

**Česká zemědělská univerzita v Praze**  
**Provozně ekonomická fakulta**  
**Katedra informačních technologií**



**Bakalářská práce**

**Vývoj mobilní aplikace pro platformu iOS**

**David Aldorf**

### **Čestné prohlášení**

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

V Praze dne 14.3.2016

---

### **Poděkování**

Rád bych touto cestou poděkoval panu Ing. Pavlu Šimkovi, Ph.D. za vedení bakalářské práce.

# Vývoj mobilní aplikace pro platformu iOS

## Souhrn

Tato bakalářská práce se zabývá vývojem mobilní aplikace pro operační systém iOS. Vytvoření aplikace bylo provedeno na základě teoretických poznatků získaných v teoretické části bakalářské práce. Bylo provedena komparace vývojových prostředí a na základě bodovací metody bylo vybráno vývojové prostředí Xcode IDE od společnosti Apple. Poté byl vytvořen logický návrh, který obsahuje rozvržení a popisuje funkcionalitu aplikace. Na základě logického návrhu byl vytvořen grafický návrh aplikace se vzhledem opírajícím se o Material design od společnosti Google. K implementaci kódu byl použit moderní objektově orientovaný programovací jazyk Swift.

Aplikace poskytuje uživateli možnost přihlášení přes sociální síť Facebook, a nebo přes email. díky tomu je zaručena úschova dat v databázi a uživatel ke svým datům může přistoupit i z jiných zařízení iOS. Pro databázové řešení byl použit Framework FireBase. Aplikace z databáze generuje seznam receptů pro zdravý životní styl a ukládá informace o uživateli pro výpočet ideálního jídla.

**Klíčová slova:** iOS, FireBase, Xcode, Mobilní aplikace, Vývoj, databáze, Swift, Material design, Facebook

# Development of mobile application for the iOS platform

## Summary

This Bachelor's thesis deal with the development of mobile applications for iOS operating system. Creating of application was based on theoretical knowledge gained in the theoretic part of the bachelor's thesis. It was made a comparison of development environment based on chosen scoring method. The Chosen development environment was Xcode IDE developer by Apple Inc. Then it was create a logical design that contains the layout and describes the functionality of the application. Based on logical design was created graphic design based on Material design by Google.

For the implementation of code was used modern object-oriented programming language Swift. Application allows users login via social network Facebook, or via email. It ensures data storage in database FireBase and the users can access their data from other iOS devices. Application generate list of healthy life recipes from database and then keep user's informations for calculate the ideal day meal.

**Keywords:** iOS, development, mobile applications, FireBase, Facebook, Material design, database, Swift, Xcode

# Obsah

<b>1 Úvod .....</b>	<b>1</b>
<b>2 Cíl práce a metodika .....</b>	<b>2</b>
2.1 Cíl práce .....	2
2.2 Metodika .....	2
<b>3 Teoretická východiska .....</b>	<b>4</b>
3.1 Literární rešerše.....	4
3.2 Co je iOS.....	5
3.3 Historický vývoj iOS.....	5
3.3.1 iPhone OS 1 .....	5
3.3.2 iPhone OS 2 .....	5
3.3.3 iPhone OS 3 .....	5
3.3.4 iOS 4 .....	6
3.3.5 iOS 5 .....	6
3.3.6 iOS 6 .....	6
3.3.7 iOS 7 .....	6
3.3.8 iOS 8 .....	7
3.3.9 iOS 9 .....	7
3.4 Výhody iOS oproti konkurenci .....	7
3.4.1 Aktualizace systému.....	8
3.4.2 Odladění systému a aplikací.....	9
3.4.3 Bezpečnost .....	10
3.4.4 Aplikace .....	10
3.5 Architektura .....	11
3.6 iOS SDK .....	12
3.7 iOS Frameworky .....	12
3.8 Developer library .....	12
3.9 Cocoa Touch .....	13
3.9.1 App Extensions .....	13
3.9.2 HandOff .....	13
3.9.3 Document Picker.....	13
3.9.4 AirDrop .....	13
3.9.5 TextKit .....	14
3.9.6 UIKit Dynamics .....	14

3.9.7	Multitasking .....	14
3.10	Media vrstva.....	14
3.10.1	Grafické technologie .....	14
3.10.2	Technologie pro zvuk.....	15
3.10.3	Technologie pro video.....	15
3.11	Core services vrstva .....	15
3.11.1	Uložiště iCloud.....	15
3.11.2	Block Objects.....	16
3.11.3	Ochrana Dat .....	16
3.11.4	Nákup In-App.....	16
3.11.5	SQLite .....	16
3.12	Core services Framework.....	17
3.12.1	Accounts Framework .....	17
3.12.2	Adressbook Framework .....	17
3.12.3	Ad support Framework .....	17
3.12.4	CFNetwork Framework .....	17
3.12.5	CloudKit Framework .....	17
3.12.6	CoreData Framework .....	18
3.12.7	Core Foundation Framework .....	18
3.12.8	Core Location Framework .....	18
3.12.9	CoreMotion Framework.....	18
3.13	Core OS vrstva.....	19
3.13.1	Accelerate Framework .....	19
3.13.2	CoreBluetooth Framework.....	19
3.13.3	External Accessory Framework .....	19
3.13.4	Local Authentication Framework.....	19
3.13.5	Network Extension Framework .....	20
3.13.6	Security Framework .....	20
3.13.7	Systém.....	20
3.13.8	64-Bit Support.....	20
3.14	Vývoj pro platformu iOS .....	20
3.14.1	Nativní vývoj iOS .....	21
3.14.2	Xcode .....	21
3.14.3	Objective-C .....	21
3.14.4	Swift.....	22
3.15	Xamarin studio.....	22
3.16	Jazyk C#.....	23
3.17	Co je nového .....	24

3.17.1	Zdraví .....	24
3.17.2	Kontinuita.....	24
3.17.3	Rodinné sdílení.....	24
3.17.4	FireBase .....	24
<b>4</b>	<b>Vlastní práce.....</b>	<b>26</b>
4.1	Logický návrh aplikace .....	26
4.1.1	Příprava .....	26
4.1.2	Návrh drátěných modelů .....	26
4.1.3	První view – Přihlášení a registrace .....	27
4.1.4	Druhý view – Recepty.....	27
4.1.5	Třetí view - Osobní údaje.....	27
4.2	Grafický návrh aplikace .....	28
4.3	Komparace vývojových prostředí .....	28
4.3.1	Cena .....	28
4.3.2	Náročnost vývoje .....	29
4.3.3	Podpora .....	30
4.3.4	Vzhled a funkčnost.....	30
4.3.5	Doba vývoje .....	31
4.3.6	Vývojové prostředí.....	32
4.3.6.1	PhoneGap .....	32
4.3.6.2	Xcode .....	32
4.3.6.3	Xamarin Studio .....	33
4.3.7	Shrnutí komparace vývojových prostředí .....	33
4.4	Seznámení s IDE Xcode.....	34
4.5	Uživatelské rozhraní.....	35
4.5.1	První strana.....	35
4.5.2	Druhá strana .....	36
4.5.3	Detail receptu .....	36
4.5.4	Třetí strana .....	36
4.5.5	Backend – server .....	37
4.6	Implementace kódu .....	38
4.6.1	Přihlášení pomocí Facebook .....	38
4.6.2	Přihlášení pomocí emailu .....	39
4.6.3	Recepty.....	39
4.6.4	Alamofire framework.....	40
4.6.5	Osobní údaje.....	40



<b>5</b>	<b>Zhodnocení výsledků .....</b>	<b>42</b>
5.1	Logický návrh aplikace .....	42
5.2	Grafický návrh aplikace .....	42
5.3	Komparace vývojových prostředí .....	43
5.4	Uživatelské rozhraní.....	43
5.5	Implementace kódu .....	43
5.6	Shrnutí a porovnání s alternativami .....	44
<b>6</b>	<b>Závěr .....</b>	<b>46</b>
<b>7</b>	<b>Seznam použitých zdrojů .....</b>	<b>47</b>
<b>8</b>	<b>Seznam obrázků.....</b>	<b>50</b>
<b>9</b>	<b>Seznam tabulek.....</b>	<b>50</b>
<b>10</b>	<b>Přílohy .....</b>	<b>51</b>

# 1 Úvod

Časy mobilních telefonů , které poskytovaly pouze možnost volání a psaní SMS zpráv jsou pryč. Nároky lidí na techniku se postupně zvyšují a společně s nimi se i mobilní telefony vyvíjí. V dnešní době jsou mobilní telefony na velkém technologickém vzestupu a stávají se nedílnou součástí života velké části populace. Ať si to připouštíme nebo ne, jsou pro nás velkým usnadněním běžného života.

Aplikace v chytrých telefonech nám pomohou v komunikaci s lidmi na delší vzdálenost v podstatě kdekoliv, kde máme možnost mobilního internetu, či wifi. Samozřejmě je mnoho aplikací a funkcí chytrých telefonů, které se využívají v offline módu, tzn. bez připojení na internet. Velkou výhodou chytrých telefonů, jsou jejich kompaktní rozměry, při kterých si v této době ponechávají výpočetní výkon téměř na úrovni stolních počítačů, či notebooků. Další využití chytrých telefonů se nachází ve využití fotoaparátu - zařízení, která v této době téměř nahradila klasické fotoaparáty běžných uživatelů a umožňuje zachytit moment, bez toho aniž by bylo potřeba s sebou nosit navíc jedno zařízení.

Mimo mobilní aplikace a chytrá zařízení se zajímám o dnes velice populární zdravý životní styl a sport, především posilování. Díky tomu mě napadla myšlenka vytvořit aplikaci, která by mi pomohla řešit každodenní problémy spojené s plánováním stravy a tréninků obecně. Samozřejmě máme k dispozici nepřeberné množství aplikací z fitness světa a receptů, ale pro začínajícího jedince je velice obtížné sestavit si jídelníček s ohledem na jeho cíle, povětšinou zhubnutí nebo přibírání svalové hmoty.

A právě to je cíl této bakalářské práce. Vytvoření aplikace, která by pomohla navrhnout ideální jídlo během dne, s ohledem na fyzickou aktivitu daného dne. Také by měla samozřejmě sledovat postup v cíli, který si stanovíte.

## **2 Cíl práce a metodika**

### **2.1 Cíl práce**

Bakalářská práce je tematicky zaměřena na problematiku vývoje aplikace pro platformu iOS. Hlavním cílem práce je návrh, vyvinutí a otestování reálné aplikace. Dílčí cíle práce jsou: Sumarizace historického vývoje mobilních platform s důrazem na platformu iOS, komparace vývojových prostředí, komparace vyvinuté aplikace s možnými alternativními aplikacemi.

### **2.2 Metodika**

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Z poznatků založených na teoretických východiscích bakalářské práce byla provedena komparace vývojových prostředí. Komparace vývojových prostředí byla provedena na základě ohodnocení zvolených kritérií každého z vývojových prostředí, která mají přímý vliv na vývoj aplikace samotné.

Při sestavení logického návrhu aplikace byla použita metoda User experience, ve které byly vytvořeny drátěné modely aplikace pro každou její stranu. Tyto modely byly otestovány za pomoci sedmy zástupců cílové skupiny. Na základě poznatků získaných z testování byl vytvořen konečný logický návrh, ze kterého byl následně vytvořen návrh grafický. Jeho tvorba byla řízena souborem pravidel pro tvorbu mobilní aplikace v Material designu.

Pro vytvoření uživatelského rozhraní bylo použito vývojové prostředí Xcode, pomocí metod UIKit bylo vytvořeno uživatelské rozhraní na základě grafického návrhu. Pro uživatelské rozhraní byly vytvořeny třídy pro grafickou úpravu jednotlivých prvků na vzhled Material designu. Implementace kódu pro uživatelské rozhraní je rozdělena do ViewControlleru aplikace, neboli hlavních stran aplikace, které obsahují danou funkcionalitu.

Pro první stranu aplikace určenou pro přihlašování a registraci uživatelů do databáze byly využity frameworky FireBase a Facebook SDK. Pro přístup k databázi FireBase byly použity metodické postupy založené na využití Firebase framework, který zajišťuje zapsání a čtení obsahu v databázi. Dále byla vytvořena aplikace pro přístup k Facebooku SDK na stránce developers.facebook.com, pomocí které je zajištěný přístup aplikace k API Facebooku. V kódu aplikace byla vytvořena funkcionalita pro přihlášení uživatele a jeho zapsání do databáze. Pro druhou stranu aplikace byla

použita metoda pro zobrazení dat z databáze, kterými byly naplněny jednotlivé buňky TableViewControlleru. Pro získání obrázků z databáze FireBase byl použit Alamofire Framework, pomocí kterého byly obrázky v aplikaci zobrazeny a upraveny na určenou velikost. Na základě syntézy teoretických poznatků a výsledků praktické části byly formulovány závěry bakalářské práce.

## 3 Teoretická východiska

### 3.1 Literární rešerše

V teoretické části této bakalářské práce byli informace o operačním systému iOS a vývoji mobilních aplikací pro operační systém iOS čerpány především z oficiální dokumentace operačního systému iOS [1] a vývojového prostředí IDE Xcode [2], společnosti Apple a dále knihy vydané společností Apple The Swift programming language. [3]

Z knihy The Swift programming language, bylo čerpáno především v praktické části bakalářské práce, díky ukázkovému kódu přímo od společnosti Apple za pomoci iOS Developer Library [2]. Kniha se zabývá výukou programování v jazyce Swift, který byl použit pro vytvoření aplikace v této bakalářské práci. V knize je nastíněna problematika vývoje od deklarace proměnných, až po práci s daty aplikace. Většina použitých materiálů v této bakalářské práci je v angličtině, je to z důvodu poněkud mladé komunity vývojářů iOS v České republice a také z důvodu zvoleného nativního přístupu vývoje aplikace pro operační systém iOS za použití poměrně mladého programovacího jazyka Swift. Pomoc při vytváření grafického návrhu aplikace poskytla dokumentace společnosti Google Material design [4]. Dokumentace poskytuje názorné ukázky a příklady pro vytvoření designu aplikace. Dokumentace Material design je vytvořena předními designéry Google.

Dílčí části bakalářské práce čerpají informace především z internetových magazínů a oficiálních dokumentací společností, které byli v bakalářské práci zmíněny, např. PhoneGap, Alamofire, Xamarin IDE apod.

## **3.2 Co je iOS**

iOS je operační systém, který je základem mobilních zařízení společnosti Apple. Mezi zařízení podporující iOS patří iPad, iPhone a iPod. Operační systém spravuje Hardware zařízení a poskytuje technologie pro implementaci nativních aplikací. iOS je mobilní verzi operačního systému OS X určené pro desktopové zařízení iMac a MacBook společnosti Apple. Operační systém iOS je na zařízení dodáván společně s různými systémovými aplikacemi, jako je například aplikace telefon, mail a safari, které poskytují standartní systémové služby pro uživatele. [1] [2]

## **3.3 Historický vývoj iOS**

### **3.3.1 iPhone OS 1**

V roce 2007 byla zveřejněna první verze operačního systému od společnosti Apple Inc. Operační systém byl v tu dobu určen pouze pro první generaci mobilního telefonu iPhone. Do roku 2008, kdy byl vydán iPhone software development kit, nebyl oficiálně pojmenován. Od tohoto roku nesl operační systém název iPhone OS. V první verzi nebylo možné instalovat aplikace, ale i přesto společnost Apple Inc. udala jasný směr v uživatelském prostředí, kterým se výrobci mobilních zařízení v dalších letech dali.

### **3.3.2 iPhone OS 2**

V roce 2008 společně s vydáním iPhone 3G, přichází druhá generace iPhone OS. Tato generace přichází s revolucí v aplikacích a tím se stal konkrétně App Store. Uživatelé již nebyli nuceni instalovat pomocí svého počítače a hledat je z různých internetových zdrojů. Protože Apple vymyslel centrální místo, odkud byly aplikace dostupné všem uživatelům přímo z jejich mobilních zařízení.

### **3.3.3 iPhone OS 3**

O rok později přichází společnost s další verzí iPhone OS 3. Ve třetí verzi jsou přidány pouze některé funkce, navíc např. centrum notifikací, funkce pro kopírování a vkládání textu a aplikaci Find my iPhone (bezpečností aplikaci pro nalezení ztraceného či odcizeného iPhone).

### **3.3.4 iOS 4**

V roce 2010 byla zveřejněna 4. generace operačního systému, poprvé nesoucí název iOS. Tato verze jako první nepodporuje všechna zařízení, konkrétně iPhone první generace. Nová verze obsahuje multitasking, tj. práce s více spuštěnými aplikacemi najednou, provozování videohovorů mezi zařízeními iOS, za pomoci aplikace FaceTime, vytváření složek s aplikacemi, vzdálené přehrávání multimédií AirPlay a vzdálený tisk pomocí AirPrint.

### **3.3.5 iOS 5**

V roce 2011 byla zveřejněna 5. generace operačního systému iOS, která s sebou přináší hlasového klienta Siri - aplikaci rozpoznávající dotazy ,na základě nichž vyhledává požadované informace. Dále poskytuje službu iMessage, textová komunikace mezi zařízeními iOS a cloudové úložiště iCloud, pro zálohování zařízení.

### **3.3.6 iOS 6**

V roce 2012 společnost Apple uvedla 6. generaci iOS. Přináší novou aplikaci pro mapy, která umožňuje hlasovou navigaci na pozadí, předinstalovanou sociální síť Facebook a funkci pro sdílení fotografií.

### **3.3.7 iOS 7**

V roce 2013 byla vydána 7. generace iOS. Největší devizou byl vzhled systému, ve kterém jsou použity bílé odstíny barev, průhledné efekty, nový vzhled ikon aplikací a ovládací centrum.

Rozšíření se týká také notificačního centra, které místo jedné záložky obsahuje tři záložky a aplikaci AirDrop, která slouží pro sdílení obsahu mezi iOS zařízeními.

### **3.3.8 iOS 8**

V roce 2014 Apple představil iOS 8. generace. Zařízení podporující iOS 8 jsou iPhone 4S, iPad 2, iPod Touch (5. generace) a novější. iOS 8 přinesla velké změny převážně pro vývojáře aplikací. Bylo zpřístupněno velké množství API a byl představen programovací jazyk Swift. V nové verzi systému je použito provázání aplikací. Kontinuita zařízení, tj. propojení zařízení iOS, OSX a jejich funkcí. Jsou přidány také interaktivní notifikace a další rozšíření aplikací iOS.

### **3.3.9 iOS 9**

V roce 2015 Apple představil iOS 9. Jedná se o aktuální verzi systému od společnosti Apple. iOS 9 přináší plnohodnotný multitasking, tj. SlideOver pro iPady, uložiště iCloud Drive, aplikace pro nahrávání dokumentů, videí, fotografií a jejich sdílení a je upravena aplikace pro poznámky. Inovace se týkají dále aplikací News, Siri a byl přidán úsporný režim pro delší výdrž baterie zařízení.

[5]

## **3.4 Výhody iOS oproti konkurenci**

V této kapitole představují konkurenci společnosti Google a Microsoft a jejich mobilní platformy (operační systémy) Android a Windows Phone (Windows Mobile).

Viz tabulka č.1



Tabulka 1 - Počet prodaných kusů

Systém / počet kusů (mil.)	4Q 2013	2013	4Q 2014	2014
Android	227,3	780,8	291,7	1 042,70
iOS	51	153,4	74,5	192,7
Windows Phone	9,6	35,8	11,5	38,8
Ostatní	2,3	20	2,6	9,3
<b>Celkem</b>	290,2	990	380,1	1 283,50

Zdroj: [6]

Operační systém iOS zabírá 2. místo na světovém trhu v počtu prodaných mobilních zařízení. Jeho ztráta oproti androidu je způsobena několika důvody. Tím prvním a největším důvodem nižšího počtu mobilních zařízení je především cena zařízení od společnosti Apple. Průměrná cena prodaného iPhone je 687 dolarů což činí v přepočtu okolo 16 900 Kč. Průměrná cena mobilního zařízení Android je 254 dolarů v přepočtu okolo 6 300 Kč. [6][7][8]

### 3.4.1 Aktualizace systému

Google kromě zařízení z vlastní produkce, ponechává kontrolu nad aktualizacemi výrobcům a mobilním operátorům. Výsledkem toho je, že uživatelé starších zařízení čekají dlouhé měsíce, než budou moci nahrát do svých smartphonů a tabletů novou verzi Androidu. Dle statistik na nejnovější verzi operačního systému Android s názvem Marshmallow bylo aktualizováno zhruba 1,5 %, na předcházející verzi s označením Lollipop bylo aktualizováno zhruba 35 %.

Naopak největší výhodou iOS jsou aktualizace. Apple si vytváří vlastní software pro vlastní zařízení a každá nová verze má podporu pro všechny modely, s výjimkou již několik let starého iPhone 4, iPhone 3G a původního iPhone. Po vydání nového systému je dostupný k aktualizaci.

V lednu 2016 byla nejnovější verze systému iOS s pořadovým číslem 9 aktualizována na zhruba 75% mobilních zařízení společnosti Apple.

Podobně jako Android je na tom operační systém Windows Phone (Windows 10 mobile). Windows Phone je operační systém společnosti Microsoft. Na nejnovější verzi s označením Windows 10 mobile bylo aktualizováno v listopadu 2015 zhruba 7% zařízení.

[9][10]

### **3.4.2 Odladění systému a aplikací**

Existuje mnoho androidů s různou velikostí, rozlišení displeje a různým hardwarem či softwarem v podobě upravených ROM (tzn. nadstavby operačního systému od výrobce mobilního zařízení). V tomto případě mají developéři aplikací velmi náročný úkol. Pokud by mělo být 100% zabezpečení her nebo aplikací, potřebovali by tisíce testovacích zařízení a potom otestovat nový software na téměř všech verzích systému Android. To samé platí i pro operační systém, díky rozmanitosti hardwaru a ROM, Android není na všech zařízeních odladěn co se týče plynulosti systému, tak jako jeho konkurence v podobě iOS a nebo Windows.

U iOS tato situace nenastává díky menšímu portfoliu zařízení Applu, aplikace se tak mohou testovat na iPhonech, iPodech a iPadech a tím dochází k lepšímu odladění i zabezpečení aplikací. U iOS je díky tomu zaručena větší kompatibilita s HW a systém je optimalizovanější. Přes pomalejší procesory, než jsou použity v zařízeních android, se systém projevuje jako plynulejší. Napomáhá zde například také multitasking (správa více aplikací najednou), jenž využívá méně paměti RAM, oproti konkurenci v podobě Androidu.

V případě Windows phone se odladěnost systému odráží na počtu zařízení. Microsoft situaci řeší obdobným přístupem jako Apple. Není zde vysoký nárok na HW, tzn. a při pomalejším procesoru a menší paměti RAM systém zaručí větší plynulost systému než u platformy Android.

[9]

### 3.4.3 Bezpečnost

Android od společnosti Google je velmi otevřený systém, kde největší hrozbou jsou útoky aplikací 3. stran. Uživatelé Androidu mohou aplikace stahovat a následně instalovat z neoficiálních zdrojů, kde nemají zaručenou bezpečnost svého zařízení.

Majitelé iOS zařízení jsou limitováni stahováním aplikací pouze přes AppStore, centrální uložisko aplikací pro Apple zařízení. Všechny nové aplikace od vývojářů jsou podrobeny precizní kontrole a vyžadují schválení od společnosti Apple. Výjimkou je neoficiální cesta pro nákup aplikací pomocí Jailbrake.

Windows phone poskytuje obdobnou ochranu při stahování aplikací a to prostřednictvím Store. Využití je stejné jako u iOS. Aplikace jsou kontrolovány a následně schvalovány společností Microsoft.

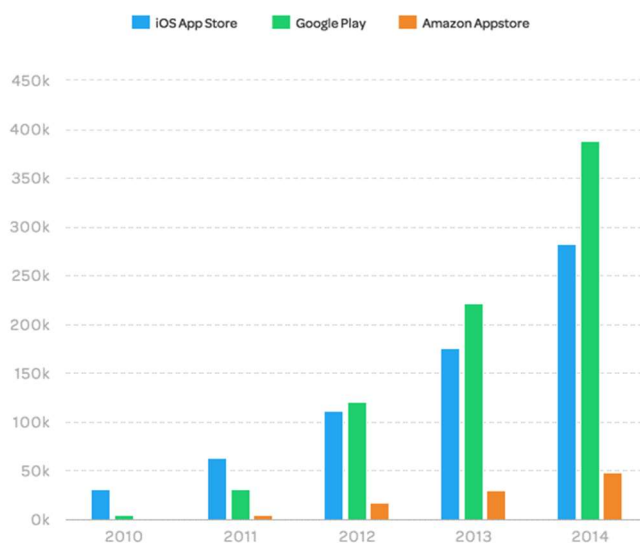
[9]

### 3.4.4 Aplikace

Statistiky analytické společnosti App Annie ukazují, že AppStore generuje až o 85% vyšší měsíční zisky než Google Play a to i přes jeho převahu v počtu aktivních zařízení viz graf č. 1.

Graf 1 Počet vývojářů mezi obchody s aplikacemi

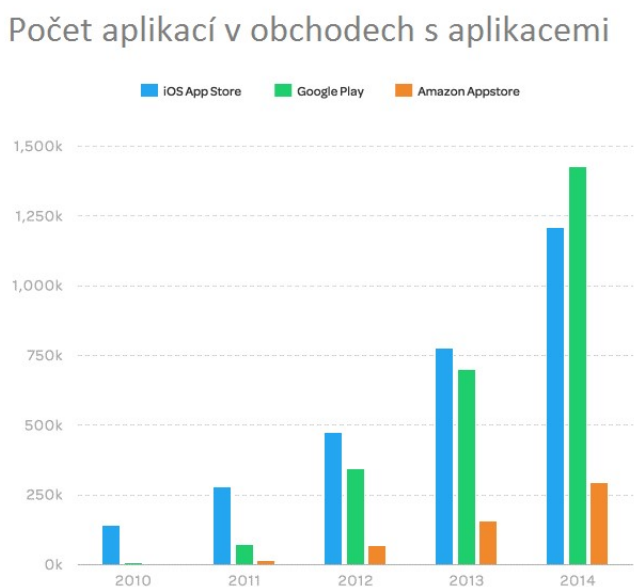
#### Počet vývojářů pro obchody s aplikacemi



Zdroj: [11]

Developeři tak mají větší finanční prostředky na vývoj, čemuž odpovídá i zmiňovaná kvalita samotných aplikací a zájem o vývoj pro platformu iOS viz graf č. 2. AppStore obsahuje přes 1,5 milionu aplikací. Výhoda iOS je že Apple vyvíjí Hardware a operační systém najednou, oproti Androidu, kde Google vytváří software a výrobci mobilů hardware. Z toho plyne, že se buď musí dodatečně přizpůsobit Sw na Hw nebo obráceně. Ještě horší stav je u aktualizací na rok starém zařízení.

Graf 2 - Počet aplikací v obchodech s aplikacemi

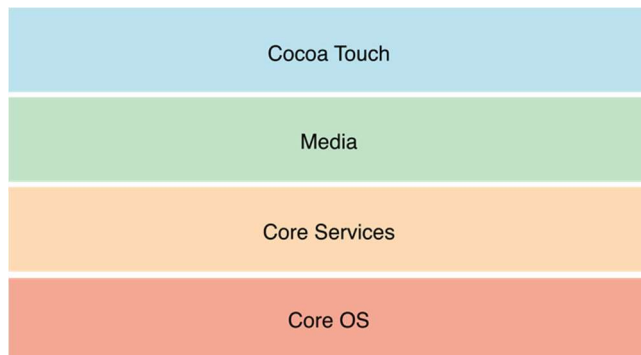


Zdroj: [11]

### 3.5 Architektura

iOS je odlehčenou verzí operačního systému OS X, který je používán u desktopových a přenosných počítačů MAC. IOS je operační systém UNIXového typu určený pro mobilní zařízení, proto není potřeba, aby obsahoval veškerou funkcionalitu OS X. Na druhou stranu, je zde oproti OS X přidána podpora dotykového ovládání. Operační systém iOS se dělí na čtyři základní vrstvy Cocoa Touch, Media, Core Services a Core OS. Každá z vrstev obsahuje balíček frameworků a rozhraní, pomocí kterých mohou být při vývoji aplikace použity.

Obrázek 2 - Základní vrstvy iOS



Zdroj: [2]

### 3.6 iOS SDK

IOS Software Development Kit (dále jen iOS SDK) obsahuje nástroje a rozhraní potřebné pro vývoj, instalaci, spuštění a testování nativní aplikace, které jsou zprostředkovány uživateli na domovské obrazovce zařízení. Nativní aplikace jsou vytvořeny za použití iOS systémových frameworků a programovacího jazyku Objective-C, aplikace jsou spustitelné přímo na iOS. Nativní aplikace jsou na rozdíl od aplikací webových, nainstalovány přímo na zařízení a proto je jejich obsah dostupný i v režimu Offline, tzn. bez přístupu k internetu. Nativní aplikace jsou umístěny vedle dalších systémových aplikací a jejich uživatelská data jsou synchronizována do desktopového zařízení pomocí iTunes. [2]

### 3.7 iOS Frameworky

Frameworky jsou speciální balíčky pomocí kterých Apple, poskytuje většinu svých systémových rozhraní. Framework je adresář, který obsahuje dynamicky sdílené knihovny a zdroje, např. hlavičkové soubory, obrázky a pomocné aplikace, potřebné k podpoře knihovny.

### 3.8 Developer library

Developer library iOS je důležitým zdrojem informací k použití během vývoje aplikací. Obsahuje reference všech systémových API, programovací příručky, technické poznámky, ukázkový kód a mnoho dalších zdrojů nabízejících tipy a rady při vytváření aplikace přímo od vývojářů společnosti Apple.

## 3.9 Cocoa Touch

Vrstva Cocoa Touch je Framework (knihovna) uživatelského rozhraní od společnosti Apple pro platformu iOS. Je založen na systému OS X jako mobilní verze Cocoa. Uživatelské prostředí, využívá vysokých programovacích jazyků pro nativní vývoj tzn. Objective-C a nově také Swift. Použití Cocoa Touch frameworku v první řadě přináší zjednodušení implementovaného kódu. Pomocí uživatelského rozhraní je možné přistupovat k funkcím iOS jako je např. Multitasking, jenž zabezpečuje práci a přepínání s dvěma a více spuštěných aplikací.

### 3.9.1 App Extensions

Operační systém iOS poskytuje rozšíření vybrané oblasti systému s použitím předem nakonfigurovaných App Extensions. App Extensions podporují sdílení obsahu se sociálními sítěmi např. Facebook a Twitter, provést jednoduchý úkol s aktuálním obsahem, poskytnout rychlou aktualizaci a zobrazení úkolu v oznamovacím centru zařízení, provádět úpravu fotografií a videa v aplikaci Fotky a poskytují místo pro ukládání dokumentů, pro které je vytvořen přístup i z ostatních aplikací.

### 3.9.2 HandOff

HandOff je funkce v operačních systémech OS X a iOS. Umožňuje uživatelům zahájit činnost v aplikaci na jednom zařízení od společnosti Apple a dokončit je na druhém. Při předávání aktivit se přiřazují aplikacím API v oblasti Foundation. Aktivita musí být reprezentována v aplikaci jako objekt uživatelské činnosti (User activity object).

### 3.9.3 Document Picker

Document Picker, neboli document picker controller, poskytuje uživatelům přístup k souborům mimo sandbox aplikace. Jedná se o sdílení dokumentů mezi aplikacemi. Vývojáři třetích stran mohou poskytnout dokumenty ke sdílení pomocí Storage Provider

### 3.9.4 AirDrop

AirDrop umožňuje uživatelům sdílet fotografie, dokumenty a další data s blízkými zařízeními iOS.

### 3.9.5 TextKit

TextKit je sada tříd pro manipulaci s textem a typografií. Požití TextKit poskytne aplikaci rozvržení textu do odstavců, sloupců, nebo obtékání grafického objektu textem. TextKit je integrován se všemi textovými ovládacími prvky UIKit.

### 3.9.6 UIKit Dynamics

UIKit Dynamics určí aplikaci dynamické chování pro objekty, které jsou podporovány protokolem UIDynamicItem, tyto objekty se nazývají dynamické položky. Dynamické položky navodí uživateli příjemnější požití z používání aplikace. Objekty se stanou dynamicky aktivní, po přidání UIDynamicAnimator třídy.

### 3.9.7 Multitasking

Multitasking poskytuje přepínání mezi aplikacemi, při kterém dochází ke zneaktivnění první aplikace a následně zaktivnění aplikace druhé. Při zneaktivnění, aplikaci je umožněno požádat o konečné množství času ke zpracování úkolu, např. přehrávání hudby. [12]

## 3.10 Media vrstva

Umožňuje vytvářet graficky a zvukově propracované aplikace a také plynulé přehrávání animací, zvuků a videí

### 3.10.1 Grafické technologie

Ve většině případů lze využít grafické rozhraní definované samotným systémem. Pro ostatní případy je možné použít následující technologie.

- Core graphics (Quartz) - 2D vektory a renderované obrázky
- Core animation - Pokročilé animace obrázků
- OpenGL ES - HW akcelerované vykreslování 2D/3D obrázků
- Core Text - Sofistikovaný engine pro vykreslování textu
- Image I/O - Čtení a zápis grafických formátů
- The Assets library framework - Přístup k obrázkové knihovně uživatele

### **3.10.2 Technologie pro zvuk**

Umožňuje přehrávání videozáznamů a používání vibrací. Podporuje následující frameworky.

- The Media Player framework - Umožňuje přístup k iTunes knihovně
- AV Foundation - Rozhraní pro správu a přehrávání záznamu zvuku
- OpenAL - Multiplatformní pozicování zvuku (3D)
- Core Audio framework - Poskytuje rozhraní pro přehrávání a záznam zvuku

### **3.10.3 Technologie pro video**

Umožňuje přehrávání a záznam videozáznamu a pracovat s nimi v aplikacích. Podporovány jsou následující frameworky.

- Media Player framework - Umožňuje přehrávání videí
- AV foundation - Rozhraní pro záznam a přehrávání videa
- Core media - Popisuje nízkoúrovňové typy a rozhraní používané ve vysokoúrovňových frameworkích

## **3.11 Core services vrstva**

Obsahuje základní systémové služby pro aplikace. Klíčové jsou služby Core Foundation a Foundation frameworks, které definují základní typy pro používání všech aplikací. Tato vrstva podporuje funkce např. lokace, iCloud, sociální sítě a síťové připojení.

### **Služba Peer-to-Peer**

Rámec Multipeer Connectivity poskytuje peer-to-peer připojení pomocí technologie bluetooth. Zařízení umožňuje zahájení komunikace po síti peer-to-peer s dalšími blízkými zařízeními. Multipeer Connectivity se používá především ve hrách a podobných typech aplikací.

#### **3.11.1 Uložiště iCloud**

Umožňuje aplikaci zápis dat a uživatelských dokumentů do centrálního umístění. Do uložště iCloud uživatelé mají přístup ze všech počítačů a zařízení iOS. Úprava a zobrazení souborů je poskytnuta bez nutnosti synchronizace nebo samostatného přenosu souborů.



iCloud document storage se používá pro ukládání a zobrazení dokumentů a dat na iCloud účtu uživatele. iCloud key-value data storage tato funkce slouží ke sdílení dat mezi instancemi aplikace. CloudKit storage, tato funkce se používá pro vytvoření veřejně sdíleného obsahu, nebo při řízení přenosu dat vývojáři.

### **3.11.2 Block Objects**

Je konstrukt programovacího jazyka C, tzv. C-Level, který je možné začlenit do kódu společně s C a Objective-C. Blokové objekty jsou nejčastěji používány pro nahrazení delegátů a delegovacích metod, jako náhrada za rekurzivní funkce a pro usnadnění provádění úlohy pro všechny body v kolekci.

### **3.11.3 Ochrana Dat**

Umožňuje aplikacím práci s citlivými daty uživatele při použití integrovaného šifrování. Pokud je zařízení zamknuté, obsah souborů je nepřístupný. Při odemčení zařízení je vytvořen dešifrovací klíč, který přístup k aplikacím a souborům zpřístupní. Implementace ochrany dat vyžaduje vytvoření a následnou správu chráněných dat.

### **3.11.4 Nákup In-App**

Poskytuje aplikace možnost koupi obsahu uvnitř aplikace, prostřednictvím AppStore nebo iTunes. Tato funkce je implementována pomocí frameworku StoreKit, ten poskytuje infrastrukturu potřebnou pro zpracování finančních transakcí pomocí účtu iTunes.

### **3.11.5 SQLite**

Zprostředkovává vložení lehké SQL databáze do aplikace bez použití samostatného vzdáleného databázového serveru. Knihovna je navržena a optimalizována pro rychlý přístup k záznamům databáze.

#### **Podpora XML**

Foundation Framework poskytuje NSXML parser, třída pro načítání prvků ze souboru XML. V iOS\_SDK jsou umístěny knihovny pro zápis dat pomocí XML souboru a jeho transformace do jazyka HTML.

[13]

## **3.12 Core services Framework**

### **3.12.1 Accounts Framework**

Poskytuje single sign-on model pro uživatelské účty. Single sign-on zlepšuje uživatelský dojem z používání aplikace, eliminuje potřebu přístupu uživatele k více účtům. Tento rámec lze použít společně s rámcem Social Framework.

### **3.12.2 Adressbook Framework**

Umožňuje aplikaci přístup ke kontaktům uživatele, jejich pro jejich zobrazení, ale i úpravu. Přístup ke kontaktům vyžaduje povolení ze strany uživatele. Proto by měla být aplikace připravena pro zamítnutí uživatele k přístupu. V souboru info.plist musí být uvedeno z jakého důvodu aplikace o přístup ke kontaktům žádá.

### **3.12.3 Ad support Framework**

Rámec Ad support umožňuje přístup k identifikátoru, který aplikace využívají k reklamním účelům. Rámec kontroluje odhlášení uživatele ze sledování reklamy.

### **3.12.4 CFNetwork Framework**

Je sada výkoných rozhraní, které využívají objektově orientovanou abstrakci pro práci se síťovými protokoly, jsou založené na bázi programovacího jazyka C. Tento rámec se používá pro zjednodušení komunikace s FTP a http servery, nebo jako řešení DNS hostitele. Umožňuje použití BSD socketů, vytvoření šifrovaného připojení pomocí SSL a TLS. Publikovat a procházet služby Bonjour.

### **3.12.5 CloudKit Framework**

Poskytuje kanál pro přesun dat mezi aplikací iCloud a aplikací. Rámec je možné použít pro správu všech typů dat. Aplikace kde je přímo použit CloudKit mohou ukládat data do složky sdílené všemi uživateli aplikace. Tato veřejná uložště jsou zpřístupněna i na zařízeních bez registrovaného účtu iCloud.

### **3.12.6 CoreData Framework**

Rámec pro správu datového modelu ModelViewController. Rámec CoreData je určen pro použití aplikací, ve kterých je model dat vysoce strukturovaný. Namísto definování datových struktur implementací kódu, je možné použít nástroje v IDE Xcode a vytvořit schéma datového modelu graficky.

Ukládání objektových dat do databáze SQLite , správa funkce dopředu/zpět nad rámec základní editace textu, podpora šíření změn při zachování konzistentního vztahu mezi objekty.

### **3.12.7 Core Foundation Framework**

Podpora následujících typů pole, sady, řady, uzly, String řetězce, Datum a čas, Raw data block, preference, URL a stream, threads a run loops, komunikace Port a socket. Rámec je úzce spjat s Foundation Framework.

### **3.12.8 Core Location Framework**

Poskytuje aplikaci informace o umístění. V případě informací o poloze rámec využívá GPS, wifi a mobilní připojení k nalezení zeměpisné šířky a délky umístění zařízení. Tato technologie se využívá v aplikacích. Další funkce jsou přístup ke kompasu iOS zařízení obsahující magnetometr, podpora oblasti monitorování založené na zeměpisné poloze, podpora využití mobilní sítě s nízkým výkonem a spolupráce s MapKit za účelem zlepšení kvality lokalizačních údajů

### **CoreMedia Framework**

Stanovuje nízkoúrovňové typy médií používané ve AVFoundation Framework.

### **3.12.9 CoreMotion Framework**

Poskytuje jedinou sadu rozhraní pro přístup k dispozici všechna data o pohybu dostupná na zařízení, pomocí akcelerometru za použití nové sady Block-based rozhraní. U zařízení s vestaveným Gyroskopem, načítá gyroskopické data. Rámec nachází časté použití ve hrách a v dalších aplikacích používající pohyb, např. sportovní aplikace.

[13]

### **3.13 Core OS vrstva**

Poskytuje nízkoúrovňové funkce ostatním technologiím, které jsou na ní postaveny.

#### **3.13.1 Accelerate Framework**

Accelerate Framework obsahuje rozhraní pro zpracování digitálního signálu DSP, lineární algebry a výpočty pro zpracování obrazu. Výhodou rámce je optimalizace pro všechna iOS zařízení.

#### **3.13.2 CoreBluetooth Framework**

CoreBluetooth Framework zajišťuje aplikaci interakci s nízkoenergetickými bluetooth zařízeními. Využívá se pro hledání ostatních bluetooth zařízení a jejich připojení, či odpojení, předávání informací pomocí iBeacon mezi zařízeními iOS a pro získání informací o dostupnosti periferních bluetooth zařízení.

#### **3.13.3 External Accessory Framework**

External Accessory Framework poskytuje podporu pro komunikaci s hardwarovými doplňky připojených k iOS zařízení. Příslušenství je možné připojit pomocí dokovacího konektoru nebo bezdrátově pomocí bluetooth.

#### **3.13.4 Local Authentication Framework**

Local Authentication Framework umožňuje použití Touch ID k ověření uživatele. Aplikace mohou vyžadovat ověření uživatele pro přístup k celému obsahu aplikace nebo k jeho části. Uživatel musí být řádně informován o důvodu použití ověření. Pokud je ověření úspěšné aplikace povolí přístup.

### **3.13.5 Network Extension Framework**

Network Extension Framework poskytuje podporu pro konfiguraci a řízení virtuální privátní sítě, tzv. VPN. Aplikace může VPN spustit manuálně nebo může vydávat pravidla pro vyžádání k jejímu spuštění.

### **3.13.6 Security Framework**

Security Framework poskytuje rozhraní pro správu certifikátů, veřejných a soukromých klíčů a zásad důvěryhodnosti. Podporuje generování náhodných šifrovaných čísel. Také podporuje uložení certifikátů a kryptografických klíčů v klíčovém řetězci, využívané pro zabezpečení citlivých uživatelských dat v aplikaci. Umožňuje sdílení klíčových řetězců mezi více aplikacemi, díky tomu není uživatel nucen do všech aplikací zvlášť zadávat své heslo, vyžaduje konfiguraci aplikací v IDE Xcode pomocí zásad důvěryhodnosti.

### **3.13.7 Systém**

Systém zahrnuje kernelové prostředí, ovladače a nízkourovňové UNIX rozhraní operačního systému. Kernel je zodpovědný za každý aspekt operačního systému. Systém spravuje virtuální paměť, souborový systém a komunikaci uvnitř procesů. Ovladače poskytují rozhraní mezi dostupným Hardwarem a systémovými frameworky. Z bezpečnostních důvodů je přístup ke kernelu a ovladačům omezen sadou frameworků a aplikacemi. Využití je v přístupu k souborovému systému, ke standardu I/O, k alokaci paměti a k matematickým výpočtům.

### **3.13.8 64-Bit Support**

Operační systém iOS byl původně navržen pro podporu binárních souborů ve 32-Bitové architektuře. Od operačního systému iOS 7 byla představena kompilace a ladění binárních souborů ve 64-Bitové architektuře. Všechny systémové knihovny a frameworky jsou 64-Bit ready, což znamená, že mohou být použity jak v 32-Bitové tak ve 64-Bitové architektuře.

[14]

## **3.14 Vývoj pro platformu iOS**

S rozvíjejícím se trendem v oblasti mobilních zařízení, se postupně vyvíjí i možnosti vývoje aplikací určených pro operační systém iOS. V iOS je možné vyvíjet aplikace napsané v jazyku C, v

jeho nadstavbě Objective-C a nebo v jazyce Swift. Jedná se o nativní vývoj pro tuto platformu. Dlouhou dobu bylo možné vyvíjet aplikace pouze v aplikaci XCode, což je vývojové prostředí od firmy Apple, nabízené a dostupné zdarma. Toto vývojové prostředí, neboli IDE, je však dostupné pouze pro operační systém Mac OS X, tudíž vývoj v ostatních operačních systémech např. ve Windows či Linuxu nebyl možný. Tento problém se pokusilo vyřešit několik projektů, které se snažily kompilovat programy napsané v jiných programovacích jazycích do nativního kódu Objective-C, nebo Swift. Asi největším počinem v této oblasti je krok společnosti Adobe, která v nové verzi svého nástroje pro vývoj aplikací Flash umožňuje kompilovat právě do programu určeného pro iOS. Tento a jemu podobné nástroje však byly zakázány v licenčním ujednání, ale po velké nevoli ze strany vývojářů byly opět povoleny. Dnes existuje nepřehledné množství vývojových prostředí ať už nativně pro iOS nebo hybridně (crossplatform, vývoj pro více mobilních platform zároveň) např. Xamarin, který úzce souvisí s vývojovým prostředím Visual studio od společnosti Microsoft, nebo frameworky založené na HTML 5 a javascriptu např. ReactNative, Ionic a jiné, které se stávají mezi vývojáři velice populární.

### **3.14.1 Nativní vývoj iOS**

### **3.14.2 Xcode**

Xcode je IDE společnosti Apple, které obsahuje balíček profesionálních vývojářských nástrojů pro vývoj softwarových aplikací na platformy iOS a OS X. Nejnovější dostupná oficiální verze je 7.2, ta umožňuje vyvíjet aplikace na nejnovější verze operačních systémů Apple iOS 9 a OS X El Capitan. Apple ho nabízí volně ke stažení z App Store, ale pouze pro operační systémy OS X. V Xcode je možné implementovat kód v jazyce C, Objective-C a v jazyce Swift. Pro uživatelské rozhraní jsou zde použity frameworky (knihovny) Cocoa, pro OS X a Cocoa Touch pro iOS aplikace.

[15]

### **3.14.3 Objective-C**

Objective-C, často nazývaný ObjC, je objektově orientovaný programovací jazyk implementovaný jako rozšíření jazyka C, do kterého byl přidán systém zasílání zpráv z jazyka Smalltalk. V současné době je používán v operačních systémech Mac OS X, iOS a v GNU projektu GNUstep. Obě prostředí jsou založena na standardu OpenStep.

Překladač tohoto jazyka je součástí GCC. Ovšem nejpoužívanějším překladačem v současné době je clang, díky jeho použití firmou Apple v Xcode.

#### 3.14.4 Swift

Swift je kompilovaný programovací jazyk od společnosti Apple určený pro vývoj na platformách Mac OS X a iOS. Vývoj tohoto programovacího jazyka začal v roce 2010 společností Apple. Oficiálně byl představen v roce 2014 na Apple WWDC 2014. Je považován za alternativu Objective-C a je bezpečnější než jeho předchůdce tzn. nedovolí programátorovi tolik chyb. Zápis kódu je zjednodušený a umožní nám kratší zápis. Umí spolupracovat s existujícími frameworky Cocoa a Cocoa Touch. Swift je kompilován pomocí LLVM a ve stejném programu může být spolu s kódem v jazycích C, Objective-C a Objective-C++. Dalšími výhodami Swiftu jsou: možnost kratšího zápisu kódu, oproti Objective-C a C a dále rychlost operací - např. řazení velkého počtu dat umožní o 50% rychleji.

Ukázka zápisu kódu v Objective-C a Swiftu

```
NSString *str = @"hello,";  
str = [str stringByAppendingString:@" world"];
```

kód v Objective-C

```
var str = "hello,"  
str += " world"
```

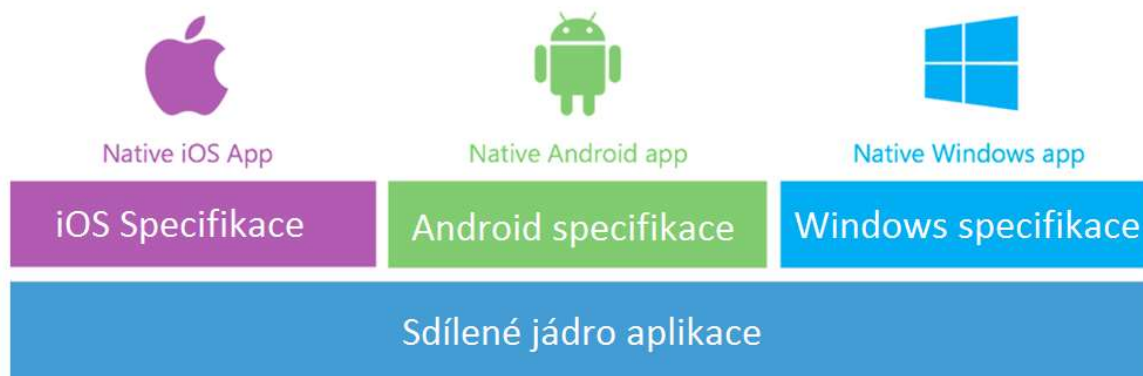
kód ve Swiftu

[16]

### 3.15 Xamarin studio

Xamarin studio je framework a IDE společnosti Microsoft. Jedná se o plnohodnotné moderní vývojové prostředí určené pro vývoj mobilních aplikací pro platformy iOS, WindowsPhone a

Android. V Xamarin studio IDE je vývojářům poskytnuta možnost vytvořit sdílené jádro aplikace, tedy hlavní funkcionalitu a tu použít v jednom kódu pro všechny tři mobilní platformy Android, iOS a WindowsPhone. Programovací jazyk je zde použit C#, objektově orientovaný moderní jazyk od společnosti Microsoft.



### 3.16 Jazyk C#

Je objektově orientovaný programovací jazyk vyvinutý společností Microsoft. Je založený na jazycích C, C++ a Java. Jazyk C# je přeložitelný do jazyka C a C++. Podporuje koncepty zapouzdření, dědičnost a polymorfismus. Zdrojový soubor C# může definovat libovolný počet tříd, struktur, rozhraní a událostí. Porovnání s jazykem Swift viz příloha č.4.

Příklad:

```
string firstName = "John";
```

```
string lastName = "Doe";
```

```
Console.WriteLine("Name: " + firstName + " " + lastName);
```

```
Console.WriteLine("New name: " + firstName + " " + lastName);
```

```
Console.ReadLine();
```

[17]



## 3.17 Co je nového

### 3.17.1 Zdraví

S verzí iOS 8 je do systému integrována funkce zdraví, která pomocí nástroje HealthKit vytváří podmínky pro spolupráci se všemi zdravotními a tréninkovými aplikacemi. Zdravotní a tréninkové aplikace dovedou shromažďovat data jako tepová frekvence, spálené kalorie, cukr v krvi nebo cholesterol.

### 3.17.2 Kontinuita

Novinkou systému je také spolupráce mezi zařízeními značky Apple. Nově pomocí funkce Handoff je možné začít pracovat na mobilním telefonu a práci dokončit u stolního počítače. Stejně tak je možné začít prohlížet webové stránky na tabletu a prohlížení dokončit na telefonu nebo notebooku. Také je nově možné přijímat telefonní hovory skrze iPad nebo zařízení s OS X Yosemite - jedinou podmínkou je připojení obou zařízení ke stejné Wi-Fi síti. Podobně také můžete odesílat SMS zprávy, kteréhokoli typu (SMS, iMessage...) z kteréhokoli zařízení.

### 3.17.3 Rodinné sdílení

Nově je možné sdílet obsah nakoupený v iTunes, iBooks a App Store mezi až 6 členy rodiny, aniž by museli sdílet jeden účet mezi sebou.

Možné je také platit nákupy pomocí jedné platební karty a schvalovat nákupy ostatních (děti). Sdílet jdou také fotografie, rodinný kalendář a ostatní obsah.

[18]

### 3.17.4 FireBase

FireBase je webová aplikace zprostředkávající komunikaci uživatele a serveru. Aplikace se v roce 2014 stala součástí společnosti Google. FireBase funguje způsobem Webservices. A při vývoji iOS aplikace se k němu dá přistoupit pomocí jeho API a SDK -tedy rozšířeného rozhraní a knihoven, které jsou volně ke stažení na webu [firebase.com](http://firebase.com).

Data ve FireBase databázi jsou uloženy v podobě JSON objektu, jedná se o formát pro výměnu dat, založený na jazyce Javascript, v programovacích jazycích Swift a C# se většinou jeví jako pole nebo objekt. Jeho výhodou je podpora mobilních platform Android, iOS a Javascript SDK. V případě

vývoje aplikace pro více mobilních platforem, používají jednu a tu samou FireBase databázi. Další výhodou je ušetření nákladů na správu serveru. V případě FireBase je vývojář při verzi zdarma limitován 100 připojeními naráz a 1GB uložště, při placených verzích od 49 dolarů měsíčně poskytne neomezené množství připojení a uložště např 10GB. V tomto případě je to určeno rozsáhlejšími aplikacím.

Zabezpečení aplikace je zaručeno 2048-bit certifikátem a secure SSL připojením. Databáze umožňuje offline úpravu. Ihned po připojení internetu a přihlášení do databáze se databáze zaktualizuje podle vytvořených úprav. Dalším důležitým aspektem je umožnění registrace a přihlašování uživatelů pomocí Facebook, GitHub, Google a emailem.

[19]

## **4 Vlastní práce**

### **4.1 Logický návrh aplikace**

#### **4.1.1 Příprava**

Při vytváření logického návrhu aplikace je nutno brát v potaz cílovou skupinu uživatelů a jejich návyky v oblasti mobilních aplikací. Když vezmeme v potaz různorodost aplikací a jejich koncipovanost, uvidíme, že aplikace určené pro vzdělávání dětí předškolního věku jsou již v počátcích navrženy jinak, než aplikace určená pro podnikatele. Proto je potřeba využít User Experience (dále UX) – sadu metod a technik využívaných pro návrh uživatelského rozhraní - v tomto případě mobilní aplikace, tak abychom maximalizovali uživatelský prožitek. Díky využití správných metod UX je možné předejít případným nedostatkům v přehlednosti aplikace a usnadnit budoucím uživatelům její používání.

Návrh aplikace byl posouzen s konkurenčními aplikacemi cílícími na potenciálně shodnou skupinu uživatelů. V tomto případě lidmi ve věku 15 až 30 let, kteří se zajímají o zdravý životní styl. Poté byl vytvořen drátěný model, který byl konzultován se sedmi zástupci této skupiny. Předem testován a konzultován s potenciálními uživateli. Poté bylo přistoupeno k vytvoření logického návrhu, potažmo drátěného modelu. Logický návrh obsahuje rozvržení a funkcionalitu prvků v aplikaci. Pro každé view (stránku aplikace) je vytvořen samostatný drátěný model.

#### **4.1.2 Návrh drátěných modelů**

K návrhu drátěných modelů byla použita webová aplikace Ninjamock. Aplikace je určena pro vytváření návrhů mobilních aplikací. Umožňuje nám vybrat si mezi mobilními platformami Android, iOS a windows phone a surface, nebo www stránkami. Po výběru platformy ještě lze vybrat v pravé části aplikace, konkrétní typ zařízení ( zde byl použit iPhone 6), viz příloha č.1.

Při vytváření návrhu aplikace pracuje s grafickými prvky a elementy uživatelského rozhraní, které jsou umístěny v levé části aplikace v rozbalovacím menu a jednoduchým tažením myši se dají přesunout na view. Komponenty lze upravovat, měnit jim vlastnosti např. velikost, barvu, u textu barvu písma atd. vše po vzoru uživatelského rozhraní a to vše v pravém panelu webové aplikace. Aplikace si zachovává rozpracovaný projekt - pokud by došlo k zavření webového prohlížeče,

k projektu je možné se znovu vrátit v bodu, kde byl uzavřen. Po vytvoření uživatelského účtu, lze projekty ukládat do souboru nebo případně sdílet na sociálních sítích a přes email.

#### **4.1.3 První view – Přihlášení a registrace**

Pro přihlášení a registraci bylo zvoleno rozložení komponentů podobné jako u mobilních aplikací konkurence tzn. tlačítko pro možnost přihlášení přes Facebook, které umožní uživateli přístup do aplikace skrz jeho Facebookový účet - pro tlačítko byl zvolen popis Přihlásit se přes Facebook. Dalšími komponenty budou 2 textová pole, 2 texty a tlačítko. První textové pole slouží pro zadání emailu, pro přihlášení do aplikace s popiskem „Zadejte emailovou adresu“ text před tlačítkem bude s popiskem „Email“. Obdobně bylo vytvořeno druhé textové pole, které slouží pro zadávání hesla s popiskem „Zadejte heslo“ a před ním text s popiskem „Heslo“. Pod uvedenými komponenty bylo vytvořeno tlačítko „Přihlásit se přes email“, po jeho stisknutí se uživatel do aplikace přihlásí přes zadanou emailovou adresu.

#### **4.1.4 Druhý view – Recepty**

Na druhou stranu aplikace se uživatel dostane po úspěšném přihlášení. Na této straně se vyskytují recepty a také je to jedna strana ze tří, které budou sloužit jako blok, mezi kterými se pomocí posouvání prstu na displeji ze strany na stranu, bude možnost přepínat. Na této straně bude umístěno v pravém horním rohu logo aplikace. V sekci pro recepty, které budou seřazeny pod sebou, byl umístěn label „Název receptu“. Pod ním Foto receptu jako obrázek a posuvné textové pole s popisem receptu pro ingredience a postup. V dolní části stránky bude graficky znázorněno v jakém view z bloku se uživatel nachází.

#### **4.1.5 Třetí view - Osobní údaje**

Třetí strana slouží pro rozvržení makro živin tzn. bílkovin, sacharidů a tuků do ideálního denního jídla. Jsou zde komponenty label „Jméno“, které si uživatel zadá. Poté následuje textbox a label pro Váhu a změnu. Dále se určuje věk a výška, pro obě možnosti bude vytvořen label pro popis a textbox pro zadání. Pro další kritérium určení makroživin, je důležité určit fyzickou aktivitu uživatele, pro kterou je vytvořené tlačítko s popiskem „fyzická aktivita“, které nás odkáže na rolovací seznam - viz příloha č 3. Po výběru položky v seznamu se aplikace vrátí na „Osobní údaje“. Poté budou přidána 2 přepínací tlačítka pro výběr, buď „Hubnutí“ a nebo „Nabírání svalové hmoty“. Všechna uvedená kritéria ovlivňují výsledek, který je uveden ve spodní části aplikace v Textovém poli. Ve výsledku

bude uvedeno kolik bílkovin, sacharidů a tuků by měl uživatel přijmout ve stravě a také je zde uveden příklad ideálního jídla v podobě kuřecího masa, rýže, olivového oleje a zeleniny v množství určeném na základě výsledných hodnot.

## 4.2 Grafický návrh aplikace

Grafický návrh aplikace byl inspirován Material designem od společnosti Google, který je specifický ostrými hranami, ale i barvami a stínováním jednotlivých komponent - viz zdroj. Grafický návrh byl vytvořen na základě námětu „zdravého životního stylu a fitness“. Proto logem aplikace byla zvolena činka, která vystihuje určení aplikace pro tento námět.

Při spuštění aplikace se nejprve objeví tzv. Launcher view (dále jen Launcher), který je na úvodní obrazovce před načtením komponent a prvků aplikace v zařízení. Na Launcheru je ve středu view logo aplikace.

Následuje první view určené pro přihlášení. Komponenty byly rozděleny do bílých obdélníků, jenž jsou opticky nad vrstvou v pozadí. Tlačítko pro přihlášení přes Facebook je znázorněno modrou barvou, v zápise RGB (59,89,152), typickou pro tuto sociální síť, popisek tlačítka je bílou barvou. Font písma je použitý Google Noto určený pro Material design.

Textová pole jsou hranatá a pod nimi je tlačítko v zelené barvě. Rozměrově stejně veliké jako tlačítko pro přihlášení přes facebook, které poslouží k přihlášení přes email.

## 4.3 Komparace vývojových prostředí

V této kapitole byl porovnán nativní a hybridní vývoj (crossplatform). Porovnání bylo posouzeno důležitými aspekty pro vývoj mobilních aplikací platformy iOS - jimiž jsou cena, náročnost vývoje, podpora, vzhled, funkčnost, doba vývoje a vývojové prostředí. Budou porovnána tři nejpoužívanější vývojová prostředí - PhoneGap, Xcode a Xamarin studio od společnosti Microsoft.

### 4.3.1 Cena

V ceně jsou hlavními kritérii lidské zdroje a finance vynaložené na vývoj. Samozřejmě se cena řídí mnoha dalšími kritérii, která jí ovlivňují a to především podstatou a funkcemi aplikace - čím více bude aplikace komplikovaná, tím déle se vyvíjí a náklady se zvyšují.

Př. aplikace stopky, která měří čas, bude méně nákladná než 3D hra s umělou inteligencí. Cenu také ovlivní aktuální situace na trhu, či nedostatek kvalifikovaných vývojářů.

V obecné rovině lze určit, že vývoj aplikace hybridním způsobem použitím frameworku HTML5 a javascriptu je levnější, než vývoj nativní v IDE Xcode a nebo v Xamarin studiu.

Průměrný roční plat nativního iOS vývojáře je zhruba 2 miliony Kč ročně a průměrný roční plat vývojáře .NET pro Xamarin studio se průměrně pohybuje okolo 1,6 miliónů Kč, v porovnání s tím průměrný plat vývojáře v HTML5 je zhruba 1,3 milionu Kč ročně. Díky tomu jsou náklady nativního vývoje vyšší než u HTML 5. Jeden z důvodů je vysoký počet vývojářů se znalostí HTML 5 a javascriptu a to je také jeho výhodou oproti nativnímu vývoji. Pokud budeme brát v úvahu vývoj pouze pro mobilní platformu iOS, tak zde se vyšší cena neodrazí jako v případě vývoje aplikace pro více platforem např. Android. Zde už rozdíl ceny vývoje nativního a hybridního je markantní. Stále cenou vyhrává vývoj v HTML 5 frameworku oproti vývoji v Xamarin studio, kde se také jedná o vývoj pro více platforem.

#### **4.3.2 Náročnost vývoje**

Náročnost vývoje je velice subjektivní aspekt. Nativní styl vývoje obnáší obecně rozsáhlejší znalosti, než je to u vývoje hybridního. I přesto závisí na rozsáhlosti aplikace. Také závisí na zkušenostech vývojáře a podpoře komunity, jenž může v případě nedostatku znalostí využít. Dostáváme se k použitým programovacím jazykům a frameworkům, které náročnost vývoje ovlivňují nejvíce. Co se týče nativního vývoje v prostředí Xcode - od roku 2014 kdy byl vydán programovací jazyk Swift, se náročnost vývoje značně zjednodušila, kód je kratší a Apple jazyk vyvinul přímo pro vývojáře iOS a OS X. Nejen že vývojáři mohou použít programovací jazyk Swift ale také s ním zároveň v jednom kódu jazyky jako Objective-C a C.

V Xamarin studiu se využívá programovací jazyk C# - jedná se o jednoduchý, moderní a objektově orientovaný programovací jazyk, založený na syntaxi jazyku C, stejně jako Swift. Porovnání zápisu kódu v jazyce Swift a C#.

V hybridním vývoji PhoneGap se využívá kód v jazyce HTML5 a javascriptu. Vývoj je podobný vývoji webové aplikace. Kód v jazyce HTML je graficky upravován pomocí stylů a je zde nutná znalost i frameworku javascriptu využívající například opensourcové vývojové prostředí Apache cordova, konkurent PhoneGap.

### 4.3.3 Podpora

U vývoje softwaru, ať už jde o vývoj mobilních aplikací a nebo aplikací určených pro desktop, může nastat problém se kterým si vývojář neví rady a je potřeba se na někoho zkušeného obrátit. Tuto pomoc lze získat z knižních vydání, týkajících se dané problematiky pro určené odvětví, z odborných článků na internetu, video tutoriálů nebo na internetových fórech - nejčastěji Github.com, Stackoverflow.com a dalších komunitních fórech týkajících se daného vývoje, lze také využít placené podpory.

Nativní vývoj, co se týká podpory, jednoznačně vede. Počet knižních publikací i internetových článků u hybridního vývoje je výrazně nižší. Také počet dotazů zodpovězených na diskuzním fóru stackoverflow.com jednoznačně mluví ve prospěch nativního vývoje. Dotazů s označením Swift je okolo 80 tisíc, dotazů s označením „C# Xamarin“ je okolo 20 tisíc, o něco lépe jsou na tom dotazy „PhoneGap“ ty čítají okolo 45 tisíc.

### 4.3.4 Vzhled a funkčnost

Co se týče kvality, zde byl hodnocen vzhled a funkčnost aplikace. Vzhled aplikace z velké části určuje vývojové prostředí. V tomto ohledu vítězí nativní vývoj. Uživatelé mobilních zařízení jsou často zvyklí na určité typy aplikací a jejich chování, tzn. orientace v aplikaci a použité prvky. V tomto je jedna z větších nevýhod hybridního vývoje. Hybridní vývoj nepodporuje uživatelské rozhraní aplikace od společnosti Apple - Cocoa Touch. Uživatelské rozhraní může být napodobeno a nahrazeno pomocí CSS stylů a javascriptu. Lze napodobit vzhled i chování, ale bohužel ne zcela dokonale. Nejedná se pouze o použité komponenty, ale i například přecházení mezi obrazovkami a o odezvu aplikace při chodu aplikace.

Uživatelský zážitek z používání aplikace není tak příjemný, jako u nativního vývoje. Jednoduše řečeno, uživatel od mobilní platformy, kterou používá, očekává obdobné reakce **jako u ostatních aplikací**. Ovládání aplikace má být pro uživatele příjemné, aby se snadno s aplikací naučil pracovat. U hybridního vývoje je tento uživatelský zážitek znatelně méně příjemný.

Vývojové prostředí Xcode, ale i Xamarin studio, podporují uživatelské rozhraní Cocoa Touch. V tomto ohledu jsou obě 2 vývojové prostředí na stejné úrovni. Chování aplikace je u obou IDE také stejné a aplikace by měla uživateli poskytnout v obou případech obdobné vlastnosti.

Z hlediska výkonu vše také mluví ve prospěch nativního vývoje v Xcode a Xamarin studio. Výkon je jedním z kritérií kvůli kterému se často volí nativní přístup oproti přístupu hybridnímu např.

PhoneGap. Aplikace náročné na grafiku a výkon např. 3D hry, využívají ve PhoneGap javascriptový engine, který se samozřejmě postupem času zrychluje, stále pořád nepodává žádaný výpočetní výkon jako je tomu u nativního vývoje. Pro náročnější aplikace při žádaném uživatelském zážitku, není PhoneGap a jiné frameworky pro hybridní vývoj ideální výběr.

Jedním z klíčových kritérií při vývoji aplikací je využitelnost API (rozhraní pro programování aplikací). API umožňují přístup k různým webovým službám a aplikacím 3 stran, např. Facebook, mapy, GPS atd. U vývoje aplikací hybridním přístupem není zaručena podpora všech nativních rozhraní, jako je tomu u nativního vývoje, díky. PhoneGap využívá v dnešní době 18 API oproti využití v Xcode nebo v Xamarin je to řádově menší číslo, Nativní vývoj jak v Xamarinu tak v Xcode. podporuje všechna nativní rozšíření mobilních aplikací, které mohou vývojáři využívat.

#### **4.3.5 Doba vývoje**

Co se týče vývoje aplikace pouze pro platformu iOS. Se doba vývoje aplikace v hybridním i nativním vývoji téměř rovná. Doba vývoje nejvíce závisí na rozsáhlosti dané aplikace a především na počtu vývojářů, které na dané aplikaci pracují. Celý vývoj aplikace samozřejmě není spojen pouze s vývojáři, jedná se zde také o UX designéry, grafiky apod.

Pokud je žádaný vývoj pro více mobilních platform, jde o přenositelnost aplikace. Nejkratší dobu vývoje vykáže hybridní přístup PhoneGap. Ve kterém se využívá jeden zdrojový kód pro přenesení na ostatní mobilní platformy např. Windows Phone, Android atd.

Xamarin studio nabízí podobnou možnost jako u hybridního vývoje. V jazyce C# se napíše jádro aplikace, které je přenositelné mezi mobilními platformami a jediné co vývojáři upravují je uživatelské rozhraní, které je pro každou mobilní platformu specifické. V tomto ohledu je Xamarin o něco více časově náročný, než například PhoneGap, přesto však doba vývoje není tak vysoká.

Jako je tomu u použití nativního přístupu vývoje pro iOS pomocí Xcode, pro Android pomocí Android studio a pro Windows Phone pomocí Visual studia. Tento způsob vývoje pro více mobilních platform, je velmi časově náročný a žádá si i mimo jiné dobrou komunikaci mezi vývojáři samotnými, doba vývoje se často prodlužuje.



### 4.3.6 Vývojové prostředí

#### 4.3.6.1 PhoneGap

Je vývojové prostředí společnosti Adobe. Umožňuje kompilování HTML5 a javascript kódu do programovacího jazyka Objective-C. Vývojové prostředí si vývojáři mohou zvolit, kterékoliv je určené pro vývoj webových aplikací. Důležitým kritériem je i volba emulátoru mobilního zařízení, který simuluje reálné zařízení a vývojáři mohou aplikaci otestovat, ještě před nasazením do produkce. Vývoj hybridních aplikací je co se týče emulátorů odkázaný na nativní vývojové prostředí, v tomto případě emulátor v aplikaci Xcode a nebo Xamarin.

Při volbě ladících a kompilujících nástrojů jsou také odkázány na IDE nativního vývoje.

#### 4.3.6.2 Xcode

IDE společnosti Apple obsahuje veškeré náležitosti potřebné k vývoji aplikace pro iOS a je celý zdarma ke stažení na App Store. Xcode je dostupný pouze pro uživatele operačního systému OS X. Ladící a testovací nástroje jsou součástí IDE. Ladící nástroje tzv. debugging umožňují vývojáři kontrolu jejich kódu. Ladící nástroje jsou součástí sady kompilační open-source sady LLVM, slouží především ke generování a kompilaci kódu tzn. optimalizaci. LLDB debugger je součástí kompilační sady LLVM. Debugger slouží k hledání chyb při ladění kódu. LLVM i LLDB jsou integrovány do IDE Xcode a přistupuje se k nim pomocí uživatelského rozhraní.

V IDE je integrován také emulátor zařízení, které reálně simuluje zařízení iPhone, iPad, iPod touch, ve všech jejich verzích. Vývojář má poté možnost aplikaci ozkoušet na více než jednom zařízení a odladit tím případné chyby spojené s rozvržením aplikace a velikostí zařízení. V emulátoru je možné modifikovat, podle potřeb vývojáře, podporuje přístup k API rozhraní, ať už jde o rozhraní zařízení, jako správa baterie, notifikace, kalendář tak API aplikací třetích stran např. Facebook, Twitter, Evernote apod.

Kód v IDE Xcode je možné psát v objektově orientovaném jazyce Swift, nebo jeho předchůdcích Objective-C a C, na kterých je programovací jazyk postaven. Při psaní kódu v jazyce Swift je možné souběžně s ním psát kód v programovacím jazyce C i Objective-C. Viz příloha č.2.

#### 4.3.6.3 Xamarin Studio

Je platforma IDE Visual studio od společnosti Microsoft. Kód je psaný v objektově orientovaném jazyce C#, který stejně jako programovací jazyky v podobě konkurence Apple, je postavený na základě jazyku C. Největší výhodou Xamarinu oproti Xcode je možnost vyvíjet aplikace pro více mobilních platform zároveň. Xamarin je dostupný pro uživatele Windows i OS X. Při vývoji pro iOS je bohužel limitován také operačním systémem OS X. Pro vytvoření aplikace je nutné mít registrované developerské AppleID. Stejně jako u Xcode. Xamarin s Visual studio IDE je dostupný zdarma pouze po dobu 30 dní, poté je za cenu 25 dolarů měsíčně pro základní komerční využití. Lze si připlatit za placenou podporu. Pro studentské nekomerční využití je pro registrované školy u Microsoft zcela zdarma.

Ladící a kompilující sadu využívá pro iOS stejnou jako Xcode. Ladící nástroj využívá LLDB a pro kompilaci a generování kódu LLVM. Pro opravu a kontrolu chyb používá díky tomu stejné postupy např. breakpoint, pro přerušení debuggeru a další.

Uživatelské prostředí IDE je postaveno obdobně jako u IDE Xcode viz příloha č.10.

Emulátor pro zařízení iOS se do Xamarin studia dá doinstalovat zdarma. Poté je možné kód reálně testovat přímo v IDE. Jsou zde k dispozici všechna zařízení podporující nové iOS 9.

#### 4.3.7 Shrnutí komparace vývojových prostředí

Pro vývoj náročné aplikace pro výpočet je téměř jedinou volbou nativní vývoj aplikace v Xcode nebo v Xamarin studio. Hybridní vývoj ve PhoneGap nepodporuje velké množství rozšíření API a uživatelský zážitek není příjemný. Podpora v podobě komunity a tutoriálu se také nedá s nativním vývojem srovnávat, proto nebyl zvolen. Po konečném srovnání bylo zvoleno IDE Xcode od společnosti Apple. Funkcionalitou, vývojovým prostředím se s Xamarin Studií rovnají, ale co se týče podpory je Xcode stále v popředí. Ve shrnutí byla znázorněna bodovací metoda v podobě tabulky, kde byly zahrnuty všechna porovnávací kritéria.

Tabulka 3 - Porovnání vývojových prostředí

	Xamarin studio	Xcode	PhoneGap
Cena	3	3	5
Náročnost vývoje	3	3	5
Podpora	3	5	2
Vzhled	5	5	2
Funkčnost	5	5	3
Doba vývoje	3	3	5
Vývojové prostředí	5	5	1
Celkem:	27	29	23

Zdroj: [Autorský archiv]

#### 4.4 Seznámení s IDE Xcode

Jak bylo zmíněno v komparaci vývojových prostředí. IDE Xcode je zcela zdarma a dostupný je pouze v operačním systému OS X od Apple. Pokud není vývojář již vlastníkem zařízení s OS X, musí uživatelé ostatních OS např. Windows mají možnost zvolit nainstalování jejich systému pomocí virtuálního prostředí např. Oracle VMVirtualBox nebo VMWare player. Bohužel virtuální operační systém nedosahuje zdaleka takového výkonu, jako zařízení od společnosti Apple, ale při absenci těchto zařízení je pro vývoj v IDE Xcode nezbytný.

IDE Xcode lze nainstalovat přímo z aplikace AppStore viz **příloha**. Xcode zabírá místo na pevném disku zhruba 4 GB. Při spuštění Xcode lze zvolit vytvoření nového projektu, playground a otevření existujícího projektu. V pravém panelu jsou vidět projekty, které byli nedávno spuštěny. Při vybrání nového projektu je poskytnut výběr operačních systémů pro, které bude projekt, tedy námi tvořená aplikace, vytvořena. Na výběr je iOS, OS X a WatchOS.

Při vytváření iOS aplikace lze upřesnit pro jaký typ zařízení se aplikace vytváří. Pro tuto aplikaci byl vybrán ty „Universal“, který podporuje grafické rozhraní iPhone, iPod Touch i iPad tzn. pro všechny tyto typy rozlišení, byla aplikace optimalizována a pomocí emulátoru je možné ji testovat.

## 4.5 Uživatelské rozhraní

Na základě grafického návrhu bylo vytvořeno nejprve uživatelské rozhraní. Do aplikace byl přidán font písma Google Noto, který Xcode fontech chybí, v info.plist byli v položce Fonts provided by application přidán. Uložen byl do nově vytvořené složky Resources.

### LauncherScreen

LauncherScreen je úvodní strana aplikace, která se ukáže ještě před načtením aplikace a prvního view. V této první stránce je ve středu zobrazení logo aplikace, obrázky se do uživatelského prostředí vkládají pomocí UIImageView.

#### 4.5.1 První strana

V prvním view aplikace byli přidány 2 view v horní části view v zelené barvě a opticky ho překrývající bílé view. View v pozadí bylo zabarveno světle šedou barvou. Poté byla vytvořena třída MaterialView, která udává stínování a ostatní vlastnosti překrývajícího view, tedy opticky vystupuje nad ostatní objekty, použitím například layer.cornerRadius, layer.borderColor. Při vytváření třídy je nutné správně zvolit podtřídu, v tomto případě UIView.

Nad MaterialView byl umístěn do levého horního rohu Label nápisem „Přihlášení/Registrace“ v bílé barvě, do pravého horního rohu bylo umístěno logo aplikace. Dále bylo přidáno tlačítko pro přihlášení přes Facebook a ikona Facebooku. Pro tlačítko byla vytvořena třída MaterialButton ve které byly použity obdobné vlastnosti jako tomu je u třídy MaterialView.

V části „Přihlášení přes email“ byl vytvořen View s použitím třídy MaterialView. Dále Label pro nadpis Oddílu „Přihlášení přes email“. Byla přidána dvě textová pole, pro která byla vytvořena třída MaterialTextField, ve které se upravují vlastnosti na základě návrhu.

První textové pole je určeno pro zadávání emailové adresy. Proto mu byl nastaven placeholder „Email“, tzn. text který je v textovém poli ještě před zadáním textu uživatelem. Při zadávání text automaticky zmizí a nahradí se zadaným textem obvykle stylisticky rozlišným. V tomto případě byla

pro placeholder použita světle šedá barva a na text při zadávání byla použita barva tmavě šedá. Tak aby uživatel na první pohled uviděl rozdíl.

Pro druhé textové pole byl nastaven placeholder „Heslo“, se stejnou stylizovanou třídou `MaterialTextField`. Viz příloha č.8.

#### **4.5.2 Druhá strana**

Na této straně se zobrazují recepty. Byl zde vytvořen `ViewController` společně s `NavigationController`em. Slouží pro navigaci mezi různými `ViewControllery`. V tomto případě nás navede na nově vytvořený `ViewControler`, který obsahuje `TableView`, to nám umožňuje pomocí `TableViewCell`s, což jsou jednotlivé buňky `TableView`, umístit obsah vertikálně pod sebe seřazený pomocí využití např. `FireBase` databáze, která byla vytvořena.

Pro tento krok se všechny potřebné komponenty pro jednotlivé recepty umístí do buňky `TableView`. Každý recept má název, pomocí `Labelu`, obrázek, pomocí `ImageView`, dobu přípravy, pomocí `Labelu` a poté je přidán obrázek srdíčka pro možnost zvolení „To se mi líbí“ a počet „To se mi líbí“ u každého receptu. Všechny tyto prvky byly umístěny na vytvořené `View` s použitím vytvořené třídy `MaterialView` pro vzhled `Material` designu. Celá buňka slouží stejně jako tlačítko díky stisknutí obrázku nebo názvu receptu, nás odkáže na další `ViewController`, kde je recept zobrazen v detailním popisu. Viz příloha. Pozadí `ViewControlleru` je světle šedou barvou a `NavigationBar` je barvou zelenou. Stejnou jako bylo vytvořená první strana.

#### **4.5.3 Detail receptu**

Na této straně byl umístěn detail receptu. Recept obsahuje název, obrázek, počet „To se mi líbí“, obrázek srdce pro „To se mi líbí“, dobu přípravy a také Popis postupu. Všechna tyto data budou získána z databáze `FireBase`. Všechny komponenty jsou zarovnány na střed.

`NavigationBar` byl stylizován zelenou barvou a bylo zde přidáno tlačítko zpět. V pravém horním rohu bylo umístěné logo.

#### **4.5.4 Třetí strana**

Na této straně si uživatel zadá údaje potřebné k vypočítání ideálního denního jídla, které může zařadit do jídelníčku. Jídlo obsahuje poměrný počet makroživin. Skládá se z kuřecího masa, olivového oleje,

ryže a zeleniny. Tyto suroviny byly zvoleny na základě poměrně snadné dostupnosti v obchodech s potravinami a supermarketech a jejich nutričních hodnot.

V první řadě byl vytvořen label pro uživatelské jméno s názvem „Jméno“. Údaje k sestavení jídla jsou váha, věk, výška a fyzická aktivita. Kromě těchto atributů je potřeba zvolit si dosažení cíle v podobě hubnutí, nebo nabírání svalové hmoty.

Údaje váha, věk a výška byly vyřešeny pomocí textového pole a labelu, který je popisuje. Textové pole jsou stejně velká a tyto prvky jsou rozděleny do dvou sloupců. Váha a výška jsou umístěny v prvním sloupci a ve sloupci druhém jsou umístěny věk a tlačítko pro výběr fyzické aktivity, které nás odkáže na stranu s rolovacím seznamem, kde jsou aktivity vyjmenovány.

Všechny potřebné údaje pro sestavení jídla a uživatelské jméno budou umístěny ve vytvořeném View se třídou MaterialView. Textovým polím bude přidělena třída MaterialTextField a tlačítku třída MaterialButton, tak aby se ve vzhledu aplikace zachoval vzhled Material designu.

V dalším kroku bylo vytvořené View se třídou MaterialDesign, ve kterém byli umístěny hodnoty sestaveného jídla. Pomocí textového pole. Hodnoty se vypočítají podle zadaných údajů a vzorce pro výpočet Makroživin a kalorického příjmu. Výsledek tohoto vzorce bude převeden na gramy jednotlivých surovin ve vytvořeném jídle. Viz příloha č. 8.

#### **4.5.5 Backend – server**

V aplikaci byla potřeba komunikace se serverem, pro generování receptů na druhém view aplikace. Na server se uložili recepty a aplikace si je při připojení na internet automaticky načte. Recepty v aplikaci nebudou všechny uloženy v offline režimu, tzn. režim bez přístupu zařízení na internet, na druhou stranu uživateli ušetří místo v zařízení.

Pro serverové řešení byla použita webová aplikace Firebase od společnosti Google. Její výhodou je jednoduchost v ovládání i bez znalostí programování aplikace na straně serveru. Zároveň je Firebase již zabezpečená proti případným útokům a podporuje jazyky Swift, Java, Javascript a Objective-C, pro které je vytvořena také dokumentace viz [zdroj](#). Obsahuje pro aplikaci klíčovou iOS SDK.

Ve Firebase bylo potřeba aplikaci nejdříve nastavit, pomocí jednoduchého nastavení a návodu na stránce. V nastavení aplikace v záložce Login and Auth, byla povolena autentikace pomocí emailu a Facebooku, která je pro nás klíčová v přihlášení uživatele do aplikace. Firebase komunikuje s API třetích stran a k Facebook API bude přistupovat z důvodu autentikace uživatelů.

V sekci data byly vytvořeny 2 objekty users a recipes. Do objektu users se ukládají vytvořené uživatelské účty pomocí přihlášení přes Facebook nebo přes email. Každému vytvořenému uživateli se vytvoří ID, uživatelské jméno a provider, tzn. pomocí které služby se uživatel přihlásil.

## 4.6 Implementace kódu

### 4.6.1 Přihlášení pomocí Facebook

Pro autentikaci uživatele v aplikaci pomocí Facebooku je nutné využít Facebook API, tzn. rozšíření rozhraní aplikace pro přístup k Facebook službám. Vše co bylo potřeba postupovat podle dokumentace jak Facebooku tak Firebase. Na stránce developers.facebook.com, je detailní návod jak využít právě Facebook API, je nutné se přihlásit na Facebookový účet nebo účet vytvořit. Na developerském účtu Facebooku je potřeba vytvořit aplikaci u které se zvolí pouze jméno a zaměření aplikace. V tomto případě bylo zvoleno „Zdraví a fitness“. Poté bylo zvoleno určení aplikace a to konkrétně pro platformu iOS.

V nastavení je potřeba vyplnit v sekci Security redirect OAuth url a Bundle ID aplikace, Bundle ID je stejné jako Bundle Identifier v IDE Xcode. Které je dostupné ve Firebase jako URL adresa aplikace. V sekci Status and Review je potřeba povolit možnost veřejného publikování. Ve Firebase je v sekci Login and Auth potřeba vyplnit Facebook App secret a Facebook App ID, které bylo získáno z nastavení Facebook App.

V dalším kroku bylo staženy Facebook SDK v aplikaci byla vytvořena nová složka s názvem Frameworks a do ní byli uloženy všechny frameworky z SDK. Pomocí CocoaPods aplikace byli frameworky Firebase a Facebook společně nainstalovány.

Dalším krokem byla úprava souboru Info.plist v IDE Xcode a poté do něj byl vložen kód pro přístup k API viz příloha. V souboru AppDelegate.swift byli importovány knihovny z Facebook SDK a přístup k Facebook API byl aktivován ve funkci viz příloha.

Ve ViewControlleru první strany byly importovány knihovny FBSDKCoreKit a FBSDKLoginKit a ve třídě ViewController byla vytvořena funkce pro tlačítko „Přihlásit se přes Facebook“. Pro metodu FBSDKLoginManager() byla vytvořena proměnná, kterou byla metoda nahrazena pro zkrácení zápisu kódu. Funkce fbBtnPressed znázorňuje funkci při stisknutí tlačítka.

Ve funkci byla implementována funkce logInWithReadPermissions() ve které bylo zajištěno přihlášení přes Facebook viz příloha. Pokud se uživatel přihlásí pomocí Facebooku v konzoli bylo

zapsáno „Úspěšně jste se přihlásili přes Facebook.“ a Facebook API vrátil AccessToken uživatele, v případě že uživatel zadal chybné heslo nebo uživatelské jméno v konzoli bylo zapsáno „Error, neúspěšné přihlášení přes Facebook“ a uživatel se vrátí zpět na stranu pro přihlášení. Viz příloha č.5.

#### **4.6.2 Přihlášení pomocí emailu**

V aplikaci byla vytvořena funkce attemptLogin při stisknutí tlačítka „přihlásit se přes Email“, pro ověření zadání emailu a hesla. Které nám pomocí vytvořené funkce showErrorAlert, která uživateli zobrazí alertController, při nezadání hesla nebo emailu do textového pole, s nápisem „Zadejte email a heslo“.

Dalším krokem je správná autentikace uživatele, pokud již uživatel vytvořený účet v databázi má, pouze ho do aplikace přihlásí. Pokud email v databázi není evidován, účet mu bude vytvořen.

V souboru Constants.swift byla přidána proměnná SEGUE\_LOGGED\_IN a STATUS\_ACCOUNT\_NONEXIST, obě dvě proměnné jsou konstanty. Typ proměnné „let“, konstantní proměnná, které není možné měnit. Ve ViewControlleru ve funkci DataService.ds.REF\_BASE.createUser je do databáze vytvořen nový uživatel pomocí emailu a hesla. Email a heslo jsou zapsány do databáze. Pokud je zadáno chybné heslo, které se neshoduje s databází. Zobrazí se alertController s nápisem „Chyba v přihlášení. Zadejte Váš email a heslo.“. Viz příloha č. 6, 9 a 10

#### **4.6.3 Recepty**

V této kapitole byla provedena implementace kódu pro 2 stranu aplikace. Pro zobrazení seznamu receptů byl vytvořen TableViewController, který recepty seřazuje pod sebe. Každá buňka seznamu obsahuje název receptu, obrázek, počet „To se mi líbí“ a možnost zvolení „To se mi líbí“. Tlačítko pro zvolení sympatií receptu „To se mi líbí“, bylo implementováno z důvodu lepšího uživatelského dojmu z používání aplikace a také možnosti sledovat, které recepty se líbí ostatním uživatelům aplikace. Každá buňka je naplněna z databáze FireBase, pomocí vytvořené třídy DataServices a PostCell, který představuje implementace funkcionality, tzn. získání dat z databáze, jejich následné převedení na potřebný formát pomocí frameworku Alamofire viz zdroj, který umožňuje úpravu a práci s obrázky.



Pomocí kliknutí na buňky v TableView se zobrazí další strana na které je detail receptu, který obsahuje název, popis, obrázek a také tlačítko pro zvolení „To se mi líbí“ a jejich počet. Vše je vygenerováno také pomocí databáze FireBase.

#### 4.6.4 Alamofire framework

Alamofire je networking HTTP knihovna napsaná v Swift. Alamofire Framework obsahuje knihovnu AlamofireImage, která obsahuje rozšíření pro systémové třídy UIImage a UIImageView, vlastní obrazové filtry a prioritně založený systém pro stahování obrázků. Další obsaženou knihovnou v Alamofire je AlamofireNetworkActivityIndicator, který řídí viditelnost indikátoru síťové aktivity na iOS zařízeních a obsahuje konfigurovatelné spoždění časovače pro zmírnění blikání a podporuje NSURLSession instance, které nejsou řízeny Alamofire. Pro použití Alamofire je důležité aplikace vyvíjet pro verzi operačního systému iOS 8 a novějších, za použití IDE Xcode ve verzi 7.2 a novější.

#### 4.6.5 Osobní údaje

Na druhé stránce aplikace „Osobní údaje“ byla vytvořena funkcionality pro výpočet ideálního denního jídla. Na základě výpočtu indexu pro bazální metabolismus BMR.  $BMR(\text{ženy}) = 655,0955 + (9,5634 \times \text{váha v kg}) + (1,8496 \times \text{výška v cm}) - (4,6756 \times \text{věk v letech})$   
 $BMR(\text{muži}) = 66,473 + (13,7516 \times \text{váha v kg}) + (5,0033 \times \text{výška v cm}) - (6,755 \times \text{věk v letech})$

Pomocí tohoto vzorce a hodnot zadaných uživatelem do textového pole. Počítá se zde s průměrnou stavbou těla, určení ideálního jídla není vždy jednoznačné, každý metabolismus reaguje rozdílně, proto nelze přesně definovat, kolik energie z jídla by měl člověk přijmout, aby dosáhl svého cíle. Nicméně tato aplikace slouží pouze jako vodítko k tomu jak by mohlo ideální jídlo vypadat. Vypočtené BMR bylo zvýšeno o energii, která se spotřebovává při fyzicky náročných aktivitách.

Tabulka 4 seznam pohybových aktivit

Pohybová aktivita:	Muži	Ženy
Žádná	1,4	1,4
Lehká (méně než hodinu denně)	1,5	1,5
Mírná (hodinu denně)	1,7	1,6
Střední (1-2 hodiny denně)	1,8	1,7
Těžká (více než hodinu denně)	2,1	1,8

Zdroj: [25]

Výsledná hodnota byla energie, kterou tělo musí přijmout z potravy za celý den. Je potřeba rozpočítat hodnotu pro 1 denní jídlo a to tím, že výsledná hodnota byla vydělena pěti. Číslo pět bylo zvoleno z důvodu 5 denních jídel, které by měl správně každý člověk přijmout. Z procentuální množství makroživin v jídle pro hubnutí tuku by mělo být v poměru 40% bílkovin, 40% sacharidů a 20% tuku pro přibírání svalové hmoty se BMR vynásobí dvakrát a poměr makroživin bude 60 % sacharidů, 25 % tuků a 15 % bílkovin.

Výsledek se uživateli zobrazí v poli TextField, kde se do textového řetězce vypíše „Kuřecí maso: %d, Rýže basmati: %d, Olivový olej: %d a Zelenina: %d“.

## **5 Zhodnocení výsledků**

### **5.1 Logický návrh aplikace**

Logický návrh aplikace byl vytvořen na základě vytvořených drátěných modelů, které byly otestovány na zástupcích cílové skupiny, pomocí metod User experience. Testování drátěných modelů bylo velmi časově náročné. Rozmístění komponent aplikace bylo upraveno na konečný výsledek, tak aby uživateli bylo používání aplikace příjemné. Logický návrh čerpal z aplikací zabývajících se zdravím životním stylem a recepty např. Kalorické tabulky a Fit recepty. Při vytváření drátěných modelů byla využita aplikace Ninjamock. Vytváření modelů pomocí aplikace Ninjamock je velice intuitivní a je určeno pro uživatele, kteří nedisponují zkušenostmi z oblasti User experience. V logickém návrhu byli zváženy možnosti přihlašování, nakonec díky testování byla zvolena verze s dvěma druhy přihlašování. Tím prvním je pomocí účtu sociální sítě Facebook a tím druhým je přihlašování pomocí emailu.

Dále bylo řešeno rozmístění a velikost komponent na druhé straně aplikace, která se zabývá seznamem receptů. Každému receptu byl přiřazen název, postup a obrázek. Recepty jsou řazeny postupně pod sebou. Třetí strana aplikace obsahuje prvky pro výpočet jídla. Ty byly rozmístěny tak aby byl uživatelský požitek z aplikace příjemný.

Logický návrh aplikace si vzal inspiraci z aplikací zabývajících se recepty a zdravým životním stylem např. Fitrecepty a Kalorické tabulky.

### **5.2 Grafický návrh aplikace**

Grafický návrh byl vytvořen pomocí Material designu. Bylo nutné vytvořit komponenty a přidat jim stínování a příslušné barvy. Všechny podklady pro tvorbu Material designu byly čerpány z oficiální dokumentace Material designu od společnosti Google. Nejdůležitějším faktorem pro tvorbu materiálu designu bylo použité stínování komponent, pro některé komponenty bylo stínování určeno pro jiné naopak bylo stínování zakázáno. Komponenty použité jako bílý podklad stínování obsahují, vyvolávají dojem vystupování komponenty na povrch, zatímco například textové pole stínování neobsahovalo, ale byly dané tloušťky ohraničení a font písma. Viz. Příloha č. 8.

### 5.3 Komparace vývojových prostředí

Komparace vývojových prostředí byla spojena se studováním a testováním tří vývojových prostředí Xcode IDE, Xamarin a PhoneGap. Studie jednotlivých vývojových prostředí byla spojena se sběrem informací z jejich technických dokumentací a odborných článků zabývajících se daným vývojovým prostředím. Pro komparaci vývojových prostředí byly stanoveny kritéria hodnocení, které mají přímý dopad na vývoj aplikace. Komparace by měla potenciálním vývojářům nastínit výhody a nevýhody jednotlivých vývojových prostředí. Pro hodnocení kritérií byla využita bodovací metoda. Výsledné hodnoty byly mezi sebou porovnány a následně bylo vybráno vývojové prostředí, ve kterém byla provedena implementace kódu. Nejlépe hodnoceným prostředím se stal Xcode IDE od společnosti Apple. Byl vybrán z důvodu použití nativního vývoje a tím byla zaručena plnohodnotná funkcionálnost aplikace a podpora komunity nebo oficiálních zdrojů společnosti Apple pro vývoj iOS. Pro implementaci kódu jsou určeny k použití dva programovací jazyky Swift a Objective-C. Zvolen byl objektově orientovaný programovací jazyk Swift z důvodu využití moderních postupů vývoje, Swift také poskytuje kratší zápis kódu nežli je tomu v Objective-C.

### 5.4 Uživatelské rozhraní

Vytvoření uživatelského rozhraní vycházelo z grafického návrhu aplikace. Pro komponenty byly vytvořeny třídy, pomocí kterých byl upraven vzhled. Hlavní problematikou těchto vytvořených tříd bylo určení stínování, případně zaoblení rohů jednotlivých komponent. Při vytváření tříd byly použity atributy vycházející z dokumentace Material designu. Další důležitou věcí je použití Constraints. Constraints se používá jako uchycení jednotlivých komponent a udávání vzdálenosti od nejbližší komponenty, tzn. pomocí Constraints se umístí komponenty na ViewController, tak aby byla zaručena kompatibilita se všemi iOS zařízeními. Uchycení komponent pomocí ViewControlleru bylo provedeno na všech stranách aplikace a následně bylo rozmístění aplikací otestováno na simulátoru poskytovaného ve vývojovém prostředí Xcode.

### 5.5 Implementace kódu

K implementaci kódu bylo zvoleno vývojové prostředí Xcode IDE a objektově orientovaný programovací jazyk Swift. Prvním řešeným problémem byla implementace kódu pro přihlášení pomocí sociální sítě Facebook a emailu. Bylo stěžejní vytvořit aplikace i na

developers.facebook.com a na FireBase.com. Z obou aplikací byly staženy a nainstalovány frameworky do Xcode IDE. Frameworky obsahují balík metod, které mohou být využity pro komunikaci s databází FireBase a nebo ke komunikaci s Facebook API.

Problém nastal při implementaci kódu pro první stranu aplikace a to pro přihlášení přes Facebook. Na stránkách developers.facebook.com je dokumentace pro vývoj iOS aplikací uvedená v jazyce Objective-C. Nabízeli se dvě varianty jak problém vyřešit a to přeložit kód do jazyka Swift, a nebo využít funkce pro použití Objective-C jazyku v souborech typu \*.swift. Zvolen byl překlad Objective-C do jazyka Swift. Přihlášení přes Facebook bylo otestováno i pro zapomenuté heslo, nebo při několika násobném zadání chybného hesla. Pro přihlášení přes email, byla testována funkcionálnost, zda se uživatel zdárně zapsal do databáze. V případě chybného hesla je uživatel upozorněn pomocí UIAlertView na chybné heslo. V přihlášení přes email bylo testováno zadání chybného hesla, nezadání hesla, nezadání emailu a nezadání emailu ani hesla.

Na druhé straně aplikace byli vytvořeny funkce pro naplnění buňky TableViewControlleru z databáze FireBase. Z databáze se získávají data v podobě „Snapshotů“, neboli JSON objektů generovaných z databáze. Jediným problémem spojeným s generovanými daty z databáze byly obrázky. Obrázky musí být do buňky TableViewControlleru naplněny pomocí frameworku Alamofire, který zabezpečil zobrazení obrázku v definovaném tvaru. Aplikace recepty hierarchicky řadí vertikálně. Pro ideální chod aplikace byl zvolen počet deseti buněk na stránku.

Ve třetí straně aplikace byla vytvořena funkce na výpočet ideálního jídla. Parametry jsou získány z vyplněných textových polí uživatelem. Pro výpočet ideálního jídla byl použit vzorec pro výpočet BMR a následně na základě něho byl proveden výpočet pro každou dílčí surovinu v jídle.

## 5.6 Shrnutí a porovnání s alternativami

Vytvořená aplikace dává uživateli možnost vytvořit si příklad ideálního denního jídla a dále jsou mu poskytnuty recepty zdravé výživy. V porovnání aplikací s konkurencí jsem vybral zástupce Fitrecepty, tato aplikace produkuje pouze recepty, možnost sestavení příkladu ideálního jídla zde chybí. Dalším možným konkurentem jsou Kalorické tabulky, ve kterých lze po zadaných údajích sestavit vlastní jídelníček, tak aby splňoval uživateli kritéria. Při sestavování jídla musí uživatel volit potraviny a vypočítat jejich hodnoty tak aby splnil denní cíl. Obě výše zmíněné aplikace jsou dostupné v obchodu s aplikacemi AppStore. Proto bylo v bakalářské práci vytvořena aplikace, která spojuje tyto dvě zmíněné aplikace Kalorické tabulky a Fitrecepty, tak aby pro uživatele bylo

snadné zjistit příklad vhodného denního jídla. Pro výpočet ideálního jídla je dostupná webová stránka <http://scoobysworkshop.com/> [26], ve které je uživateli umožněna podobná funkcionality jako je ve vytvořené aplikaci bakalářské práce, bohužel její mobilní aplikace není zatím k dispozici.

Vytvořená aplikace je vhodná pro její další rozvoj, tzn. přidání dalších funkcionalit nebo rozšíření lokalizace do dalších světových jazyků např. angličtina a němčina.

## 6 Závěr

Mobilní aplikace jsou v dnešní době trendem, který jde neúprosně kupředu. Společně s designem aplikací se vyvíjí zároveň funkcionalita. Například není tomu dávno, kdy se v mobilních zařízeních začali komerčně používat čtečky otisku prstů, využívané k vyšší ochraně zařízení, či možnost rozdělat aktivitu na jednom zařízení a dokončit je na druhém. Tak jako se vyvíjí mobilní aplikace, tak se vyvíjí i jejich vytváření a to v podobě, buď nativního, nebo hybridního vývoje. V této bakalářské práci byla nastíněna problematika jak nativního tak hybridního vývoje a jejich klady a zápory. Potenciální vývojáři se mohou na základě zjištěných poznatků rozhodnout, jakou metodu vývoje použijí a případně co k tomu budou potřebovat, v porovnávání byli posouzeny tři vývojové prostředí PhoneGap, Xcode IDE a IDE Xamarin. V porovnávání byli brány v potaz kritéria spojená s problematikou vývoje mobilních aplikací, jako jsou doba vývoje aplikace, vzhled a funkčnost, podpora, cena a náročnost. V této bakalářské práci bylo použito vývojové prostředí Xcode od společnosti Apple, jakožto zástupce nativního vývoje, společně se zvoleným poměrně mladým programovacím jazykem Swift. Před začátkem implementace kódu bylo zapotřebí vytvořit logický a poté grafický návrh aplikace, ve kterých byly rozmístěny prvky a nastíněna potřebná funkčnost aplikace. Grafické zpracování aplikace bylo zvoleno po vzoru konkurenčního systému Android pro mobilní zařízení a to pomocí Material designu. Aplikace poskytuje uživateli seznam receptů určené pro zdravý životní styl a také poskytuje možnost vygenerovat sestavení ideálního denního jídla, pomocí kterého se mohou řídit při sestavování vlastního zdravého jídelníčku. Aplikace je určena pro orientaci ve zdravém životním stylu.

## 7 Seznam použitých zdrojů

1. iOS 9. *Apple (CZ)*. [online]. [cit. 2016-03-14]. Dostupné z: <https://www.apple.com/cz/ios/>
2. iOS technologie. *iOS Developer Library*. [online]. 17.9.2016 [cit. 2016-03-14]. Dostupné z: [https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple\\_ref/doc/uid/TP40007898-CH1-SW1](https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1)
3. Apple Inc.. *The Swift Programming language* [online]. 9.6.2015. [cit. 2016-03-14]. Dostupné z: <https://itunes.apple.com/us/book/swift-programming-language/id1002622538?mt=11>
4. Úvod do Material designu. *Google design guidelines*. [online]. [cit. 2016-03-14]. Dostupné z: <https://www.google.com/design/spec/material-design/introduction.html>
5. David Grebeň. Kompletní historie iOS: od prvního iPhoneu až po iOS 9. *Letem světem applem*. [online]. 6.3.2016 [cit. 2016-03-14]. Dostupné z: <https://www.letemsvetemapplem.eu/2016/03/06/kompletni-historie-ios/>
6. Michal Pavlíček. Android drží spolu s iOS více než 96% podíl mezi operačními systémy. *Mobilenet.cz*. [online]. 30.1.2015 [cit. 2016-03-14]. Dostupné z: <http://mobilenet.cz/clanky/android-spolu-s-ios-drzi-vice-nez-96-podil-mezi-operacnimi-systemy-18872>
7. Michal Pavlíček. Apple hlásí rekordní zisk i prodeje iPhoneů. *Mobilenet.cz*. [online]. 27.01.2016 [cit. 2016-03-14]. Dostupné z: <http://mobilenet.cz/clanky/apple-hlasi-rekordni-zisk-i-prodeje-iphonu-29336>
8. Martin Fajmon. Průměrná cena prodaného iPhoneu je 687 dolarů, Android ovlivňuje Čína. *Mobilenet.cz*. [online]. 04.02.2015 [cit. 2016-03-14]. Dostupné z: <http://mobilenet.cz/clanky/prumerna-cena-prodaneho-iphonu-je-687-dolaru-android-ovlivnuje-cina-18914>
9. Náзор servismanů iOS a Android zařízení: V čem je iOS lepší než Android?. *App-magazin*. [online]. 10.4.2013 [cit. 2016-03-14]. Dostupné z: <http://www.app-magazin.cz/slider/nazor-servismana-ios-a-android-zarizeni-v-cem-je-ios-lepsi-nez-android/>
10. Hammad Saleem. Windows 10 Mobile running on 7 percent devices. *App-magazin*. [online]. 27.11.2015 [cit. 2016-03-14]. Dostupné z: <http://www.winbeta.org/news/windows-10-mobile-running-7-percent-devices-says-adduplex>
11. Ryan Moore. Google Play tops Apple in new Apps & Developers. *Android Coliseum*. [online]. 13.1.2015 [cit. 2016-03-14]. Dostupné z: <http://www.androidcoliseum.com/2015/01/google-play-tops-apple-in-new-apps.html>



12. Cocoa (Touch). *iOS Developer Library*. [online]. 21.10.2015 [cit. 2016-03-14]. Dostupné z: <https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html>
13. Vrstva Core Services. *iOS Developer Library*. [online]. 17.9.2014 [cit. 2016-03-14]. Dostupné z: <https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.html>
14. Core OS vrstva. *iOS Developer Library*. [online]. 17.9.2014 [cit. 2016-03-14]. Dostupné z: <https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html>
15. LLDB a Xcode. *iOS Developer Library*. [online]. 18.9.2013 [cit. 2016-03-14]. Dostupné z: [https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/gdb\\_to\\_lldb\\_transition\\_guide/document/Introduction.html#//apple\\_ref/doc/uid/TP40012917](https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/gdb_to_lldb_transition_guide/document/Introduction.html#//apple_ref/doc/uid/TP40012917)
16. Swift – shrnutí. *Apple Developer*. [online]. 2016 [cit. 2016-03-14]. Dostupné z: <https://developer.apple.com/swift/>
17. Swift je podobný C#. *qiniudn.com*. [online]. [cit. 2016-03-14]. Dostupné z: <http://swiftcomparsion.qiniudn.com/>
18. iOS 8 – Co je nového?. *Apple (CZ)*. [online]. [cit. 2016-03-14]. Dostupné z: <https://www.apple.com/cz/ios/whats-new/>
19. iOS Guide – FireBase. *FireBase*. [online]. [cit. 2016-03-14]. Dostupné z: <https://www.firebase.com/docs/ios/guide/>
20. Plat iOS vývojářů. *PayScale*. [online]. 12.1.2016 [cit. 2016-03-14]. Dostupné z: [http://www.payscale.com/research/US/Job=iOS\\_Developer/Salary](http://www.payscale.com/research/US/Job=iOS_Developer/Salary)
21. Plat webových vývojářů. *PayScale*. [online]. 12.1.2016 [cit. 2016-03-14]. Dostupné z: [http://www.payscale.com/research/US/Job=Web\\_Developer/Salary](http://www.payscale.com/research/US/Job=Web_Developer/Salary)
22. Plugin APIs. *Adobe PhoneGap*. [online]. 2015 [cit. 2016-03-14]. Dostupné z: <http://docs.phonegap.com/plugin-apis/>
23. Úvod do programovacího jazyku C# a frameworku .NET. *Microsoft – Developer Network*. [online]. 2016 [cit. 2016-03-14]. Dostupné z: <https://msdn.microsoft.com/cs-cz/library/z1zx9t92.aspx>
24. Alamofire. *GitHub*. [online]. 27.2.2016 [cit. 2016-03-14]. Dostupné z: <https://github.com/Alamofire/Alamofire>
25. Kolik jídla bychom měli sníst. *MTE*. [online]. [cit. 2016-03-14]. Dostupné z: <http://www.mte.cz/stravovani-kolik-jidla.htm>

26. Free Fat Loss Calculator. *<http://scoobysworkshop.com/calorie-calculator/>*. [online]. 2015 [cit. 2016-03-14]. Dostupné z:*<http://scoobysworkshop.com/calorie-calculator/>*

## **8 Seznam obrázků**

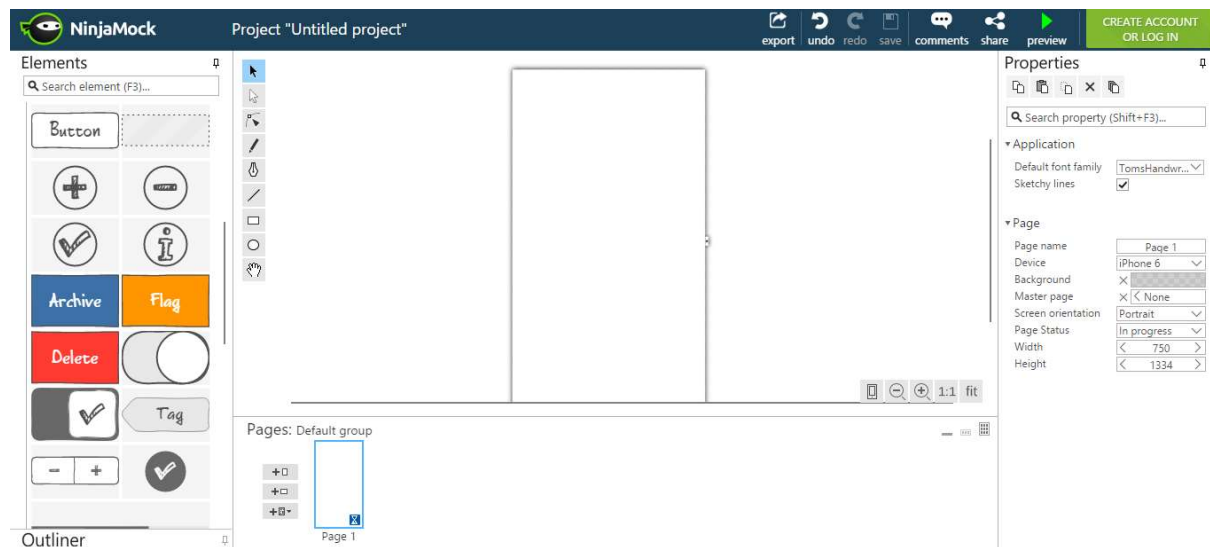
Graf 3-1 Počet vývojářů mezi obchody s aplikacemi .....	10
Obrázek 3-2 - Základní vrstvy iOS .....	12

## **9 Seznam tabulek**

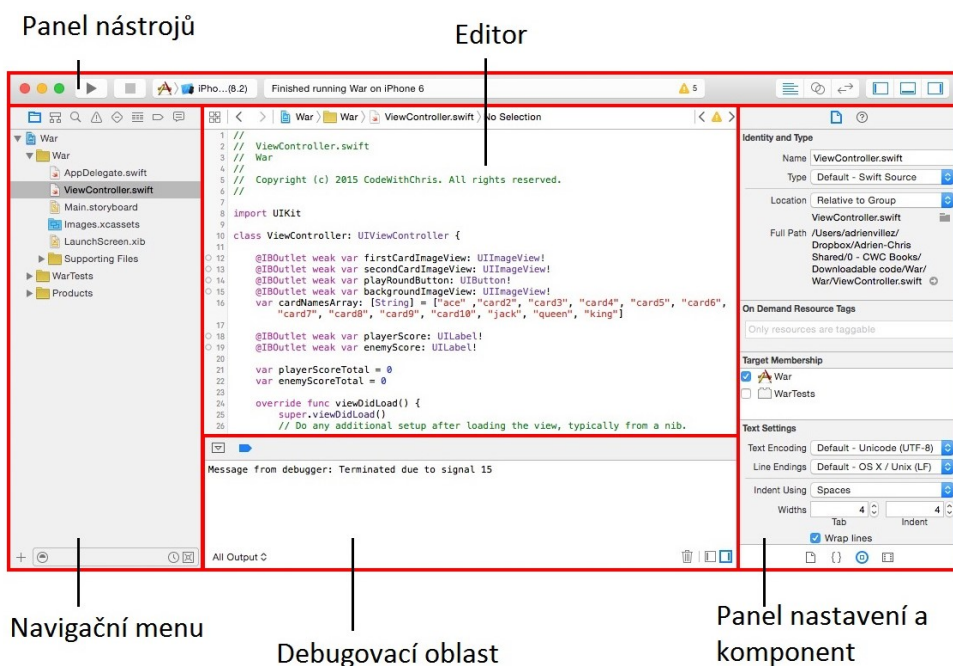
Tabulka 1 - Počet prodaných kusů .....	8
Tabulka 3 - Porovnání vývojových prostředí.....	34
Tabulka 4 seznam pohybových aktivit.....	41

## 10 Přílohy

### Příloha č.1



### Příloha č. 2



### Příloha č.3



## Příloha č.4

### SWIFT

```
extension Double {
    var km: Double { return self * 1_000.0 }
    var m: Double { return self }
    var cm: Double { return self / 100.0 }
    var mm: Double { return self / 1_000.0 }
    var ft: Double { return self / 3.28084 }
}
let oneInch = 25.4.mm
println("One inch is \(oneInch) meters")
// prints "One inch is 0.0254 meters"
let threeFeet = 3.ft
println("Three feet is \(threeFeet) meters")
// prints "Three feet is 0.914399970739201 meters"
```

### C#

```
public static class DoubleExtension {
    public static double km(this double number) {
        return number * 1000.0;
    }

    public static double m(this double number) {
        return number;
    }

    public static double cm(this double number) {
        return number / 100.0;
    }

    public static double mm(this double number) {
        return number / 1000.0;
    }

    public static double ft(this double number) {
        return number / 3.28084;
    }
}

val oneInch = (25.4).mm();
Console.WriteLine("One inch is {0} meters", oneInch);
// prints "One inch is 0.0254 meters"
var threeFeet = (3.0).ft();
Console.WriteLine("Three feet is {0} meters", threeFeet);
// prints "Three feet is 0.914399970739201 meters"
```

## Příloha č. 5

<key>CFBundleURLTypes</key>

<array>

<dict>

<key>CFBundleURLSchemes</key>

<array>

<string>fb1504903686483540</string>

</array>

</dict>

</array>

<key>FacebookAppID</key>

<string>1504903686483540</string>

<key>FacebookDisplayName</key>

<string>Fit-life</string>

```
<key>NSAppTransportSecurity</key>

<dict>

  <key>NSExceptionDomains</key>

  <dict>

    <key>facebook.com</key>

    <dict>

      <key>NSIncludesSubdomains</key>

      <true/>

      <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>

      <false/>

    </dict>

    <key>fbcdn.net</key>

    <dict>

      <key>NSIncludesSubdomains</key>

      <true/>

      <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>

      <false/>

    </dict>

    <key>akamaihd.net</key>

    <dict>

      <key>NSIncludesSubdomains</key>

      <true/>

      <key>NSThirdPartyExceptionRequiresForwardSecrecy</key>

      <false/>

    </dict>

  </dict>

</dict>
```

```

    </dict>

</dict>

<key>LSApplicationQueriesSchemes</key>

<array>

    <string>fbapi</string>

    <string>fb-messenger-api</string>

    <string>fbauth2</string>

    <string>fbshareextension</string>

</array>

```

## Příloha č. 6

```

1. import FBSDKCoreKit
2. import FBSDKLoginKit
3. class AppDelegate : UIResponder, UIApplicationDelegate {
4.     func application(application: UIApplication,
5.         didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
6.
7.         return FBSDKApplicationDelegate.sharedInstance()
8.             .application(application, didFinishLaunchingWithOptions: launchOptions)
9.     }
10.    func applicationDidBecomeActive(application: UIApplication) {
11.        FBSDKAppEvents.activateApp()
12.    }
13.    func application(application: UIApplication, openURL url: NSURL,
14.        sourceApplication: String?, annotation: AnyObject?) -> Bool {
15.        return FBSDKApplicationDelegate.sharedInstance()

```



```

16.     .application(application, openURL: url,
17.         sourceApplication: sourceApplication, annotation: annotation)
18.     }
19. }

```

## Příloha č.7

Uživatelské jméno

Váha:  Kg      Věk:  Let


Výška:  cm     


Zvolte si svůj cíl:

Lorem ipsum dolor sit er elit lamet, consectetur cillum adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

## Příloha č.8

Přihlášení/registrace 



## Příloha č.9

```
override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}

@IBAction func attemptLogin(sender: UIButton!) {

    //Make sure there is an email and a password
    if let email = emailField.text where email != "", let pwd = passwordField.text where pwd != "" {
        DataService.ds.REF_BASE.authUser(email, password: pwd) { error, authData in

            if error != nil {
                print(error.code)

                if error.code == STATUS_ACCOUNT_NOEXIST {
                    DataService.ds.REF_BASE.createUser(email, password: pwd,
                        withValueCompletionBlock: { error, result in
                            if error != nil {
                                self.showErrorAlert("Could not create account", msg: "Problem creating account. Try something else")
                            } else {
                                let uid = result["uid"] as? String
                                UserDefaults.standardUserDefaults().setValue(uid, forKey: KEY_UID)

                                DataService.ds.REF_BASE.authUser(email, password: pwd, withCompletionBlock: {
                                    error, authData in

                                        //Store what type of account this is
                                        let user = ["provider": authData.provider!]
                                        DataService.ds.createFirebaseUser(uid!, user: user)

                                    })
                                self.performSegueWithIdentifier("loggedIn", sender: nil)
                            }
                        })
                } else {
                    self.showErrorAlert("Error login in", msg: "Could not log in. Check your username and password")
                }

            } else {
                self.performSegueWithIdentifier("loggedIn", sender: nil)
            }
        }
    } else {
        showErrorAlert("Email & Password Required", msg: "You must enter an email address and a password")
    }
}

func showErrorAlert(title: String, msg: String) {
    let alert = UIAlertController(title: title, message: msg, preferredStyle: UIAlertControllerStyle.Alert)
    let action = UIAlertAction(title: "Ok", style: UIAlertActionStyle.Default, handler: nil)
}
```

## Příloha č.10

