme do Xcode, kde aplikaci, kterou chceme nahrát na App Store archivujeme. Po archivování se objeví nové okno, kde se zobrazí aplikace i s potřebnými certifikáty a následně nám už jen stačí stisknout tlačítko odeslat na App Store. V případě, že je vše nastaveno správně, objeví se upozornění, že se odeslání zdařilo a aplikace byla tím pádem odeslána ke schválení.

7 Závěr

Má bakalářská práce se zabývá tvorbou mobilních aplikaci pro pro platformu iOS v Objective-C. Cílem práce bylo představit systém iOS vývojářům, kteří chtějí zjistit specifika a možnosti tvorby nativních aplikací, a prověřit tak možnosti vývoje pro mobilní zařízení s tímto operačním systémem. Práce je rozdělena na teoretickou a praktickou část.

V teoretické části čtenáře seznamuji s platformou iOS a jednotlivými verzemi tohoto systému. Dále popisuji, jaké požadavky je nutné splnit v případě, že se rozhodne vyvíjet aplikace pro tuto platformu. Konkrétně jáké jsou minimální hardwarové požadavky, popis vývojové prostředí Xcode a možnost získání vývojářské licence. V práci dále popisuji architekturu iOS a s ní spojené vrstvy systému. Konkrétně vysokoúrovňové služby a frameworky dostupné v jednotlivých vrstvách.

V praktické části se zabývám vývojem mobilních aplikací. V úvodu praktické části jsem vytvořil dílčí aplikace, na kterých jsem demonstroval práci s vybranými frameworky. Jedná se o aplikace využívající frameworky Core Location, Core Image a MapKit.

V druhé části praktické části se zabývám tvorbou nativní aplikace pro Katedru informatiky Pedagogické fakulty Jihočeské univerzity v Českých Budějovicích, která byla hlavním cílem mé bakalářské práce a má sloužit jako rozcestník pro studenty a uchazeče o studium. V této části popisuji, jaké možnosti mají v současné chvíli studenti a uchazeči k přístupu informací ze stránek katedry na iOS zařízeních. Dále popisuji, jak jsem postupoval při návrhu aplikace, naleznete zde ukázky zdrojových kódu, popisuji, jak jsem aplikaci testoval v simulátoru a na fyzickém zařízení. Na závěr jsem popsal postup, jak výslednou aplikaci nahrát na App Store.

Cíle bakalářské práce byly splněny.

Seznam použité literatury

[1] BUSTER, Hein. *The Evolution Of iOS: From iPhone OS To iOS 7.* [online]. [cit. 2015-02-12] Dostupné z: http://www.cultofmac.com/191340/the-evolution-of-ios-from-iphone-os-to-ios-6-gallery/

[2] *iOS: A visual history.* [online]. [cit. 2015-02-12] Dostupné z: http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad

[3] *iOS Dev Center.* [online]. [cit. 2015-03-16] Dostupné z: https://developer.apple.com/support/ios/ios-dev-center.php

[4] *OS X Mavericks system requirements.* [online]. [cit. 2015-03-16] Dostupné z: https://support.apple.com/en-us/HT201364

[5] KOCHAN, Stephen G. *Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone*, Vyd. 1. Brno: Computer Press, 2010, 550 s. ISBN 978-80-251-2654-7.

[6] SHARP, Maurice, Rod STROUGO a Erica SADUN. *Learning iOS development: a hands-on guide to the fundamentals of iOS programming.*, Vyd. 1. New Jersey: Pearson Education, Inc., 2013, 556 s. ISBN 978-0-321-86296-9.

https://developer.apple.com/xcode/

[7] *Xcode the complete toolset for building great apps.* [online]. [cit. 2015-03-17] Dostupné z: https://developer.apple.com/xcode/

[8] *iOS Developer Program.* [online]. [cit. 2015-03-18] Dostupné z: https://developer.apple.com/programs/ios/

[9] APPLE INC. *IOS Technology Overview* [online]. [cit. 2015-03-24]. Dostupné z: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iOSTechOverview.pdf

[10] *Cocoa Touch Layer.* [online]. [cit. 2015-04-01] Dostupné z: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html

[11] *Media Layer.* [online]. [cit. 2015-04-03] Dostupné z: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html

[12] *Cocoa Services Layer.* [online]. [cit. 2015-04-04] Dostupné z: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.htm

[13] *Cocoa OS Layer.* [online]. [cit. 2015-04-07] Dostupné z: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html

[14] *Understanding the Global Positioning System (GPS).* [online]. [cit. 2015-04-12] Dostupné z: http://www.montana.edu/gps/understd.html

[15] *Jak určit polohu mobilního telefonu.* [online]. [cit. 2015-04-12] Dostupné z: http://www.mobilmania.cz/jak-urcit-polohu-mobilniho-telefonu/a-1107567/

[16] *Wi-Fi Positioning System.* [online]. [cit. 2015-04-12] Dostupné z: http://gps.about.com/od/glossary/g/wifi_position.htm

[17] *Beacons: Everything you need to know.* [online]. [cit. 2015-04-12] Dostupné z: http://blog.pointrlabs.com/beacons-everything-you-need-to-know/

[18] MARK, Dave a Jeff LAMARCHE. *IPhone SDK: průvodce vývojem aplikací pro iPhone a iPod touch*, Vyd. 1. Brno: Computer Press, 2010, 480 s. ISBN 978-80-251-2820-6.

[19] *Submitting Your App to the Store.* [online]. [cit. 2015-04-13] Dostupné z: https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/SubmittingYourApp/SubmittingYourApp.html

Seznam obrázků

Přílohy

1. CD - na přiloženém CD se nachází plné znění bakalářské práce pod názvem `stefancik_bp.pdf`, dále se na CD nachází zdrojové soubory k dílčím aplikacím a k aplikaci Katedry informatiky.