

Nativní iOS aplikace s využitím GPS - Lodní deník

Native iOS GPS Application - Log-book

Libor Kobyda

Bakalářská práce
2014



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Libor Kobyda**
Osobní číslo: **A11124**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **prezenční**

Téma práce: **Nativní iOS aplikace s využitím GPS – Lodní deník**

Zásady pro vypracování:

1. Zpracujte literární rešerši na téma mobilní operační systém iOS a tvorba nativních mobilních aplikací.
2. Stručně popište jazyk Objective-C a postup návrhu GUI nativní iOS aplikace pomocí tzv. Storyboards.
3. Navrhňte GUI aplikace Lodní deník a implementujte aplikační logiku jednotlivých pohledů.
4. Naprogramujte také funkce pro záznam trasy pomocí GPS souřadnic.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MARK, Dave a Jeff LAMARCHE. iPhone SDK: průvodce vývojem aplikací pro iPhone a iPod touch. Vyd. 1. Brno: Computer Press, 2010, 480 s. ISBN 978-80-251-2820-6.**
2. **VÁVRŮ, Jiří. iPhone: vývoj aplikací. 1. vyd. Praha: Grada, 2012, 179 s. Průvodce (Grada). ISBN 978-80-247-4457-5.**
3. **KOCHAN, Stephen G. Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone. Vyd. 1. Brno: Computer Press, 2010, 550 s. ISBN 978-80-251-2654-7.**
4. **IOS Developer Library [online]. 2014 [cit. 2014-02-03]. Dostupné z: <https://developer.apple.com/library/ios/navigation/index.html>**
5. **IOS Source Code Examples. iOS App Development Libraries, Controls and Examples – Open Source iPhone/iPad Apps – Code4App.net [online]. 2012 [cit. 2014-02-03]. Dostupné z: <http://code4app.net>**

Vedoucí bakalářské práce:

Ing. Radek Vala

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

28. února 2014

Termín odevzdání bakalářské práce:

13. června 2014

Ve Zlíně dne 28. února 2014



prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Cieľom bakalárskej práce bolo naprogramovať funkčnú aplikáciu pre operačný systém iOS. Tento operačný systém vytvorila firma Apple a používa ho vo svojich mobilných zariadeniach. Výsledkom práce je teda aplikácia Lodný denník, ktorá má za úlohu nahradiť papierovú formu lodného denníka a zároveň urýchliť prácu kapitánovi lode.

Klíčová slova:

iOS, Xcode, ObjectiveC, Apple, iPhone, GPS

ABSTRACT

The goal of this thesis was to program a functional app for iOS operating system. This operating system was created by Apple Inc. and is used in their mobile devices. The result of this work is a logbook application, which is supposed to replace paper-based logbook and to speed up the work for the captain of the ship.

Keywords:

iOS, Xcode, ObjectiveC, Apple, iPhone, GPS

Týmto by som chcel poďakovať vedúcemu práce Ing. Radkovi Valovi za jeho čas a ochotu pri vedení tejto práce.

„Váš čas je obmedzený, tak ho nemíňajte na život niekoho iného. Nenechajte sa ovplyvniť názormi iných. Prehlušia váš vnútorný hlas. A čo je najdôležitejšie, nájdite v sebe odvahu nasledovať srdce a vlastnú intuíciu. Ony totiž už dopredu vedia čím sa skutočne chcete stať. Všetko ostatné je druhotné.“ [Steve Jobs]


Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo - bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně


.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČASŤ	10
1 JAZYK OBJECTIVE-C	11
1.1 DÁTOVÉ TYPY	11
1.1.1 Kvalifikátory	11
1.2 TRIEDA, OBJEKT, METÓDA.....	12
1.2.1 Trieda	12
1.2.2 Objekt.....	12
1.2.3 Metóda.....	12
1.2.4 Založenie projektu v prostredí Xcode	12
1.2.5 Príklad	13
1.3 DEDIČNOSŤ	14
1.3.1 Často používané objekty	14
1.4 KATEGÓRIE A PROTOKOLY	14
1.4.1 Kategórie	14
1.4.2 Protokoly	15
2 COCOA	16
2.1 COCOA TOUCH.....	16
2.1.1 UIKit	16
2.1.2 Core Animation.....	17
2.1.2.1 OpenGL ES.....	17
2.1.2.2 Core Image.....	17
2.1.3 Core Audio	17
2.1.3.1 AirPlay	17
2.1.4 Core Data	18
2.1.5 Core Location.....	18
2.1.5.1 Základné funkcie Core Location.....	19
3 IPHONE SDK	21
3.1 IBOULET	21
3.2 IBACTION	21
4 POPIS IOS	22
4.1 DESIGN IOS.....	22
4.1.1 Vrstvy.....	23
4.1.1.1 Control Center.....	23
5 APPLE IPHONE	24
II PRAKTICKÁ ČASŤ	25
6 POŽIADAVKY NA PROGRAMOVANIE PRE IOS	26
6.1 XCODE	26
6.2 REÁLNE ZARIADENIE	27
7 APLIKÁCIA LODNÝ DENNÍK	28

7.1	DIAGRAM POUŽITIA	28
7.2	STORYBOARD	28
7.3	IKONA A LAUNCH IMAGE APLIKÁCIE	30
7.3.1	Ikony	30
7.3.2	Launch image	31
7.4	UŽÍVATELSKÝ POPIS PROGRAMU	31
7.4.1	Info	31
7.4.2	Prevodník	32
7.4.3	Databáza	33
7.4.3.1	Plavba.....	33
7.4.3.2	Etapa	34
7.4.3.3	Lodný denník	35
7.4.4	GPS	35
7.5	ZÁKLADNÉ ČASTI PROGRAMU	36
7.5.1	Info	36
7.5.2	Prevodník	38
7.5.2.1	Tvorba dynamickej tabuľky.....	40
7.5.3	GPS	43
7.5.4	Databáza	46
7.5.4.1	Plavba.....	50
7.5.4.2	Etapa	52
7.5.4.3	Lodný denník	53
	ZÁVER	56
	ZOZNAM POUŽITEJ LITERATÚRY	57
	ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK.....	58
	ZOZNAM OBRÁZKOV	59
	ZOZNAM TABULIEK	62
	ZOZNAM PRÍLOH.....	63

ÚVOD

Vďaka tomu, že GPS je dostupné širokej verejnosti je možné túto technológiu využívať na monitorovanie akýchkoľvek dopravných prostriedkov, na navigáciu jak turistickú, tak aj cestnú. GPS signál je dostupný po celom svete a jeho využívanie je zdarma.

Cieľom tejto práce bolo vytvoriť aplikáciu, ktorej úlohou je na základe vstupov vytvoriť lodný denník. Smartphony, ktoré sú dnes bežne dostupné obsahujú GPS a mnoho ďalších senzorov, preto je nimi možné nahradiť niektoré drahé zariadenia. V nasledovné kapitoly sú venované vysvetleniu základných princípov programovania pre operačný systém iOS. V praktickej časti sú potom podrobne popísané požiadavky pre tvorbu aplikácií na operačný systém iOS a následne tvorba konkrétnej aplikácie od základných funkcií, až po zložitejšie.

I. TEORETICKÁ ČASŤ

1 JAZYK OBJECTIVE-C

Jazyk Objective-C bol vytvorený v 80. rokoch 20. storočia. Objective-C vychádza z jazyka C, ale je doplnený o rôzne funkcie, ktoré umožňujú tvorbu objektov a manipuláciu s nimi. Tento programovací jazyk si licencovala firma NEXT, ktorá vyvinula vývojové prostredie NEXTSTEP. Dnes je Objective-C základom OS X a iOS. [1]

1.1 Dátové typy

- **int** - Dátový typ `int` používame pre ukladanie celočíselnej hodnoty.
- **float** - Používame pre ukladanie hodnôt s desatinnou čiarkou.
- **double** - Je podobný typu `float`, ale používa sa vtedy, keď už nedostačuje rozsah čísel, ktoré nám ponúka dátový typ `float`.
- **char** - Dátový typ `char` používame pre ukladanie jedného znaku do premennej.
- **id** - Tento dátový typ sa používa pre ukladanie objektu ľubovoľného typu. V podstate sa jedná o obecný objektový typ. [2]

1.1.1 Kvalifikátory

Samozrejme môžeme použiť aj takzvané kvalifikátory, ktoré premennú viac špecifikujú na určitú situáciu.

- **long** - Ak použijeme napríklad premennú `long int pokus`, potom bude mať premenná `pokus` rozsah do 32 bitov. (V niektorých operačných systémoch sa môže veľkosť premennej líšiť).
- **longlong** - Pri tomto kvalifikátore je možné premennú `int` rozšíriť minimálne na 64-bitovú premennú.
- **short** - Kvalifikátor `short` sa používa vtedy, keď potrebujeme šetriť pamäť. Ak použijeme `short int pokus`, potom bude mať premenná `pokus` veľkosť minimálne 16 bitov.
- **unsigned** - Kvalifikátorom `unsigned` špecifikujeme premennú, do ktorej je možné zadať len kladnú hodnotu.
- **signed** - Kvalifikátor `signed` označuje premennú, do ktorej je možné zapísať hodnotu s rôznym znamienkom. [3]

1.2 Trieda, objekt, metóda

Nasledovné kapitoly popisujú základne stavebné jednoty objektového programovania.

1.2.1 Trieda

Trieda je obecná definícia objektu. Ak si spravíme napríklad triedu auto, bude mať všeobecné vlastnosti auta a zároveň sa tu budú definovať všeobecné metódy pre každé auto. [4]

1.2.2 Objekt

Objekt je určitá vec, ktorá predstavuje predmet činnosti. Napríklad môžeme mať objekt moje Auto, s ktorým môžem vykonávať určité operácie, ako napríklad: tankovať, umývať alebo riadiť. [5]

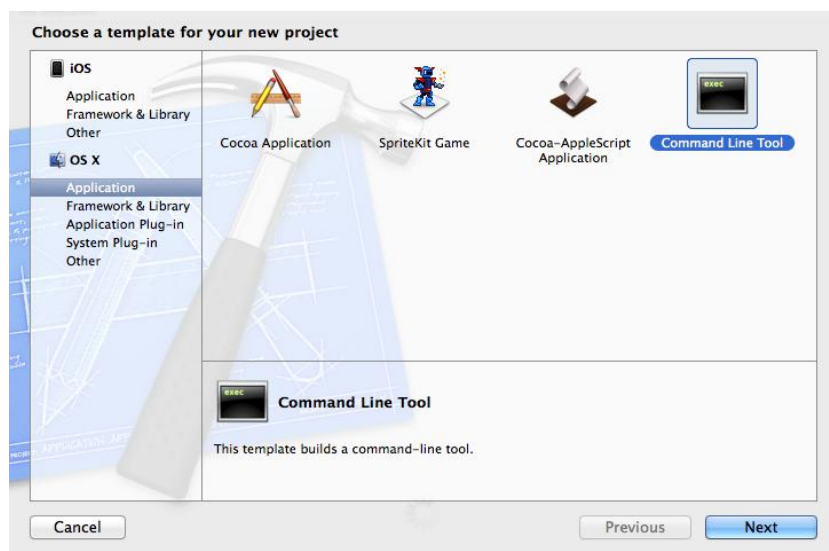
1.2.3 Metóda

Metóda je funkcia, ktorá vykonáva určitú operáciu s objektom. Napríklad ak máme objekt mojZlomok, je možné pomocou metódy zadajCitatel zadať čitateľ do objektu mojZlomok. Jazyk Objective-C má nasledujúci zápis volania metód. [6]

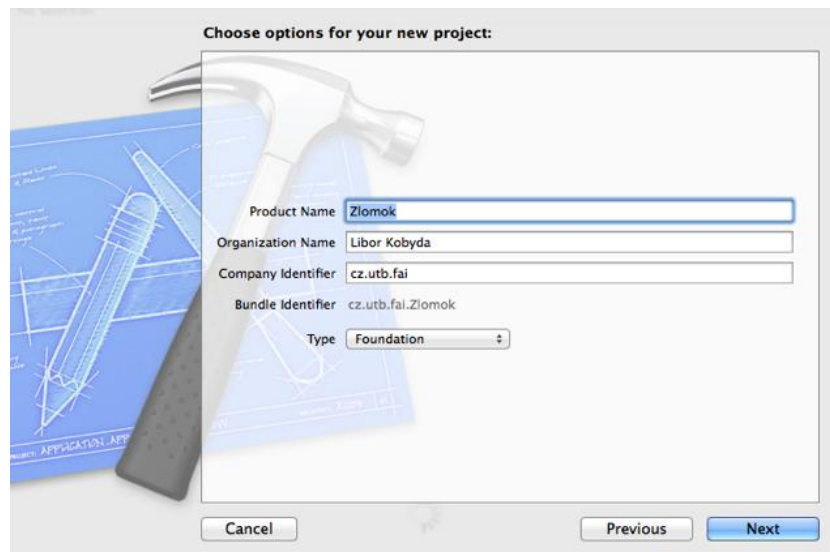
[TriedaAleboInštancia metóda];

[mojZlomok zadajCitatel:4];

1.2.4 Založenie projektu v prostredí Xcode



Obrázok 1: Založenie nového projektu



Obrázok 2: Výber typu nového projektu

Pre otestovanie nasledujúceho kódu je potrebné si založiť nový projekt tak ako bolo naznačené na Obrázku 1 a Obrázku 2.

1.2.5 Príklad

```

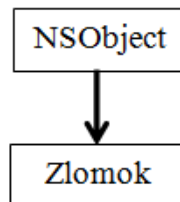
9  #import <Foundation/Foundation.h>
10
11 @interface Zlomok : NSObject{           // Definícia triedy Zlomok, ktorý bude potomkom triedy NSObject
12     int citatel;                       // Definícia instančných premenných
13     int menovatel;                     // Premenné s ktoré bude každý objekt Zlomok obsahovať
14 }
15 -(void)vypis;                          // Metódy, ktoré sa dajú používať pri práci s každým objektom Zlomok
16 -(void)zadatCitatel:(int)n;
17 -(void)zadatMenovatel:(int)d;
18
19 @end
20
21 @implementation Zlomok
22
23 -(void) vypis{                          // Implementácia metód
24     NSLog(@"%i/%i",citatel,menovatel);
25 }
26
27 -(void)zadatCitatel:(int)n{
28     citatel = n;
29 }
30 -(void)zadatMenovatel:(int)d{
31     menovatel = d;
32 }
33
34 @end
35
36 int main(int argc, const char * argv[])
37 {
38
39     @autoreleasepool {
40         Zlomok *mojZlomok = [[Zlomok alloc]init];
41
42         [mojZlomok zadatCitatel:4];      // Volanie metódy
43         [mojZlomok zadatMenovatel:7];
44         [mojZlomok vypis];
45     }
46     return 0;
47 }

```

Obrázok 3: Trieda Zlomok [7]

1.3 Dedičnosť

Je dôležité si uvedomiť, že aj rodičovská trieda môže mať svojho rodiča. Avšak sú prípady, kedy trieda nemá svojho rodiča. To znamená, že takáto trieda je úplne na vrchole hierarchie a takúto triedu nazývame koreňová trieda. V tomto prípade je NSObject koreňovou triedou a trieda Zlomok dedí vlastnosti z rodičovskej triedy NSObject. [8]



Obrázok 4: Koreňová trieda a podtrieda

Hlavnou výhodou dedičnosti je to, že podtrieda prevezme všetky inštančné premenné a môže pristupovať k metódam rodičovskej triedy ako keby má tieto metódy sama implementované. [9]

1.3.1 Často používané objekty

- **NSString** - Objekt používaný pre ukladanie znakových reťazcov. Je veľmi výhodné ho používať, pretože obsahuje mnoho metód, ktoré uľahčujú prácu s týmto reťazcom.
- **NSArray** - Objekt, do ktorého sa dá uložiť pole.
- **NSDictionary** - Objekt, do ktorého je možné vložiť pole typu kľúč hodnota.
- Prípona **Mutable** - Ak chceme, aby bolo možné počas priebehu programu upravovať objekt, tak použijeme príponu Mutable napríklad NSMutableArray. [10]

1.4 Kategórie a protokoly

V nasledujúcich kapitolách sú popísané kategórie a protokoly.

1.4.1 Kategórie

Kategórie nám dávajú možnosť implementovať nové metódy do triedy bez toho, aby sme robili jej nového potomka alebo upravovali triedu samotnú. Toto je použiteľné hlavne pri veľkých projektoch, kde každý pracuje na svojej časti kódu. Napríklad, ak by bolo potrebné doplniť nejakú metódu do triedy Zlomok môžeme použiť kategóriu. A to nasledovne. [11]

```
57 #import "Zlomok.h"
58
59 @interface Zlomok (MenoKategorie) //Týmto riadkom definujeme novú kategóriu
60 -(void)novaMetoda; //Táto metóda bude pridaná do triedy Zlomok
61 @end
```

Obrázok 5: Príklad kategórie

1.4.2 Protokoly

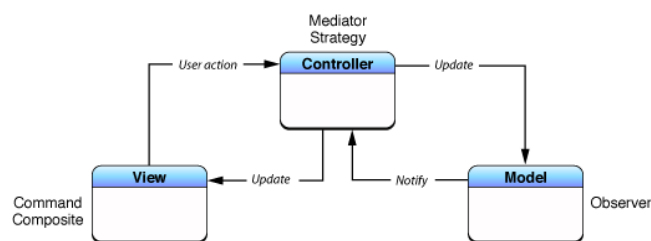
Protokol je možnosť definovať nové metódy triedy, ktoré by mali byť implementované v dcérinej triede pôvodnej triedy. [12]

2 COCOA

Cocoa je vlastne spojenie dvoch frameworkov (Foundation a AppKit) do jedného. Vďaka Cocoa vieme na pár riadkoch implementovať prácu s animáciami alebo prácu v sieti. Cocoa používa MVC model. Tento model oddeľuje od seba 3 základné časti programu, a to správcu dát, zobrazovanie a aplikačnú logiku. Vďaka MVC sa môže programátor zamerať konkrétne na jednu z týchto 3 častí. Tento framework je používaný na počítačoch MAC. [2]

2.1 Cocoa Touch

Cocoa Touch je zjednodušená a upravená forma frameworku Cocoa tak, aby bola schopná fungovať na mobilnom zariadení (iPhone, iPod, iPad). Na mobilných zariadeniach od Apple je zjednodušená forma OS X. Cocoa Touch pridáva podporu napríklad pre GPS, dotykové ovládanie, akcelerometer a ďalší hardware, ktorý na počítači MAC nenájdeme. Základným rozdielom od Cocoa je to, že Cocoa Touch obsahuje UIKit namiesto AppKit, ktorý je určený len pre počítače MAC. [3]



Obrázok 6: Zobrazenie MVC [3]

2.1.1 UIKit

UIKit je framework, ktorý obsahuje funkcie umožňujúce kontrolovať používateľské rozhranie iOS aplikácie. Toto prostredie je na rozdiel od desktopovej verzie AppKit upravené tak, aby sa dalo pohodlne ovládať pomocou dotyku. Medzi najviac používané prvky užívateľského rozhrania patria napríklad: UILabel, UITextView, UIButton a mnoho ďalších. Tieto všetky prvky majú prednastavenú grafiku, ktorú ale môžeme zmeniť. Tieto zmeny sa dajú robiť buď v grafickom rozhraní cez Storyboard (viz. kapitola..), alebo priamo v kóde aplikácie - tento spôsob poskytuje oveľa väčšie možnosti úprav. [3]

2.1.2 Core Animation

Core Animation je knižnica napísaná v Objective-C, pomocou ktorej je možné vytvárať dynamické grafické animácie užívateľského rozhrania aplikácie. [3]

2.1.2.1 OpenGL ES

Na bežné programy je dostačujúce UIKit, ale ak bude potrebné vytvoriť graficky prepracovanejšiu aplikáciu, bude nutné použiť OpenGL ES knižnicu. Táto knižnica sa používa väčšinou na tvorbu hier alebo iných 3D programov. Je to jednoduché API veľmi podobné desktopovej verzii. OpenGL ES je podmnožinou OpenGL a je navrhnutá priamo pre mobilné zariadenia. [3]

2.1.2.2 Core Image

Pomocou Core Image je možné aplikovať na fotky, či priamo na fotoaparát rôzne efekty, ktoré sú akcelerované hardwarovo. Obsahuje taktiež funkciu na odstraňovanie červených očí alebo napríklad automatické zaostrovanie. [3]



Obrázok 7: Ukážka výsledku práce v Core Animation a s OpenGL ES [3]

2.1.3 Core Audio

Tento framework umožňuje prácu s videom a zvukom. Pomocou tohto rozšírenia Cocoa dostaneme plnú kontrolu nad prehrávaním alebo zaznamenávaním videa a zvuku. Má v sebe zabudovanú podporu pre živý stream priamo cez protokol HTTP. [3]

2.1.3.1 AirPlay

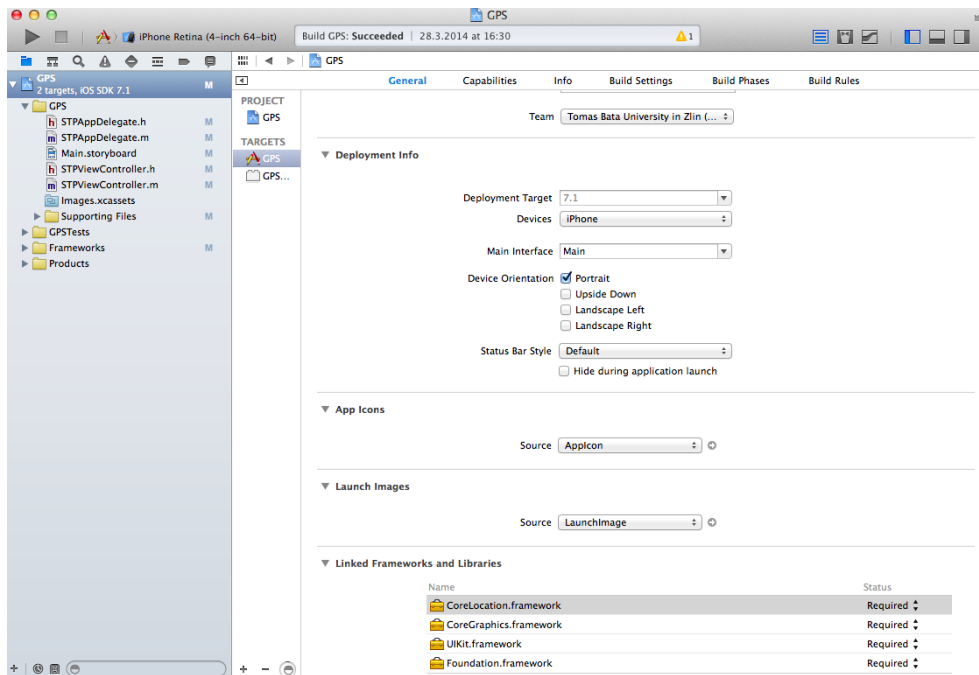
AirPlay je funkcia, ktorá umožňuje streamovať obraz z iOS zariadenia priamo do televízora. Pomocou AirPlay sa teda dá napríklad zobrazit' fotka na televízore alebo dokonca hrať nejaká hra. [3]

2.1.4 Core Data

iOS obsahuje rôzne nástroje pre ukladanie a zdieľanie dát. Je možné vytvoriť SQL Lite databázu, do ktorej si bude aplikácia ukladať dáta. Nie je to ale jediný spôsob, ako ukladať dáta. Ak nechceme tvoriť SQL Lite databázu, tak môžeme využiť napríklad súbory plist. Aplikácie tiež môžu zdieľať dáta, a to formou URL systému. Core Data je rovnako ako ostatné frameworky založený na MVC modeli, ktorý uľahčuje prácu programátorovi. Čo sa týka dát, programátor nie je ničím obmedzený. Môže si urobiť aplikáciu na ukladanie kontaktov alebo napríklad nejaký grafický program. [3]

2.1.5 Core Location

Táto bakalárska práca požadovala využitie GPS, a preto musel byť použitý framework Core Location, ktorého základné funkcie sú popísané nižšie. Tento framework umožňuje iOS zariadeniu zistiť vlastnú polohu. Túto polohu je možné získať buď pomocou GPS, trianguláciou najbližších vysieláčov, alebo pomocou WPS (pomocou WiFi). Z týchto 3 možných technológií je GPS najpresnejšia. Je treba brať ohľad na baterku, pretože určovanie polohy je pomerne náročné na energiu. Core Location automaticky vyberie najlepšiu z 3 možných technológií pre určenie polohy, a to práve podľa toho, akú presnosť určenia polohy si zvolíme. [3]



Obrázok 8: Pridanie frameworku do projektu

2.1.5.1 Základné funkcie Core Location

Ako prvé je nutné si vytvoriť objekt location Manager, ktorý bude objektom triedy CLLocationManager. Pomocou tohto objektu budeme môcť nastavovať filter, presnosť určovania polohy a volať funkcie pre zistenie polohy. Presnosť GPS určuje minimálnu presnosť, akou je potrebné zaznamenať polohu zariadenia. Pomocou filtra sa dá nastaviť, aby sa zisťovala poloha napríklad každých 50 metrov. [4]

```

18 - (void)viewDidLoad{
19     self.locationManager = [[CLLocationManager alloc] init]; // Vytvorenie objektu CLLocationManager
20     locationManager.delegate = self;
21     locationManager.desiredAccuracy = kCLLocationAccuracyNearestTenMeters; // Určenie presnosti
22     locationManager.distanceFilter = 70; // Nastavenie filtra
23     pocitadloBodovGPS = @"0";
24     [locationManager startUpdatingLocation]; // Volanie funkcie pre zistenie polohy
25     [super viewDidLoad];
26 }

```

Obrázok 9: Inicializácia objektu CLLocationManager

Prvé je nutné vytvoriť si objekt triedy CLLocation, pomocou ktorého sa dostaneme k potrebným údajom. Napríklad ak je potrebné získať informáciu o zemepisnej dĺžke stačí na objekt current zavolať metódy coordinate a longitude. Je teda možné získať informácie o zemepisnej šírke, zemepisnej dĺžke, horizontálnej presnosti, vertikálnej presnosti a napríklad aj o rýchlosti. Ak je potrebné získať informáciu uložiť, stačí si vytvoriť nový NSString, ktorému alokujeme miesto v pamäti a následne zavolať metódu initWithFormat. [4]

```

60 -(void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations{
61
62     CLLocation *current = [locations lastObject];
63
64     NSString *longitudeString = [NSString alloc] initWithFormat:@"%g°", current.coordinate.longitude];
65     NSString *latitudeString = [NSString alloc] initWithFormat:@"%g°", current.coordinate.latitude];
66     NSString *horizontalAccuracyString = [NSString alloc] initWithFormat:@"%g m", current.horizontalAccuracy];
67     NSString *verticalAccuracyString = [NSString alloc] initWithFormat:@"%g°", current.verticalAccuracy];
68
69     longitudeLabel.text = longitudeString;
70     latitudeLabel.text = latitudeString;
71     horizontalAccuracyLabel.text = horizontalAccuracyString;
72     verticalAccuracyLabel.text = verticalAccuracyString;
73
74
75     CLLocationDistance distance = [current distanceFromLocation:predoslyBod];
76     NSString *distanceString = [NSString alloc] initWithFormat:@"%g", distance];
77     NSInteger temp1 = distanceString.integerValue;
78     NSInteger temp2 = distanceString.integerValue;
79     NSInteger vysledok = temp1+temp2;
80     distanceString = [NSString alloc] initWithFormat:@"%ld m", (long)vysledok];
81     distanceLabel.text = distanceString;
82
83     predoslyBod = current;
84
85 }

```

Obrázok 10: Príklad použitia funkcie didUpdateLocations

Ako posledné dôležité pri určovaní polohy je nutné inicializovať aj funkciu, ktorá sa volá pri zistení chyby počas určovania polohy. Tieto chyby treba odchytiť, pretože inak by aplikácia mohla spadnúť. [4]

```
87 -(void)locationManager:(CLLocationManager *)manager didFailWithError:(NSError *)error{
88     NSString *chyba = (error.code == kCLErrorDenied) ? @"Pristup zamietnuty" : @"Neznáma chyba";
89
90     UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"nastala chyba gps" message:chyba delegate:nil
91                       cancelButtonTitle:@"OK" otherButtonTitles:nil];
92     [alert show];
93 }
```

Obrázok 11: Príklad použitia funkcie didFailWithError

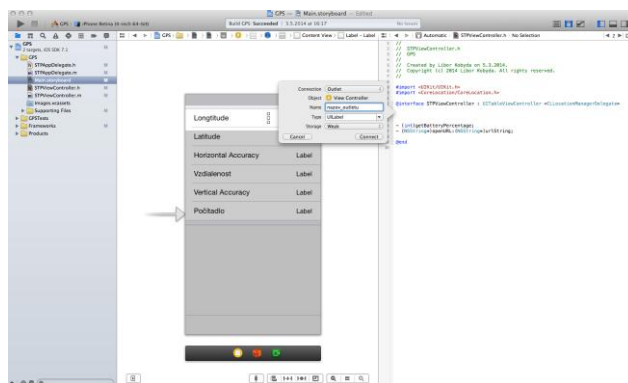
3 IPHONE SDK

iPhone SDK je súbor funkcií ktoré môžeme pri programovaní používať. Toto SDK na me-ní vždy keď Apple vydá novú verziu operačného systému iOS

3.1 IBOutlet

Toto kľúčové slovo informuje Interface Builder o tom, že sa jedná o premennú, ktorú chceme pripojiť k nejakému konkrétnemu prvku UIKitu. Táto premenná je samozrejme viazaná na konkrétnu obrazovku Storyboardu a deklaruje ju v hlavičkovom súbore. Následne prepojíme prvok UIKitu s vytvoreným outletom a cez outlet sa vieme dostať na obsah napríklad textového poľa, s ktorým sme outlet prepojili. Outlet môžeme buď napísať sami, alebo vytvoríme prvok UIKitu a outlet pre tento prvok si necháme vygenerovať. Ak chceme outlet vygenerovať, stačí, ak sa prepne do Storyboardu, v ktorom si nastavíme asistent editor, v ktorom si nájdeme hlavičkový súbor. Následne si v grafickom návrhu označíme prvok UIKitu a stlačíme klávesu ctrl a pomocou myšky prejdeme do hlavičkového súboru, kde sa nám outlet vytvorí po vyplnení jeho vlastností. Napísaný a vygenerovaný kód sú totožné. [4]

```
@property (nonatomic, retain) IBOutlet UILabel *verticalAccuracyLabel;
```



Obrázok 12: Automatické generovanie outletu

3.2 IBAction

Funguje rovnako ako outlet, až na to, že týmto kľúčovým slovom definujeme metódu, ktorá je spustiteľná pomocou prvku UIKit. Akcie môžeme tiež generovať, a to rovnakým spôsobom ako outlety. Nasledujúci príklad ukazuje vytvorenie akcie napríklad pre tlačidlo na spustenie lokalizácie. [4]

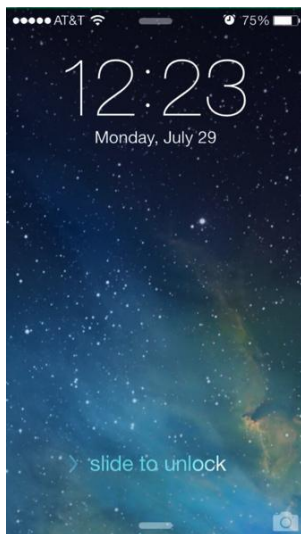
```
- (IBAction)StartLocation:(UIButton*)sender;
```

4 POPIS IOS

iOS je operačný systém, ktorý vytvoril Apple pre svoje mobilné zariadenia (iPod, iPhone, iPad). Aktuálna verzia iOS 7.1 bola vydaná 10.3.2014. Táto aktualizácia priniesla hlavne opravy chýb a úpravu užívateľského prostredia. Tento operačný systém sa vyznačuje tým, že jeho prostredie je jednoduché a prívetivé pre užívateľa. Na druhej strane iOS je uzatvorený systém, takže nie je možné inštalovať ľubovoľné aplikácie, ale len tie, ktoré sú schválené Applom. Od iOS 7 je systém aj v 64-bitovom variante, čo pomohlo k zvýšeniu svižnosti systému. [5]

4.1 Design iOS

iOS je navrhnutý tak, aby ho bolo možné používať ihneď po spustení, a to bez návodu. Apple sa snaží, aby jeho systém bol intuitívny a zábavný pri používaní. Veľkou výhodou je aj to, že tento systém je možné používať takmer bez návodu. [5]



Obrázok 13: Zamknutá obrazovka iOS [5]



Obrázok 14: Základná obrazovka iOS [5]

Na zamknutej obrazovke sa zobrazujú len základné informácie, ako čas, dátum, možnosť odomknutia a prípadne notifikácie. [5]

Na základnej obrazovke iOS sú už jednotlivé ikonky aplikácií, ktoré môže užívateľ spúšťať. Zobrazené sú samozrejme len tie základné od Applu, avšak užívateľ si môže doinštalovať akúkoľvek ďalšiu, čo sa nachádza v obchode App Store. [5]

4.1.1 Vrstvy

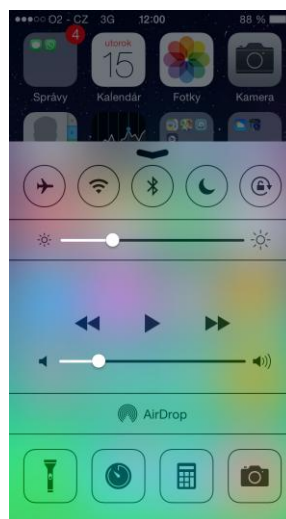
Nová verzia iOS 7 priniesla vrstvy, vďaka ktorým vzbudzuje pohľad na obrazovku 3D dojem. Ako je vidieť dole na obrázku, tak sú to v podstate maximálne 3 vrstvy, ale bežne sú vidieť len 2. 3D efekt je vytváraný pomocou gyroskopu, ktorý nakláňa vrstvu s aplikáciami a vytvára tak paralax efekt. Vďaka vrstvám je systém aj pestrejší a v podstate sa jeho farby stále menia. Toto spôsobuje priehľadnosť niektorých vrstiev. Napríklad, ak si zobrazíme notifikačnú lištu, je možné pozorovať odtiene farby nášho pozadia. [5]



Obrázok 15: Vrstvy iOS 7 [5]

4.1.1.1 Control Center

Control Center je funkcia iOS, vďaka ktorej môžeme v akejkoľvek aplikácii meniť určité nastavenia systému, a to svietivosť displeja, hlasitosť zvuku, prepínanie pesničiek, spustiť zrkadlenie, aktivovať funkciu AirDrop, a nakoniec spustiť 4 základné aplikácie. [5]



Obrázok 16: iPhone Control Center

5 APPLE IPHONE

Prvý iPhone predstavil bývalý riaditeľ spoločnosti Apple Steve Jobs, a to 9.1.2007 v San Franciscu. Postupom času každý rok Apple predstavil nový model tohto smartphonu. Dnes je už vo verzii 5S, čo je 7. generácia. [6]



Obrázok 17: Zobrazenie iPhone 5S [6]

Obal iPhone 5S je vyrobený výhradne z hliníku a skla, a kvôli tomu má telefón čistý a jednoduchý design. Podporuje štandardný 3,5mm jack pre pripojenie slúchadiel alebo iného príslušenstva. Ako dátový konektor slúži lightning, ktorý je obojstranný a ponúka rýchlosť USB. Do tohto telefónu treba vložiť nanoSIM, čo je už 3. generácia klasickej SIM karty. Ako posledné by malo byť spomenuté TouchID, čo je senzor odtlačkov prstov, ktorý zvyšuje zabezpečenie systému a zrýchľuje jeho používanie. [6]



Obrázok 18: Zobrazenie Home button s TouchID [6]

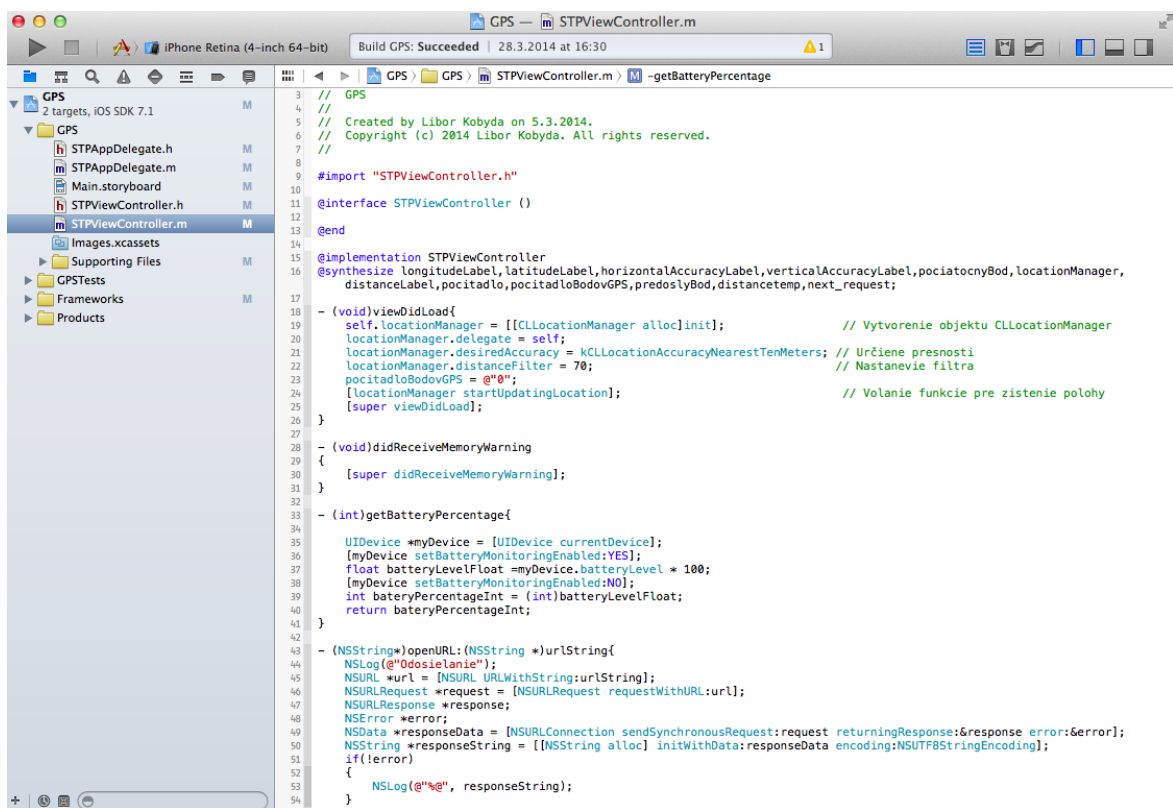
II. PRAKTICKÁ ČASŤ

6 POŽIADAVKY NA PROGRAMOVANIE PRE IOS

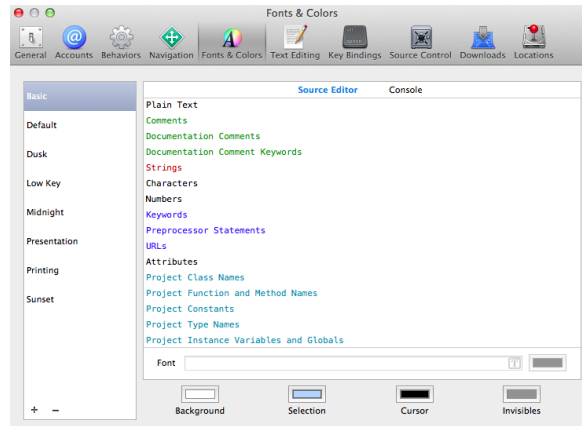
Nasledujúce kapitoly sa venujú požiadavkám na programovanie pre operačný systém iOS a OS X.

6.1 Xcode

Xcode je vývojové prostredie vyvinuté firmou Apple, ktoré slúži prevažne pre vývoj aplikácií na zariadenia OS X a iOS. Tieto aplikácie sú písané v jazyku Objective-C, avšak v Xcode je možné programovať aj v jazyku C alebo C++, čo spoľahlivo funguje. Vzhľad tohto prostredia je samozrejme nastaviteľný, a preto si ho môže programátor prispôbiť svojim potrebám. Prostredie Xcode je poskytované od polovice roku 2011 zdarma a je možné ho stiahnuť pomocou Mac App Store. Nevýhodou tohto prostredia je fakt, že pre jeho inštaláciu je nutné vlastniť hardware s operačným systémom Mac. Naopak, výhodou je to, že Apple poskytuje pravidelné aktualizácie prostredia, a hlavne to, že toto prostredie obsahuje kompletnú programátorskú dokumentáciu, či už pre iOS zariadenia, alebo aj pre počítače s OS X. [7]



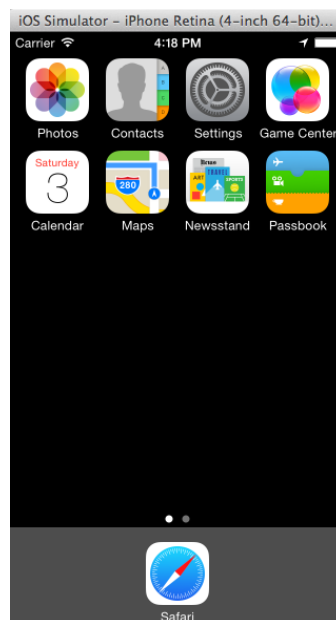
Obrázok 19: Prostredie Xcode



Obrázok 20: Možnosti nastavenia prostredia Xcode

6.2 Reálne zariadenie

Prostredie Xcode obsahuje kvalitný iOS simulátor, na ktorom sa dá testovať vyvíjaná aplikácia. Avšak simulácia nie je dokonalá, a preto ak chceme aplikáciu poriadne odladiť, je treba aplikáciu nahráť do reálneho zariadenia. Typickým prípadom, kedy je potrebné využiť reálne zariadenie, je testovanie GPS. Simulátor obsahuje len základné možnosti simulácie pohybu a činnosti GPS senzoru. [7]



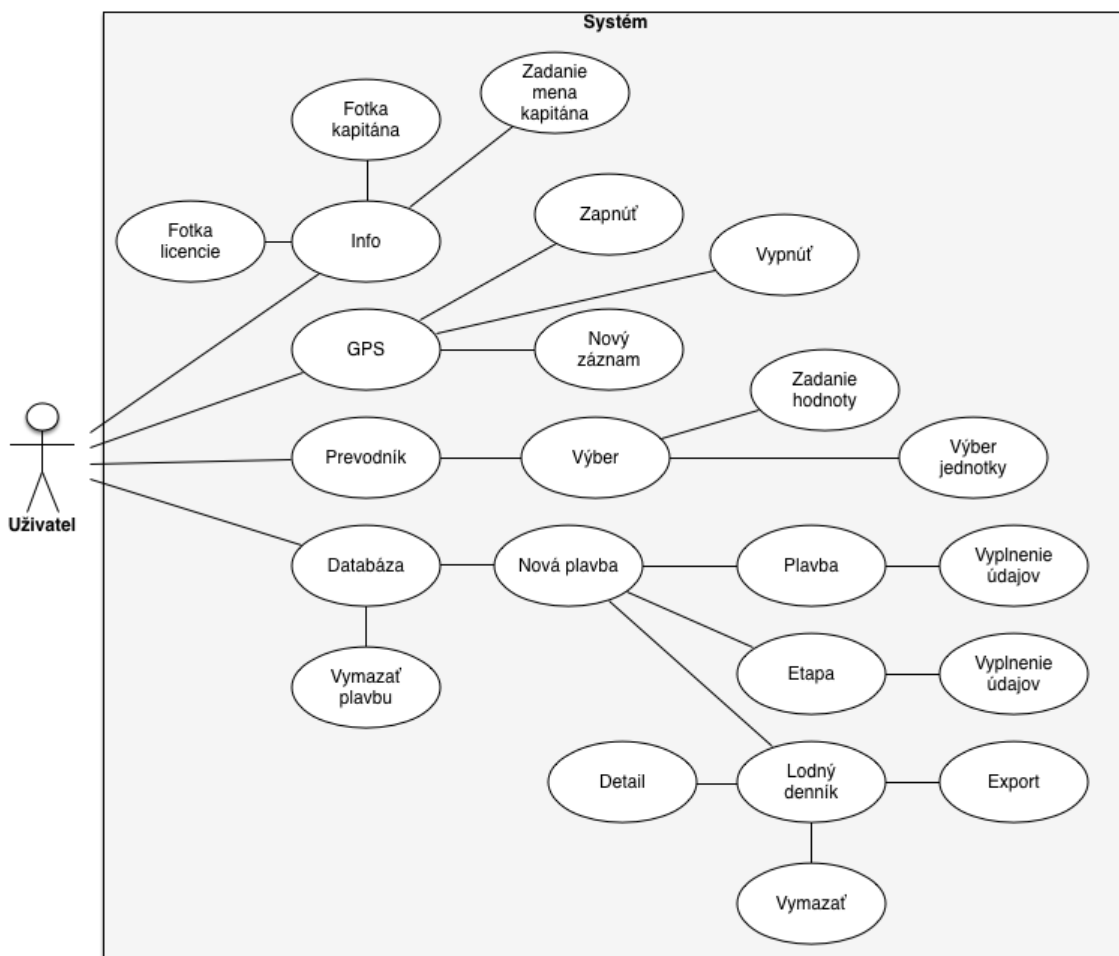
Obrázok 21: Simulátor iOS

7 APLIKÁCIA LODNÝ DENNÍK

V nasledujúcich kapitolách je popísaný Storyboard, tvorba ikoniek pre aplikáciu a na koniec programátorský popis programu.

7.1 Diagram použitia

V tejto kapitole je pomocou jazyka UML navrhnutý diagram použitia, ktorý znázorňuje možnosti užívateľa v aplikácii Lodný denník.

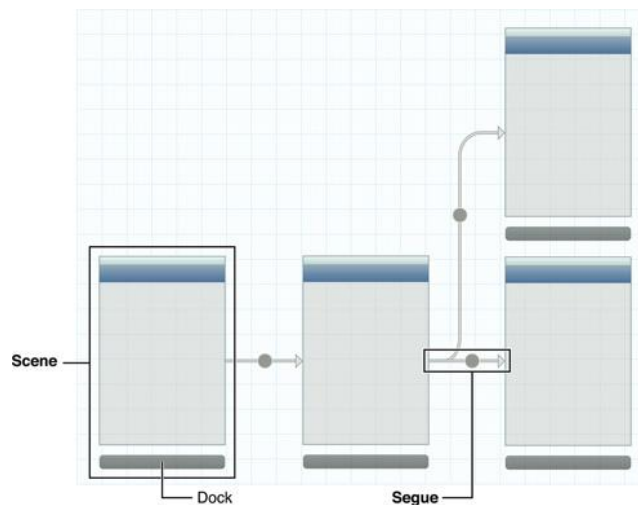


Obrázok 22: Diagram použitia aplikácie Lodný denník

7.2 Storyboard

Storyboard je vizuálna reprezentácia užívateľského rozhrania iOS aplikácie. Sú to vlastne obrazovky, na ktoré sa môže užívateľ dostať pomocou gesta alebo nejakého tlačidla. Tieto obrazovky obsahujú prvky UIKit, ktoré si programátor môže ľubovoľne umiestňovať do priestoru. Je to akási zrýchlená grafická forma tvorby vizuálnej stránky aplikácie. Medzi

obrazovkami sa nachádzajú takzvané segue, ktoré určujú nasledujúcu zobrazenú obrazovku. Tieto segue sa dajú rôzne pomenovať, čo ich umožňuje priradiť prvkom UIKitu. Takže, napríklad ak klikneme na nejaké tlačidlo, môžeme prejsť na inú obrazovku. Prechody medzi obrazovkami sú automaticky animované a prvky UIKitu majú prednastavený design po vzore iOS 7, takže po chvíli môžeme mať urobený pekný dizajn pre novú aplikáciu. Pomocou Storyboardu je tiež možné vytvárať implementácie kódu pre prvky UIKitu, takzvané IBOutlety. Súbor s príponou *.storyboard sa nachádza vo väčšine programov pre iOS a je dostupné od verzie iOS 5. Jeho úprava prebieha priamo v Xcode. [8]



Obrázok 23: Schematické rozdelenie prvkov Storyboard [9]



Obrázok 24: Storyboard aplikácie Lodný denník [9]

7.3 Ikona a launch image aplikácie

Následujúce kapitoly popisujú tvorbu hlavnej ikony, launch image a následne ich vloženie do aplikácie Lodný denník.

7.3.1 Ikony

Ikona pre aplikáciu Lodný denník bola vytvorená v programe Adobe Ilustrátor, a teda je urobená vektorovo. To má výhodu v tom, že je možné exportovať ľubovoľnú veľkosť ikony, čo aj bolo potrebné, pretože iPhone potrebuje túto ikonu až v troch veľkostiach.

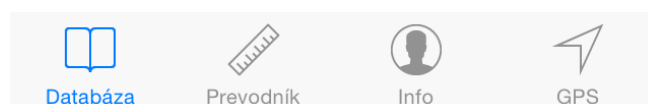
- Ikona aplikácie pre iOS 7 (120 pt)
- Ikona aplikácie pre vyhľadávanie iOS 7 (80 pt)
- Ikona aplikácie pre nastavenia (58 pt)

Táto ikona sa importuje priamo do Xcode a konkrétne do zložky Images.xcassets, ktorá sa nachádza v každom novom projekte pre iOS zariadenia.



Obrázok 25: Ikona aplikácie Lodný denník

Následne ešte bolo nutné vložiť ikonky pre Tab Bar Controller. Tieto ikonky sa vkladajú tiež do zložky Images.xcassets, v ktorej zvolíme možnosť New Image Set kde boli postupne vložené všetky potrebné ikonky.



Obrázok 26: Ikony v Tab Bar Controller [10]

Tieto ikonky sa nachádzajú na každej obrazovke a kliknutím na niektorú z nich sa okamžite dostaneme na zvolenú časť programu. Ikonky majú dva módy, a to buď aktívna, alebo neaktívna. Na obrázku 16 je aktívna ikonka databáza a značí, že sa nachádzame v databáze plavieb. Po prepnutí z databázy do prevodníka sa nám dočasne uloží aktuálna obrazovka a po návrate späť do databázy môžeme pokračovať tam, kde sme skončili. Av-

šak toto funguje len v rámci programu, ak aplikácia prejde do pozadia alebo ju vypneme, zobrazí sa nám základná obrazovka.

7.3.2 Launch image

Launch image je obrázok, ktorý je vidieť pri spustení aplikácie. Obyčajne sa skladá z nejakého jednoduchšieho pozadia, loga a názvu aplikácie. Launch image pre aplikáciu Lodný denník bol vytvorený podobne ako logo, avšak pri tvorbe tohto obrázku bol skopírovaný vektorový model z programu Adobe Illustrator, ktorý bol následne vložený do programu Pixelmator, kde bol tento obrázok dokončený do finálnej podoby.



Obrázok 27: Launch image aplikácie Lodný denník

7.4 Uživatelský popis programu

Následovné kapitoly sa zaoberajú uživatelským popisom základných obrazoviek aplikácie.

7.4.1 Info

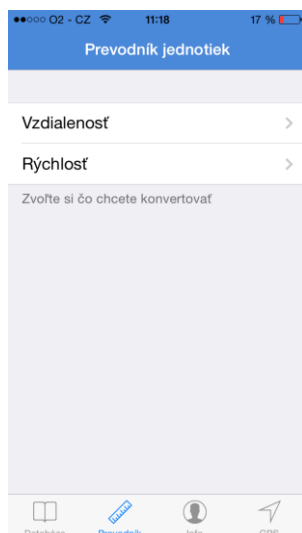
Po prvom spustení aplikácie Lodný denník sa Vám automaticky otvorí záložka Info, aby ste vyplnili meno a priezvisko kapitána. Ak ste ho vyplnili, môžete spraviť kapitánovi fotku, ktorá zostane uložená v programe. Odplávané hodiny nie sú užívateľsky editovateľné a ich hodnota sa navýši po pridanom zázname do databázy. Ak by bolo potrebné zmeniť meno kapitána, je to možné pomocou tlačidla Edit v pravom hornom rohu obrazovky Info.



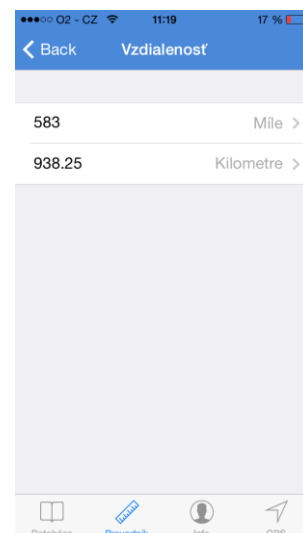
Obrázok 28: Obrazovka Info

7.4.2 Prevodník

Prevodník slúži na prevod jednotiek rýchlosti alebo vzdialenosti. Ak potrebujete konvertovať nejaké jednotky, stačí kliknúť na prevodník, kde si následne vyberiete, či chcete konvertovať jednotky vzdialenosti alebo rýchlosti. Teraz už stačí len do prvého riadku zadať hodnotu, ktorú potrebujete skonvertovať a vybrať jej jednotku. A následne v druhom riadku vybrať cieľovú jednotku a program Vám zadanú hodnotu automaticky prekonvertuje.



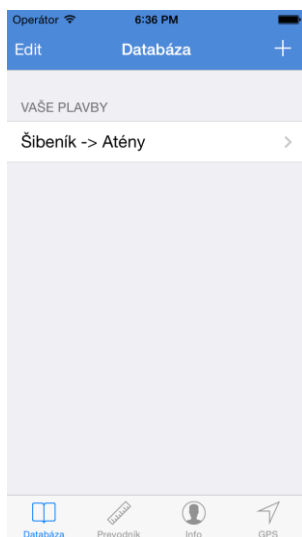
Obrázok 29: Obrazovka Prevodník



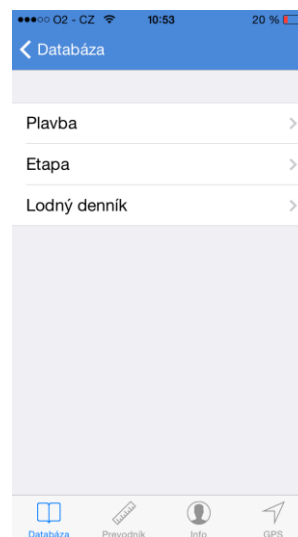
Obrázok 30: Prevodník - zadávanie

7.4.3 Databáza

Databáza zobrazuje všetky plavby, ktoré boli vykonané. Ak potrebujete založiť novú plavbu, stačí klepnúť na tlačidlo „+“ umiestnené v pravom hornom rohu. Po klepnutí na tlačidlo sa Vás program spýta na názov plavby. Ak ho zadáte a potvrdíte program, pridá novú plavbu do databázy. Naopak, ak potrebujete plavbu vymazať, stačí buď kliknúť na tlačidlo Edit, ktorým vyvoláte špeciálny režim aplikácie, v ktorom je možné mazať záznamy. Alebo jednoducho danú plavbu prstom potiahnuť sprava doľava a rovnako je možné daný riadok databázy vymazať. Ak na Vašu plavbu kliknete, presuniete sa do ďalšieho menu, kde máte na výber z troch možností, a to Plavba, Etapa, Lodný denník.



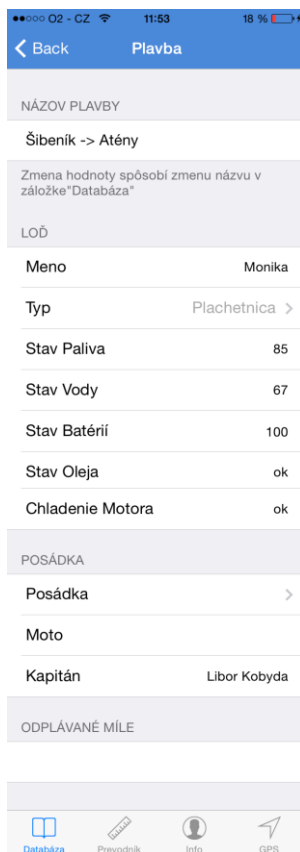
Obrázok 31: Obrazovka Databáza



Obrázok 32: Databáza - výber

7.4.3.1 Plavba

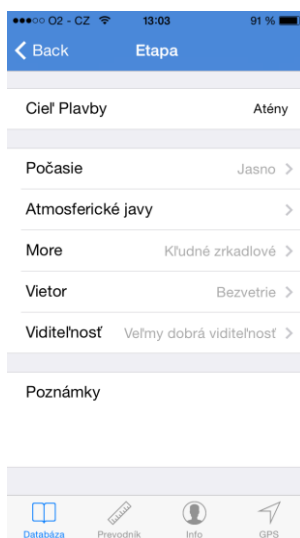
Ak vyberiete plavbu, dostanete sa na obrazovku, v ktorej je možné zmeniť názov plavby, ale hlavne sú tu údaje o lodi, ako jej meno, typ, stav paliva, stav vody, stav batérií, stav oleja a stav chladienia. Ďalej tu nájdete informácie o posádke, a to menný zoznam členov posádky, motto kapitána a meno kapitána. A ako posledné sú tu odplávané míle z plavby, ktoré nie sú ale užívateľsky editovateľné.



Obrázok 33: Databáza - plavba

7.4.3.2 Etapa

Ak z menu vyberiete etapu, dostanete sa do ďalšej obrazovky, kde je možné vyplniť cieľ plavby, poznámku a vyplniť údaje týkajúce sa počasia. Výber počasia je riešené formou výberového menu.



Obrázok 34: Databáza - etapa

7.4.3.3 *Lodný denník*

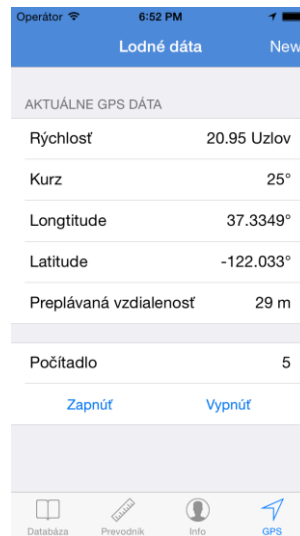
Po vytvorení záznamu sa v tejto časti pridá záznam, v ktorom sú obsiahnuté všetky údaje o plavbe v aktuálnom čase. Každý záznam si môžete otvoriť a pozrieť si jeho parametre. Po dokončení plavby stačí kliknúť na tlačidlo Export a program vygeneruje pdf súbor, ktorý automaticky nakopíruje ako prílohu do aplikácie Mail, kde už stačí len napísať mail, kam chcete lodný denník odoslať. Jednotlivé záznamy sa samozrejme dajú vymazávať podobne ako bolo popísané na začiatku v databáze.



Obrázok 35: Databáza - lodný denník

7.4.4 GPS

Hneď po kliknutí na GPS sa Vám automaticky zapne GPS modul, ktorý Vás lokalizuje a získané údaje, ako rýchlosť, aktuálny kurz, zemepisnú šírku a dĺžku a preplávanú vzdialenosť zobrazuje v tabuľke. Pri prvom spustení sa Vás systém spýta, či chcete povoliť GPS. GPS modul môžete ovládať aj ručne, a to pomocou tlačidla Zapnúť a Vypnúť. Najdôležitejšie v tejto časti ale je tlačidlo New v pravom hornom rohu, ktorým vytvoríte nový záznam do lodného denníka naposledy otvorenej plavby.



Obrázok 36: Obrazovka GPS

7.5 Základné časti programu

V nasledujúce kapitoly obsahujú programátorský popis aplikácie.

7.5.1 Info

Ako prvá bola naprogramovaná časť aplikácie info, ktorá bola aj najjednoduchšia. Info je v podstate len statická tabuľka vytvorená v Storyboarde. Keďže tabuľka je statická, nepotrebuje žiadneho delegáta v kóde, takže sa ušetrilo pár riadkov kódu. Info obsahuje jednu akciu a 2 outletry.

```

19 @property (weak, nonatomic) IBOutlet UILabel *Meno;
20 @property (weak, nonatomic) IBOutlet UILabel *OdplavaneHodiny;
21
22 -(IBAction)EditName:(id)sender;

```

Obrázok 37: Použité outletry a akcie pre Info

Outlet meno je typu UILabel a je prepojený s prvým riadkom tabuľky obrazovky Info. Outlet odplávané hodiny je tiež typu UILabel a je prepojený s posledným riadkom tabuľky. Následne je ešte v hlavičkovom súbore definovaná táto premenná

```

17 @property (nonatomic ,retain) NSMutableDictionary *data;

```

Obrázok 38: Použitá globálna premenná v kóde pre Info

Prvá premenná data je typu NSMutableDictionary, čo znamená, že je to programátorsky editovateľné pole, ktoré na každom prvku obsahuje kľúč a k nemu priradenú hodnotu. Táto premenná sa využíva na načítanie obsahu zo súboru DataProfil.plist.

Obrazovka info funguje tak, že vždy keď ju chceme zobrazit', tak ako prvé sa spustí funkcia `viewWillAppear`, v ktorej je kód, ktorý zabezpečuje vyplnenie tabuľky údajmi, ako je Meno kapitána a jeho odplávané hodiny. Ako prvé sa v tejto funkcii vyhľadá cesta k zložke Documents, kde sa ukladajú dáta aplikácie. Každá aplikácia má vlastnú zložku Documents a každá aplikácia dokáže pristupovať len k svojej zložke. Po tom, ako bola získaná cesta k zložke Documents, bola táto cesta doplnená o tento reťazec "DataProfil.plist". Týmto bola získaná kompletná cesta k súboru DataProfil.plist, ktorý bolo potrebné načítať do premennej `data`. Ešte pred načítaním dát zo súboru DataProfil.plist do premennej bolo potrebné overiť, či takýto súbor vôbec existuje. Na toto sa používa trieda `NSFileManager`, ktorá má metódu `fileExistsAtPath: filePath`. Táto metóda bola použitá v podmienke, vďaka ktorej, ak súbor DataProfil.plist existuje, tak sa zapíšu jeho hodnoty do tabuľky na príslušné miesto a ak neexistuje, tak systém vytvorí varovné okno s hlásením, že je nutné si vyplniť profil.

```

24 -(void)viewWillAppear:(BOOL)animated{
25
26     delegat = (STPAppDelegate*)[[UIApplication sharedApplication]delegate];
27     NSArray *cesta = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
28     NSString *cestaDokumenty = [cesta objectAtIndex:0];
29     NSString *NazovSuboru = [cestaDokumenty stringByAppendingPathComponent:@"DataProfil.plist"];
30     NSFileManager *fileManager = [NSFileManager defaultManager];
31
32     if ([fileManager fileExistsAtPath: NazovSuboru])
33     {
34         self.data = [[NSMutableDictionary alloc] initWithContentsOfFile: NazovSuboru];
35         NSString *dataPlist;
36         dataPlist = [self.data objectForKey:@"meno"];
37         Meno.text = dataPlist;
38         dataPlist = [self.data objectForKey:@"odplavane_hodiny"];
39         OdplavaneHodiny.text = dataPlist;
40     }
41     else
42     {
43         UIAlertView * allertVlozMeno = [[UIAlertView alloc] initWithTitle:@"Nemáte vyplnený profil
44             prosím zadajte Vaše meno" message:nil delegate:self cancelButtonTitle:@"Zrušiť"
45             otherButtonTitles:@"OK", nil];
46
47         allertVlozMeno.alertViewStyle = UIAlertViewStylePlainTextInput; // pridané textové pole
48         allertVlozMeno.tag = 0; // označenie ak používame viacero alertov v jednom view
49         [allertVlozMeno show];
50     }
51 }

```

Obrázok 39: Funkcia `viewWillAppear` - Info

Ďalej bolo potrebné, aby aplikácia vedela urobiť fotku kapitána a kapitánskej licencie. Keďže bolo nutné urobiť dve rôzne fotky, mali sme možnosť buď spraviť 2 obrazovky pre každú fotku zvlášť, alebo použiť pre obe fotky tú istú obrazovku, čím by bolo ušetrených aj zopár riadkov kódu. Takže bola vytvorená jedna obrazovka a jeden kód. Následne boli vytvorené dve prepojenia (segue) jedno z bunky „Fotka“ a druhé z bunky „Licencia“ a obe tieto prepojenia boli pomenované. Po kliknutí na objekt, ktorý ma takéto prepojenie, sa volá funkcia `prepareForSegue`, ktorá dokáže rozlíšiť o ktoré prepojenie ide.

```
86 - (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
87 {
88
89     if ([[segue identifier] isEqualToString:@"profileImage"]) {
90         [[delegat prevodnikVyber]setNuberTappedRowAtIndex:@"1":8];
91     }
92
93     if ([[segue identifier] isEqualToString:@"licenceImage"]) {
94         [[delegat prevodnikVyber]setNuberTappedRowAtIndex:@"2":8];
95     }
96
97 }
98 }
```

Obrázok 40: Funkcia prepareForSegue - Info

Pomocou riadku 91 sa uloží reťazec „1“ do tabuľky na index 8. Táto tabuľka má výhodu v tom, že k nej dokážeme pristupovať z akejkoľvek časti kódu. Takže po kliknutí na bunku „Fotka“ sa nám uloží do tabuľky jednotka a presunieme sa na druhú obrazovku. V nej sa najskôr overuje, či zariadenie má fotoaparát. Ak ho nemá, vygeneruje sa varovanie. Ak je fotoaparát dostupný, tak pokračuje ďalej. Následne podľa toho, kde sme v predošlom okne klikli, vyplní premennú nameImage reťazcom s názvom súboru. Ďalej sa pokračuje kliknutím na jedno z tlačidiel, a to buď „Urob fotku“ alebo Vyber Fotku. Pokiaľ klikneme na „Urob fotku“, program otvorí aplikáciu fotoaparát a urobí fotku, ktorú následne uloží do zložky Documents. Ak zvolíme tlačidlo s názvom Vyber fotku, program nám umožní vybrať si z galérie už urobených fotiek.

7.5.2 Prevodník

Ak klikneme na prevodník, dostaneme sa do menu, v ktorom sa dá vybrať, čo chcete konvertovať. Pokiaľ si vyberieme vzdialenosť, dostaneme sa do ďalšej obrazovky, kde je znova len tabuľka s dvoma riadkami, kde v prvom riadku je na ľavej strane UITextField a na pravej strane je UILabel. Na UITextField je napojený outlet zadanaHodnota a ešte akcia textFieldDoneEditing. Na UILabel je napojený outlet zadanaHodnotaText. Toto isté sa nachádza aj v druhom riadku, ale s inými názvami premenných. Navyše ešte bolo pridané jedno veľké neviditeľné tlačidlo, na ktoré je napojená funkcia textFieldDoneEditingBigButton. Pokiaľ užívateľ klikne na UILabel, tak sa presunie na ďalšiu obrazovku, ktorá už ale nie je obyčajná statická tabuľka, ale je to tabuľka dynamická, a teda generovaná programovo. V tejto poslednej tabuľke sa zobrazujú jednotky, v akých môžeme mať zadanú alebo vypočítanú hodnotu.

```

19 -(void)viewWillAppear:(BOOL)animated{
20     delegat = (STPAppDelegate*)[[UIApplication sharedApplication]delegate];
21     zadanaHodnotaText.text = [[delegat prevodnikVyber]getTextTappedRowAtIndex:0];
22     vypocitanaHodnotaText.text = [[delegat prevodnikVyber]getTextTappedRowAtIndex:1];
23     zadanaHodnota.text = [[delegat prevodnikVyber]getValueZadanaHodnota:0];
24     [self Vypocet];
25     vypocitanaHodnota.text = [[delegat prevodnikVyber]getValueVypocitanaHodnota:0];
26     [super viewWillAppear:animated];
27 }

```

Obrázok 41: Funkcia viewWillAppear - Prevodník

Táto funkcia sa spustí vždy, keď pristupujeme na obrazovku a zo zdieľanej tabuľky tak ako bolo napísané v kapitole vyššie, preberie parametre predošlého prevodu a aplikuje ich. Takže budeme mať nastavenú jednotku zadávaného čísla aj vypočítanej hodnoty. A vyplní sa aj zadanaHodnota, a to poslednou zadanou hodnotou. Následne sa spustí funkcia Vypocet, ktorá vypočíta novú hodnotu a výsledok uloží do zdieľanej tabuľky. Ako posledné sa vykoná načítanie hodnoty zo zdieľanej tabuľky, ktorá sa uloží a vypíše priamo na display do druhého riadku na ľavú stranu do objektu UITextField.

```

35 -(IBAction)textFieldDoneEditing:(id)sender{
36     [sender resignFirstResponder];
37     [self Vypocet];
38 }
39
40 -(IBAction)textFieldDoneEditingBigButton:(id)sender{
41     [self.view endEditing:YES];
42     [[delegat prevodnikVyber]setValueZadanaHodnota:zadanaHodnota.text:0];
43     [self Vypocet];
44 }

```

Obrázok 42: Akcie pre skrytie klávesnice - Prevodník

Funkcia textFieldDoneEditing slúži na skrytie klávesnice po ukončení zadávania hodnoty. Problémom je ale to, že klávesnica použitá v tomto prípade nemá tlačidlo, na ktoré by mohla funkcia reagovať, a teda klávesnicu nebolo možné zavrieť. Preto bola napísaná druhá funkcia, ktorá bola napojená na neviditeľné tlačidlo, a teda ak užívateľ intuitívne klepne mimo klávesnicu, tak sa zavrie. Táto funkcia aj rovno ukladá zadanú hodnotu do zdieľanej tabuľky a zavolá funkciu pre výpočet výsledku.

```

46 -(void)Vypocet{
47     [[delegat prevodnikVyber]setValueZadanaHodnota:zadanaHodnota.text:0];
48
49     NSString *hodotavyberu1 = [[delegat prevodnikVyber]getNuberTappedRowAtIndex:0];
50     NSString *hodotavyberu2 = [[delegat prevodnikVyber]getNuberTappedRowAtIndex:1];
51     int vyber_1 = [hodotavyberu1 integerValue];
52     int vyber_2 = [hodotavyberu2 integerValue];
53
54     NSString *hodnotaTextu1 = [[delegat prevodnikVyber]getTextTappedRowAtIndex:0];
55     zadanaHodnotaText.text = hodnotaTextu1;
56     NSString *hodnotaTextu2 = [[delegat prevodnikVyber]getTextTappedRowAtIndex:1];
57     vypocitanaHodnotaText.text = hodnotaTextu2;
58
59     NSString *zadana = zadanaHodnota.text; // zadaná hodnota
60     double cisloNaPrevod = [zadana doubleValue];
61
62     vysledok = 0;

```

Obrázok 43: Funkcia Vypocet - Prevodník

Funkcia Vypocet uloží do zdieľanej tabuľky zadanú hodnotu. Následne si vezme zo zdieľanej tabuľky parameter, ktorý definuje zdrojovú a cieľovú jednotku. Tieto reťazce skonvertuje na klasický integer. Následne zapíše zdrojovú a cieľovú jednotku na display a načíta zadané číslo. Zadané číslo je ale vo formáte string, a preto ho musí ešte skonvertovať na double.

```
63     switch (vyber_1)
64     {
65         case 0:
66
67             switch (vyber_2)
68             {
69                 case 0: vysledok = cisloNaPrevod; // metre na metre
70                     vysledokString = [[NSString alloc] initWithFormat:@"%4.2f", vysledok];
71                     vypocitanaHodnota.text = vysledokString;
72                     break;
73
74                 case 1: vysledok = cisloNaPrevod / 1000; // metre na kilometre
75                     vysledokString = [[NSString alloc] initWithFormat:@"%4.2f", vysledok];
76                     vypocitanaHodnota.text = vysledokString;
77                     break;
78
79                 case 2: vysledok = cisloNaPrevod / 1609.344; // metre na míle
80                     vysledokString = [[NSString alloc] initWithFormat:@"%4.2f", vysledok];
81                     vypocitanaHodnota.text = vysledokString;
82                     break;
83
84                 case 3: vysledok = cisloNaPrevod * 3.2808; // metre na stopy
85                     vysledokString = [[NSString alloc] initWithFormat:@"%4.2f", vysledok];
86                     vypocitanaHodnota.text = vysledokString;
87                     break;
88
89                 default:
90                     break;
91             }
92         break;
```

Obrázok 44: Funkcia Vypocet - riešenie výpočtu - Prevodník

Funkcia na výpočet ďalej pokračuje výpočtom výsledku, a to pomocou switchov. Ak bola vybraná jednotka meter a druhá jednotka, napríklad míle, potom sa cez prvý switch dostaneme na case 0 a cez druhý switch na case 2. Následne sa prevedie výpočet a uloží na obrazovku.

7.5.2.1 Tvorba dynamickej tabuľky

Pokiaľ klikneme na UILabel, dostaneme sa do ďalšej tabuľky, ktorá je generovaná programovo. Ako prvé potrebujeme mať nejaké pole hodnôt, ktoré budeme vkladať do buniek tabuľky. Toto pole budeme potrebovať aj ďalej, takže si spravíme ešte jedno pole, ktoré bude prístupné globálne. Po druhé bolo potrebné nejako mať označený vybraný riadok. V tomto prípade bola zvolená vlastnosť bunky UITableViewCellAccessoryCheckmark. Ak už bola nejaká jednotka vybraná, načítame si číslo vybraného riadku.

```

21 - (void)viewDidLoad
22 {
23     delegat = (STPAppDelegate*)[[UIApplication sharedApplication]delegat];
24
25     NSArray *array = [[NSArray alloc] initWithObjects:@"Metre", @"Kilometre", @"Míle", @"Stopy", nil];
26     self.listDataTable = array;
27
28     NSString *volba = [[delegat prevodnikVyber]getNuberTappedRowAtIndex:4];
29     NSUInteger volbanum = [volba intValue];
30     switch (volbanum) {
31         case 1:
32             predoslaVolba = [[delegat prevodnikVyber]getTextTappedRowAtIndex:0];
33             break;
34         case 2:
35             predoslaVolba = [[delegat prevodnikVyber]getTextTappedRowAtIndex:1];
36             break;
37         default:
38             break;
39     }
40
41     NSUInteger indexInArray = [listDataTable indexOfObject:predoslaVolba];
42
43     if (indexInArray<4) {
44         predoslaVolba = [[NSString alloc] initWithFormat:@"%ld", (unsigned long)indexInArray];
45     }else{
46         predoslaVolba = @"999";
47     }
48     [super viewDidLoad];
49 }
50 }

```

Obrázok 45: Funkcia viewDidLoad - Prevodník

Následne na to, aby sme vedeli vygenerovať tabuľku, potrebujeme poznať počet riadkov v jednej sekcii tabuľky. V tomto prípade bola použitá tabuľka len s jednou sekciov, takže na výstupe funkcie bude počet prvkov pola listDataTable .

```

60 -(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
61 {
62     return [self.listDataTable count];
63 }

```

Obrázok 46: Funkcia numberOfRowsInSection - Prevodník

Ďalšia potrebná funkcia je funkcia, ktorá sa volá vždy, kde sa vykresľuje nový riadok v tabuľke. Táto funkcia obsahuje identifikátor bunky, ktorý sa nastavuje v Storyboarde. Následne si zvolíme, aký formát má mať bunka a do novej bunky pridáme text z pola. V tejto funkcii už využijeme aj číslo predošlej voľby. Pomocou podmienky, ak sa vypisovaný riadok rovná číslu riadku, ktorý bol naposledy vybraný, tento nový riadok bude označený fajkou.

```
67 -(UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
68 {
69     static NSString *SimpleTableIdentifier = @"SimpleTableIdentifier";
70     UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:SimpleTableIdentifier];
71     if (cell == nil){cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
72         reuseIdentifier:SimpleTableIdentifier];
73     }
74     NSUInteger row = [indexPath row];
75     cell.textLabel.text = [listDataTable objectAtIndex:row];
76
77     if (indexPath.row == predoslaVolba.intValue)
78         cell.accessoryType = UITableViewCellAccessoryCheckmark;
79     else
80         cell.accessoryType = UITableViewCellAccessoryNone;
81
82     return cell;
83 }
```

Obrázok 47: Funkcia cellForRowAtIndexPath - Prevodník

Predposledná potrebná funkcia vracia indexPath bunky, na ktorú užívateľ klikol. V tejto funkcii sa dá napríklad zablokovat' interakcia s nejakou bunkou. V tomto prípade ale nič blokové nebude.

```
91 -(NSIndexPath *)tableView:(UITableView *)tableView willSelectRowAtIndexPath:(NSIndexPath *)indexPath{
92
93     return indexPath;
94 }
```

Obrázok 48: Funkcia willSelectRowAtIndexPath - Prevodník

Posledná funkcia slúži na definovanie akcie po tom, ako si užívateľ vybral bunku. Takže táto funkcia si preberie indexPath z predošlej funkcie. Následne nastavíme všetky bunky ako neodfajknuté. Potom bunku, na ktorú užívateľ klikol, ako odfajknutú. A na koniec je treba číslo bunky uložiť do zdieľanej tabuľky. Po výbere bunky sa automaticky užívateľ vráti späť, kde už bude mať nastavený príslušný parameter.

```

96 -(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
97 {
98     for (int i = 0; i<=self.listDataTable.count; i++) {
99         NSIndexPath *indexPath1 = [NSIndexPath indexPathForRow:i inSection:0];
100         UITableViewCell *cell = [self.tableView cellForRowAtIndexPath:indexPath1];
101         cell.accessoryType = UITableViewCellAccessoryNone;
102     }
103     [tableView cellForRowAtIndexPath:indexPath].accessoryType = UITableViewCellAccessoryCheckmark;
104     delegat = (STPAppDelegate*)[[UIApplication sharedApplication]delegate];
105     NSUInteger row = [indexPath row];
106     rowValue = [listDataTable objectAtIndex:row];
107     vybranabunkacislo = [[NSString alloc] initWithFormat:@"%lu", (unsigned long) row];
108
109     NSString *volba = [[delegat prevodnikVyber]getNuberTappedRowAtIndex:4];
110     NSUInteger volbanum = [volba intValue];
111     switch (volbanum) {
112         case 1:
113             [[delegat prevodnikVyber]setNuberTappedRowAtIndex:vybranabunkacislo:0];
114             [[delegat prevodnikVyber]setTextTappedRowAtIndex:rowValue:0];
115             break;
116         case 2:
117             [[delegat prevodnikVyber]setNuberTappedRowAtIndex:vybranabunkacislo:1];
118             [[delegat prevodnikVyber]setTextTappedRowAtIndex:rowValue:1];
119             break;
120         default:
121             break;
122     }
123
124     [self.navigationController pushViewControllerAnimated:YES];
125 }
126 }

```

Obrázok 49: Funkcia didSelectRowAtIndexPath - Prevodník

7.5.3 GPS

Táto časť programu využíva framework Core Location, ktorý bol už popísaný v podkapitole Základné funkcie Cole Location. Obrazovka tejto časti sa skladá len zo statickej tabuľky, takže nie je potrebné vkladať funkcie pre generovanie dynamických tabuľiek. Po tom, ako užívateľ klikne na GPS, sa vytvorí inštancia objektu CLLocationManager. Následne nastavíme parametre pre túto inštanciu, a zároveň overíme, či má zariadenie GPS modul. Po tom, ako máme overené, že GPS modul je dostupný, môžeme do tabuľky zapísať počiatočné hodnoty a spustiť GPS. Okrem tabuľky, obrazovka obsahuje aj 3 tlačidlá, z ktorých to, ktoré je umiestnené v pravej hornej časti, je najdôležitejšie, pretože toto tlačidlo slúži na vygenerovanie nového záznamu do lodného denníka naposledy otvorenej plavby. Na toto tlačidlo je napojená akcia NovyZaznam, ktorá funguje nasledovne.

Na začiatku funkcie sa získa cesta k súboru Databaza.plist kde sa ukladajú všetky dáta. Obsah celého tohto súboru uložíme do premennej dataplist. Následne si funkcia overuje, či v súbore Databaza.plist existuje, alebo ak existuje, či má nejaký obsah. Ak neexistuje alebo nemá žiadny obsah, program vygeneruje varovanie a program automaticky otvorí Databázu. Ďalej nasleduje ďalšia podmienka, ktorá testuje, či si užívateľ zvolil nejakú plavbu, ak nie, potom sa znova vygeneruje varovanie a otvorí sa Databáza. Pokiaľ ale máme vybranú aj plavbu, program pokračuje ďalej, kde si načíta do premenných všetky vlastnosti počasia,

ktoré sa zadávajú do lodného denníka. Následne si načíta už zadané hodnoty, ktoré sa nachádzajú v dataplist. Ďalej program získa poradové číslo vlastností, ktoré sa používa ďalej. Toto sa deje na základe toho, že do lodného denníka sa nepopisuje aktuálne počasie textom, ale len číselnou hodnotou. Ďalším krokom funkcie je získanie aktuálneho dátumu a času. Tieto údaje sú potrebné pre správne vyplnenie lodného denníka. Ďalej sa riešia odplávané hodiny. Toto funguje tak, že sa porovnáva posledná hodnota času s aktuálnou. Ak je aktuálna hodnota času väčšia ako posledná, potom sa k odplávaným hodinám pripočíta jednotka. Keď už máme novú hodnotu odplávaných hodín, uložíme túto hodnotu do súboru DataProfil.plist. Následne bola vytvorená premenná typu NSDictionary, do ktorej sa uložia všetky potrebné hodnoty pre vygenerovanie lodného denníka. Ďalej je podmienka, ktorá kontroluje, či boli zadané všetky parametre počasia v sekcii Etapa. Ak je táto podmienka splnená, a teda máme všetko zadané tak, ako je treba, tento záznam sa nám uloží do databázy. Ak nie, potom vyskočí varovanie.

Nakoniec je potrebné popísať funkcie, v ktorých sa získavajú údaje o aktuálnej polohe. Tieto funkcie sú dve. Prvá z nich je funkcia, ktorá sa stará o získanie aktuálneho kurzu. Táto funkcia je veľmi jednoduchá, pretože si stačí len uložiť hodnotu kurzu, ktorú nám už sama funkcia ponúka. Aktuálny kurz získame z objektu newHeading pomocou metódy magneticHeading. Následne túto informáciu len pretypujeme na integer a vypíšeme na display.

```
281 -(void)locationManager:(CLLocationManager *)manager didUpdateHeading:(CLHeading *)newHeading{
282     NSLog(@"Spustene didUpdateHeading");
283     NSString *courseString = [NSString alloc] initWithFormat:@"%g", newHeading.magneticHeading];
284     courseString = [NSString alloc] initWithFormat:@"%ld", (long)courseString.integerValue];
285     courseLabel.text = courseString;
286 }
```

Obrázok 50: Funkcia didUpdateHeading - GPS

Ďalšia funkcia získava údaje o aktuálnej polohe, rýchlosti a mnoho ďalších dát. Tá funguje nasledovne. Ako prvé si funkcia zistí, či nie je aplikácia na pozadí. Ak je aplikácia na pozadí, tak sa vypne zaznamenávanie kurzu čo šetrí baterku. Ak je aplikácia aktívna, tak sa spustí aj aktualizovanie kurzu. Ďalej bola vytvorená premenná, ktorá počítá počet spustení tejto funkcie. Nasleduje uloženie potrebných informácií do premenných. Tieto informácie sa nachádzajú v objekte current.

```
288 -(void) locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations{
289
290     UIApplicationState state = [[UIApplication sharedApplication] applicationState];
291
292     if (state == UIApplicationStateBackground || state == UIApplicationStateInactive) [locationManager
293         stopUpdatingHeading];
294
295     if (state == UIApplicationStateActive) [locationManager startUpdatingHeading];
296
297     pocitadloBodovGPS = [[NSString alloc] initWithFormat:@"%ld", (long)pocitadloBodovGPS.integerValue +
298         1];
299     pocitadlo.text = pocitadloBodovGPS;
300
301     CLLocation *current = [locations lastObject];
302     NSString *longitudeString = [[NSString alloc] initWithFormat:@"%g°", current.coordinate.longitude];
303     NSString *latitudeString = [[NSString alloc] initWithFormat:@"%g°", current.coordinate.latitude];
304     NSString *horizontalAccuracyString = [[NSString alloc] initWithFormat:@"%g m", current.
305         horizontalAccuracy];
306     NSString *altitudeString = [[NSString alloc] initWithFormat:@"%g°", current.altitude];
307     NSString *verticalAccuracyString = [[NSString alloc] initWithFormat:@"%g°", current.
308         verticalAccuracy];
309     longitudeLabel.text = longitudeString;
310     latitudeLabel.text = latitudeString;
311     horizontalAccuracyLabel.text = horizontalAccuracyString;
312     altitudeLabel.text = altitudeString;
313     verticalAccuracyLabel.text = verticalAccuracyString;
```

Obrázok 51: Funkcia didUpdateLocations - GPS

Pokiaľ je počítadlo spustení tejto funkcie väčšie alebo rovné 3, potom je tejto funkcii umožnené získať rýchlosť, ktorá je následne vypísaná na display, pokiaľ je hodnota väčšia ako 1. Ďalej bolo potrebné získať preplávanú vzdialenosť. Toto sa počíta len, ak je rýchlosť väčšia ako 0. Získanie vzdialenosti funguje tak, že porovnáva aktuálnu polohu s predošlou a na základe ich rozdielu dokáže určiť vzdialenosť medzi nimi. Pokiaľ budeme výsledky tejto funkcie neustále pričítavať do nejakej premennej, dostaneme celkovú preplávanú vzdialenosť. Ako posledné uložíme odplávané do súboru Databaza.plist.

```

311     if (pocitadloBodovGPS.integerValue >= 3) {
312
313         NSString *speedString = [NSString alloc] initWithFormat:@"%g Km/h", current.speed*3.6];
314         float speedtemp = speedString.floatValue;
315         speedString = [NSString alloc] initWithFormat:@"%f", speedtemp];
316
317         if (speedString.integerValue > 1) {speedLabel.text = speedString;}
318         if (speedString.integerValue <= 0) {speedLabel.text = @"0 Km/h";}
319
320         if (speedString.integerValue > 0)
321         {
322             CLLocationDistance distance = [current distanceFromLocation:predoslyBod];
323             NSString *distanceString = [NSString alloc] initWithFormat:@"%g", distance];
324             NSInteger temp1 = distanceString.integerValue;
325             NSInteger temp2 = distanceString.integerValue;
326             NSInteger vysledok = temp1+temp2;
327             distanceString = [NSString alloc] initWithFormat:@"%ld m", (long)vysledok];
328             distanceTraveledLabel.text = distanceString;
329         }
330
331         predoslyBod = current;
332     }
333
334     NSArray *cesta_databaza = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
335     NSUserDomainMask, YES);
336     NSString *cestaDokumenty_databaza = [cesta_databaza objectAtIndex:0];
337     NSString *NazovSuboru_databaza = [cestaDokumenty_databaza
338     stringByAppendingPathComponent:@"Databaza.plist"];
339     NSMutableArray *dataplist;
340     NSString* IDZaznamu = [[delegat prevodnikVyber]getNuberTappedRowAtIndex:5];
341     dataplist = [NSMutableArray alloc] initWithContentsOfFile:NazovSuboru_databaza];
342     NSFileManager *fileManager = [NSFileManager defaultManager];
343
344     if ([fileManager fileExistsAtPath: NazovSuboru_databaza])
345     {
346         int vzdialenost = distanceString.intValue / 1609.344;
347         NSString *vzdialenost_Mile = [NSString alloc] initWithFormat:@"%d", vzdialenost];
348         [dataplist[IDZaznamu.integerValue] setValue:vzdialenost_Mile
349         forKeyPath:@"Plavba.odplavane_mile"];
350         [dataplist writeToFile: NazovSuboru_databaza atomically:YES];
351     }
352 }

```

Obrázok 52: Funkcia didUpdateLocations - Riešenie rýchlosti a vzdialenosti - GPS

7.5.4 Databáza

Po tom, ako užívateľ klikne na databázu, spustí sa funkcia viewDidLoad, ktorá pracuje nasledovne. Ako prvé overí, či existuje DataProfil.plist, ktorý obsahuje dáta z profilu užívateľa. Ak tento súbor neexistuje, tak sa automaticky otvorí Info, kde bude užívateľ vyzvaný na vyplnenie profilu. Táto funkcia sa ďalej stará o predanie dvoch tlačidiel do hornej lišty. Prvé je tlačidlo „+“, ktoré slúži na pridanie novej plavby a druhé je Edit, ktoré umožňuje vymazávať plavby.

Ďalej nasleduje funkcia viewWillAppear, ktorá sa spúšťa pri každom prístupe do sekcie Databáza. A má za úlohu načítať všetky názvy plavieb zo súboru DatabazaList.plist a obnoviť tabuľku. Následne som si vytvoril funkciu, ktorá vracia kompletnú cestu k zadanému súboru, čo program viac sprehľadňuje. Funkcia funguje tak, že ako prvé si nájde cestu do zložky Document a následne k tomuto reťazcu pripojí ešte ďalší reťazec, v ktorom je názov súboru. Nakoniec funkcia vráti celý tento reťazec.

```

44 -(NSString*)getFillePath:(NSString*)fileName{
45
46     NSArray *cesta = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
47     NSString *cestaDokumenty = [cesta objectAtIndex:0];
48     NSString *NazovSuboru = [cestaDokumenty stringByAppendingPathComponent:fileName];
49     return NazovSuboru;
50
51 }

```

Obrázok 53: Funkcia getFillePath - Databáza

Nasledovná funkcia sa stará len o to, aby po stlačení tlačidla Edit nasledovala animácia.

```

59 -(void)setEditing:(BOOL)editing animated:(BOOL)animated{
60
61     [super setEditing:editing animated:animated];
62     [myTableView setEditing:editing animated:animated];
63 }

```

Obrázok 54: Funkcia setEditing - Databáza

Po stlačení tlačidla na pridanie novej plavby „+“ sa spustí funkcia insertNewObject, ktorá vygeneruje okno, do ktorého je možné vložiť nový názov plavby.

```

66 -(void)insertNewObject{
67     UIAlertView * alert = [[UIAlertView alloc] initWithTitle:@"Zadaj novú plavbu" message:nil
68         delegate:self cancelButtonTitle:@"Zrušiť" otherButtonTitles:@"OK", nil];
69
70     alert.alertViewStyle = UIAlertViewStylePlainTextInput; //pridanie textového poľa na vstup
71     alert.tag = 1;
72     [alert show];
73 }

```

Obrázok 55: Funkcia insertNewObject - Databáza

Ďalej nasledujú funkcie, ktoré sa starajú o to, aby fungovala dynamická tabuľka. Funkcie fungujú na tom istom princípe ako bolo popísané vyššie. Na rozdiel od predošlej dynamickej tabuľky je v tejto tabuľke tlačidlo, ktoré umožňuje mazanie záznamov, a preto bolo nutné implementovať funkciu, ktorá sa stará o mazanie záznamov. Táto funkcia funguje tak, že ako prvé vymaže záznam z premennej databáza. Obsahom tejto premennej následne nahradí obsah súboru DatabazaList.plist a ako druhé si načítame kompletný obsah súboru Databaza.plist, kde vymažeme záznam na tom riadku, ktorý sa zhoduje s riadkom, ktorý sme odstránili z tabuľky.

```

104 -(void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)
105     editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath{
106     if (editingStyle == UITableViewCellEditingStyleDelete){
107         [self.databaza removeObjectAtIndex:indexPath.row];
108         [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];
109         [self.databaza writeToFile: [self getFillePath:@"DatabazaList.plist"] atomically:YES];
110         NSMutableArray * celadatabaza = [[NSMutableArray alloc] initWithContentsOfFile:[self
111             getFillePath:@"Databaza.plist"]];
112         [celadatabaza removeObjectAtIndex:indexPath.row];
113         [celadatabaza writeToFile: [self getFillePath:@"Databaza.plist"] atomically:YES];
114     }
115 }

```

Obrázok 56: Funkcia commitEditingStyle - Databáza

Posledná funkcia, ktorá je implementovaná, sa stará o pridanie novej plavby do súboru Databaza.plist. Ako prvé bolo potrebné skontrolovať, či užívateľ vo vygenerovanom okne stlačil po zadaní názvu plavby OK. Ak bolo stlačené tlačidlo OK, potom funkcia pokračuje ďalej. Ako prvé si vytvoríme premennú typu NSMutableArray. Následne funkcia overuje, či existuje súbor DatabazaList.plist. Ak tento súbor existuje, potom sa do premennej uloží celý obsah súboru a na koniec tohto poľa sa pridá nová zadaná plavba. Ak súbor neexistuje, tak sa do premennej uloží len nová plavba. Následne sa obsah tejto premennej uloží do súboru DatabazaList.plist.

```

131 - (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex{
132
133     if(alertView.tag == 1)
134     {
135         // ok má index 1, zrušiť má index 0
136         if (buttonIndex == 1) {
137
138             if (!self.databaza){self.databaza = [[NSMutableArray alloc] init];}
139
140             NSString *NazovSuboru = [self getFilePath:@"DatabazaList.plist"];
141             NSFileManager *fileManager = [NSFileManager defaultManager];
142             NSString *tempTextField = [alertView textFieldAtIndex:0].text;
143             int nextToLastIndex = [self.databaza count];
144
145             if ([fileManager fileExistsAtPath: NazovSuboru])
146             {
147                 self.databaza = [[NSMutableArray alloc] initWithContentsOfFile: NazovSuboru];
148                 [self.databaza insertObject:tempTextField atIndex:databaza.count];
149                 NSIndexPath *indexPath = [NSIndexPath indexPathForRow:nextToLastIndex inSection:0];
150                 [self.myTableView insertRowsAtIndexPaths:@[indexPath] withRowAnimation:
151                     UITableViewRowAnimationAutomatic];
152             }
153
154             if (![fileManager fileExistsAtPath: NazovSuboru])
155             {
156                 [self.databaza insertObject:tempTextField atIndex:databaza.count];
157                 NSIndexPath *indexPath = [NSIndexPath indexPathForRow:nextToLastIndex inSection:0];
158                 [self.myTableView insertRowsAtIndexPaths:@[indexPath] withRowAnimation:
159                     UITableViewRowAnimationAutomatic];
160             }
161
162             [self.databaza writeToFile: NazovSuboru atomically:YES];

```

Obrázok 57: Funkcia clickedButtonAtIndex - Databáza

V druhej časti funkcie sa vytvára nový záznam do súboru Databaza.plist. Ako prvé bola vytvorená nová premenná typu NSMutableArray. Následne bolo potrebné vytvoriť štruktúru tohto databázového súboru. Toto bolo dosiahnuté tým, že sa vytvorili premenné typu NSDictionary. Prvá premenná tohto typu je plavba, kde sú už nachystané položky, do ktorých sa budú zapisovať hodnoty. Druhá premenná má názov etapa, kde máme taktiež nachystané položky, ktoré len stačí vyplniť. Tretia premenná je typu NSArray a má názov lodne_data. Táto premenná je zatiaľ prázdna. Následne sa vytvorila premenná typu NSDictionary s názvom dátova_struktura, ktorá má položky plavba, etapa a lodne_data. A priamo sem sa vložia už vytvorené príslušné premenné. Výsledok tejto operácie bude taký, že

v premennej datova_struktura budú položky plavba, etapa a lodne_data, ktoré budú ďalej obsahovať položky, ktoré sme vytvorili v príslušných premenných.

```

162     NazovSuboru = [self getFillePath:@"Databaza.plist"];
163     NSMutableArray * complete_plist = [[NSMutableArray alloc] init];
164
165     NSDictionary *plavba = @{
166
167         @"nazov": [NSString stringWithString:tempTextField],
168         @"typ_lode": [NSString stringWithFormat:@"%d"],
169         @"meno_lode": [NSString stringWithFormat:@"%s"],
170         @"palivo": [NSString stringWithFormat:@"%d"],
171         @"voda": [NSString stringWithFormat:@"%d"],
172         @"olej": [NSString stringWithFormat:@"%d"],
173         @"baterie": [NSString stringWithFormat:@"%d"],
174         @"chladenie": [NSString stringWithFormat:@"%d"],
175         @"kapitan": [NSString stringWithFormat:@"%s"],
176         @"posadka": [NSArray alloc] initWithObjects:nil,
177         @"moto": [NSString stringWithFormat:@"%d"],
178         @"odplavane_mile": [NSString stringWithFormat:@"%d"],
179     };
180
181     NSDictionary *etapa = @{
182         @"ciel_plavby": [NSString stringWithFormat:@"%d"],
183         @"pocasio": [NSString stringWithFormat:@"%d"],
184         @"atmosfer_javy": [NSString stringWithFormat:@"%d"],
185         @"more": [NSString stringWithFormat:@"%d"],
186         @"vietor": [NSString stringWithFormat:@"%d"],
187         @"viditelnost": [NSString stringWithFormat:@"%d"],
188         @"plachty": [NSString stringWithFormat:@"%d"],
189         @"motor": [NSString stringWithFormat:@"%d"],
190         @"teplota": [NSString stringWithFormat:@"%d"],
191         @"poznamka": [NSString stringWithFormat:@"%d"],
192     };
193
194     NSArray *lodne_data;
195
196     NSDictionary *datova_struktura = @{
197         @"Plavba" : [NSDictionary dictionaryWithDictionary:
198             plavba],
199         @"Etapa" : [NSDictionary dictionaryWithDictionary:etapa
200             ],
201         @"Lodne_data" : [NSArray arrayWithArray:lodne_data],
202     };

```

Obrázok 58: Funkcia clickedButtonAtIndex - Generovanie databázy - Databáza

Ak bolo toto vykonané, stačí výslednú premennú zapísať do súboru Databaza.plist. Ak tento súbor existuje, tak len pridáme do neho nový záznam, ale ak neexistuje, tak vytvoríme nový súbor, ktorého obsahom bude obsah premennej.

```

204     if (![fileManager fileExistsAtPath: NazovSuboru])
205     {
206         [complete_plist addObject:datova_struktura];
207         [complete_plist writeToFile: NazovSuboru atomically:YES];
208     }
209     else{
210         complete_plist = [[NSMutableArray alloc] initWithContentsOfFile: NazovSuboru];
211         [complete_plist addObject:datova_struktura];
212         [complete_plist writeToFile: NazovSuboru atomically:YES];
213     }
214 }
215 }
216 }
217 }
218 }
219 }
220 }

```

Obrázok 59: Funkcia clickedButtonAtIndex - Ukladanie databázy - Databáza

Po tom, ako užívateľ vytvorí nový záznam, tak sa pridá do tabuľky nový riadok, ktorý obsahuje to, čo užívateľ zadal vo vygenerovanom okne. Následne ak užívateľ klikne na tento riadok tabuľky, dostane sa na novú obrazovku, v ktorej je len statická tabuľka s riadkami: Plavba, Etapa a Lodný denník.

7.5.4.1 Plavba

Po spustení obrazovky plavba sa ako prvé spustia funkcie `viewDidLoad` a `viewWillAppear`. Funkcia `viewDidLoad` sa spustí len raz na rozdiel od funkcie `viewWillAppear`, ktorá sa spúšťa vždy ako sa prepne na obrazovku plavba. Táto obrazovka slúži na to, aby bolo možné vyplniť údaje o lodi a posádke. Obrazovka plavby je len statická tabuľka, takže je vytvorená priamo v storyboard. Táto tabuľka obsahuje dva riadky, na ktoré keď klikneme, presunieme sa na ďalšiu obrazovku, kde buď vyberieme typ lode, alebo pridáme posádku. Funkcia `viewDidLoad` v tomto prípade inicializuje objekt `DoneCancelButtonToolbar`, ktorý slúži na to, aby bolo možné zavrieť klávesnicu `NumberPad`. Táto funkcia teda len pridá nad klávesnicu malý panel, kde sú dve tlačidlá, pomocou ktorých vieme túto klávesnicu schovať. Funkcia `viewWillAppear` slúži na to, aby vyplnila údaje do tabuľky v prípade, že sme už nejaké zadali. Takže ako prvé si funkcia získa cestu k súboru `Databaza.plist`. Následne si preberie hodnotu, ktorá predstavuje číslo riadku, na ktorý sme klikli na obrazovke databáza. Toto číslo jednoznačne určuje plavbu, z ktorej sa budú načítavať hodnoty. Vypĺňanie funguje tak, že najskôr sa hodnota z databázy uloží do premennej `NSString` a následne sa pomocou outletu vypíše na `display`.

```

35 - (void)viewWillAppear:(BOOL)animated
36 {
37
38     delegat = (STPAppDelegate*)[[UIApplication sharedApplication]delegate];
39
40     NSArray *cesta = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
41     NSString *cestaDokumenty = [cesta objectAtIndex:0];
42     NSString *NazovSuboru = [cestaDokumenty stringByAppendingPathComponent:@"Databaza.plist"];
43     dataplist = [[NSMutableArray alloc] initWithContentsOfFile:NazovSuboru];
44     NSString* IDZaznamu = [[delegat prevodnikVyber]getNuberTappedRowAtIndex:5];
45
46
47     NSString* TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.nazov"];
48     NazovPlavby.text = TextZDatabazy;
49
50     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.typ_lode"];
51     TypLode.text = TextZDatabazy;
52
53     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.meno_lode"];
54     MenoLode.text = TextZDatabazy;
55
56     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.moto"];
57     Moto.text = TextZDatabazy;
58
59     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.odplavane_mile"];
60     PocetOdplavanychMil.text = TextZDatabazy;
61
62     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.palivo"];
63     palivo.text = TextZDatabazy;
64     //
65     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.voda"];
66     voda.text = TextZDatabazy;
67
68     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.olej"];
69     olej.text = TextZDatabazy;
70
71     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.baterie"];
72     baterie.text = TextZDatabazy;
73
74     TextZDatabazy = [dataplist[IDZaznamu.integerValue] valueForKeyPath:@"Plavba.chladienie"];
75     chladienie_motor.text = TextZDatabazy;
76
77     [super viewWillAppear:animated];
78 }

```

Obrázok 60: Funkcia viewWillAppear - Plavba

Ďalej nasleduje funkcia viewWillAppear, ktorá sa spustí vždy, keď opúšťame obrazovku Plavba. A pri tejto akcii sa spúšťa funkcia SaveData, ktorá ukladá všetky zadané hodnoty do databázy. Funkcia SaveData pracuje nasledovne. Ako prvé sa ukladá názov plavby, ktorý je totožný s názvom plavby, ktorý bol zadaný v databáze. Takže ak sa tento údaj zmení, zmení sa aj názov plavby v databáze. Tento parameter sa ukladá do súboru DatabazaList.plist. Ešte predtým, ako funkcia ukladá tento parameter, musí si nájsť cestu k tomuto súboru. Všetky ostatné parametre, vrátane názvu plavby sa ukladajú do súboru Databaza.plist do sekcie Plavba.

```

94 -(void)SaveData{
95
96     NSArray *cesta = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
97     NSString *cestaDokumenty = [cesta objectAtIndex:0];
98     NSString *NazovSuboru = [cestaDokumenty stringByAppendingPathComponent:@"Databaza.plist"];
99
100    NSArray *cesta1 = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
101    NSString *cestaDokumenty1 = [cesta1 objectAtIndex:0];
102    NSString *NazovSuboru1 = [cestaDokumenty1 stringByAppendingPathComponent:@"DatabazaList.plist"];
103    databazaList = [[NSMutableArray alloc] initWithContentsOfFile:NazovSuboru1];
104    delegat = (STPAppDelegate*)[[UIApplication sharedApplication]delegat];
105
106    NSString* IDZaznamu = [[delegat prevodnikVyber]getNuberTappedRowAtIndex:5];
107
108    [databazaList replaceObjectAtIndex:IDZaznamu.integerValue withObject:NazovPlavby.text];
109    [databazaList writeToFile: NazovSuboru1 atomically:YES];
110
111    [dataplist[IDZaznamu.integerValue] setValue:NazovPlavby.text forKeyPath:@"Plavba.nazov"];
112    [dataplist[IDZaznamu.integerValue] setValue:MenoLode.text forKeyPath:@"Plavba.meno_lode"];
113    [dataplist[IDZaznamu.integerValue] setValue:Moto.text forKeyPath:@"Plavba.moto"];
114    [dataplist[IDZaznamu.integerValue] setValue:Kapitan.text forKeyPath:@"Plavba.kapitan"];
115    [dataplist[IDZaznamu.integerValue] setValue:palivo.text forKeyPath:@"Plavba.palivo"];
116    [dataplist[IDZaznamu.integerValue] setValue:voda.text forKeyPath:@"Plavba.voda"];
117    [dataplist[IDZaznamu.integerValue] setValue:olej.text forKeyPath:@"Plavba.olej"];
118    [dataplist[IDZaznamu.integerValue] setValue:baterie.text forKeyPath:@"Plavba.baterie"];
119    [dataplist[IDZaznamu.integerValue] setValue:chladenie_motor.text forKeyPath:@"Plavba.chladenie"];
120
121    [dataplist[IDZaznamu.integerValue] writeToFile: NazovSuboru atomically:YES];
122    [dataplist writeToFile: NazovSuboru atomically:YES];
123 }

```

Obrázok 61: Funkcia SaveData - Plavba

7.5.4.2 Etapa

Obrazovka Etapa funguje takmer rovnako ako obrazovka Plavba. Hlavný rozdiel je v tom, že v obrazovka etapa sa vyplňa vždy, keď chceme vytvoriť nový záznam lodného denníka. Táto obrazovka ešte obsahuje nový prvok UIKit-u a to textView, čo je textové pole podobné tomu, čo bolo doteraz používané. Rozdiel je v tom, že textView je viacriadkový. Na to, aby sme s týmto prvkom mohli pracovať, musíme do hlavičkového súboru pridať UITextViewDelegate. Tento delegát nám umožňuje pridať funkciu, ktorá rieši schovanie klávesnice pri vyplňaní poľa typu textView. Táto funkcia zároveň zavolá funkciu, ktorá uloží všetky zadané hodnoty.

```

123 - (BOOL)textView:(UITextView *)textView shouldChangeTextInRange:(NSRange)range replacementText:
124     (NSString *)text
125 {
126     if([text isEqualToString:@"\n"])
127     {
128         [self SaveData];
129         [textView resignFirstResponder];
130         return NO;
131     }
132     return YES;
133 }

```

Obrázok 62: Funkcia shouldChangeTextInRange - Etapa

Posledný rozdiel je v tom, že táto obrazovka obsahuje viacero riadkov, ktoré sa zadávajú formou výberu zo zoznamu. Takže bol vytvorený univerzálny kód, ktorý rieši zadávanie týchto parametrov. Aby bolo možné vytvoriť takýto univerzálny kód, je nutné implemen-

tovať funkciu `prepareForSegue`, ktorá čaká na segue s konkrétnym názvom. Ako náhle klikneme na bunku, na ktorú je pripojený segue a volá sa funkcia `prepareForSegue`, ktorá uloží do zdieľanej tabuľky hodnotu, ktorá prináleží konkrétnemu segue. Na základe tejto hodnoty v zdieľanej tabuľke vieme určiť, aké hodnoty sa budú vypisovať v zozname možností.

```
135 - (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
136 {
137     [self SaveData];
138
139     if ([[segue identifier] isEqualToString:@"pocasio"]) {
140         [[delegat prevodnikVyber] setNuberTappedRowAtIndex:@"0":6];
141     }
142
143     if ([[segue identifier] isEqualToString:@"atmosjavy"]) {
144         [[delegat prevodnikVyber] setNuberTappedRowAtIndex:@"1":6];
145     }
146
147     if ([[segue identifier] isEqualToString:@"more"]) { ... }
148
149     if ([[segue identifier] isEqualToString:@"vietor"]) { ... }
150
151     if ([[segue identifier] isEqualToString:@"viditelnost"]) { ... }
152 }
153
154
155
156
157
158
159
160 }
```

Obrázok 63: Funkcia `prepareForSegue` - Etapa

Ak klikneme na Počasie, Atmosférické javy, More, Vietor alebo viditeľnosť vyvoláme ďalšiu obrazovku, ktorá má dynamickú tabuľku, ktorá vypisuje možnosti pre riadok, na ktorý sme klikli. Ako prvé sa spustí funkcia `viewDidLoad`, ktorá obsahuje všetky možnosti vypísaného textu. A zároveň získa poradové číslo už zadanej hodnoty v poli hodnôt. Toto bolo naprogramované z dôvodu, aby bol označený výber riadku z predošlého výberu. Na to, aby funkcia vedela, kde má hodnotu hľadať, využíva parametru, ktorý bol uložený v obrazovke Etapa vo funkcii `prepareForSegue`. Ďalej nasledujú funkcie pre vytvorenie dynamickej tabuľky. Posledná zásadná zmena je vykonaná vo funkcii `didSelectRowAtIndexPath`, ktorá je volaná vždy keď klikneme na nejaký riadok dynamickej tabuľky. Okrem riešenia toho, aby sa bude označovať vybraný riadok, sa v tejto funkcii aj priamo ukladá vybraná hodnota priam do súboru `Databaza.plist` a konkrétne do sekcie Etapa. Riešenie, do akého parametru etapy sa má vybraná hodnota uložiť, je riešené pomocou switchu a hodnoty zo zdieľanej tabuľky.

7.5.4.3 *Lodný denník*

Obrazovka Lodný denník je dynamická tabuľka, ktorej hodnoty sa generujú na obrazovke GPS pomocou tlačidla NEW. Zásadný rozdiel od klasickej dynamickej tabuľky je ale v tom, že na tejto obrazovke máme navyše tlačidlo export, ktoré generuje výstup v pdf

formáte. Ak teda sú nejaké hodnoty v lodnom denníku a stlačíme export, ako prvé sa volá funkcia exportPDF, ktorá ako prvé vygeneruje titulnú stranu lodného denníka a to nasledovne. Ako prvé si získa cestu k súboru LodnyDennik.pdf. Následne vytvorí jednu stranu pdf o veľkosti A4 a na zadané parametre vypíše nápis Lodný denník.

```

140 -(void)createTitlePage{
141
142     NSString* filePath = [self getFillePath:@"LodnyDennik.pdf"];
143     UIGraphicsBeginPDFContextToFile(filePath, CGRectZero, nil);
144     UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 595, 842), nil);
145     [self printText:207 :406 :190 :35 :@"Lodný denník" :30];
146 }

```

Obrázok 64: Funkcia createTitlePage - Lodný denník

Vypisovanie textu do PDF súboru prebieha pomocou funkcie printText, ktorá má niekoľko vstupných parametrov. Prvé 4 parametre určujú miesto a veľkosť štvorhranu, do ktorého sa bude text vypisovať. Ďalším parametrom je vypisovaný reťazec a posledným je veľkosť písma.

```

329 -(void)printText:(int)Ax : (int)Ay : (int)Bx : (int)By : (NSString *)text : (int)velkostPisma{
330
331     CGRect textRect = CGRectMake(Ax, Ay, Bx, By);
332     UIFont *font = [UIFont fontWithName:@"Arial" size:velkostPisma];
333     NSMutableParagraphStyle *paragraphStyle = [[NSParagraphStyle defaultParagraphStyle] mutableCopy];
334     paragraphStyle.lineBreakMode = NSLineBreakByTruncatingTail;
335     UIColor* farba = [UIColor blackColor];
336     paragraphStyle.alignment = NSTextAlignmentLeft;
337     NSDictionary *attributes = @{ NSFontAttributeName: font,
338                                   NSParagraphStyleAttributeName: paragraphStyle,
339                                   NSForegroundColorAttributeName: farba };
340     [text drawInRect:textRect withAttributes:attributes];
341 }

```

Obrázok 65: Funkcia printText - Lodný denník

Následne sa volá ďalšia funkcia, ktorá vygeneruje novú stranu pdf súboru a vykreslí tabuľku, ktorá je uložená formou obrázku.

```

148 -(void)newPDFFileWithTable{
149     UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 595, 842), nil);
150     [self drawImage];
151 }
152
153 -(void)drawImage{
154     CGRect imageRect = CGRectMake(0, 0, 595, 842);
155     UIImage *image = [UIImage imageNamed:@"LodnyDennik.png"];
156     [image drawInRect:imageRect];
157 }

```

Obrázok 66: Funkcie newPDFFileWithTable a drawImage - Lodný denník

V ďalšom kroku generovania PDF súboru sa overuje, či v súbore Databaza.plist v sekcii Lodne_data máme nejaké záznamy. Ak sú tam, potom funkcia pokračuje ďalej, kde vytvorí pole, do ktorého si pridá všetky potrebné údaje zo súboru. Následne sa už začne vyplňať prvá strana lodného denníka. Pokiaľ budú v lodnom denníku dva záznamy vytvorené v rovnakú hodinu program vypíše len prvý. Avšak poznámka vypísaná bude. Po tom, ako

sa vyplní všetkých 24 záznamov prvého dňa plavby, začne sa generovať ďalšia strana s poznámkami. Toto prebieha, až kým sa nevypíšu všetky údaje. Ako posledné sa volajú dve funkcie. Prvá `endOfPDF`, ktorá ukončí PDF súbor a uloží ho.

```
159 -(void)endOfPDF{
160     UIGraphicsEndPDFContext();
161 }
```

Obrázok 67: Funkcia `endOfPDF` - Lodný denník

A druhá funkcia, ktorá tento súbor vezme a vloží ho ako prílohu do novej emailovej správy.

```
303 -(void)sendMail{
304     if([MFMailComposeViewController canSendMail]){
305
306         MFMailComposeViewController *mail=[MFMailComposeViewController alloc]init];
307         mail.mailComposeDelegate= (id)self;
308         [mail setSubject:@"Export Lodný Denník"];
309         [[mail navigationBar] setTintColor:[UIColor whiteColor]];
310
311         NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask,
312             YES);
313         NSString *documentsDirectory = [paths objectAtIndex:0];
314         NSString *file = [documentsDirectory stringByAppendingPathComponent:@"LodnyDennik.pdf"];
315         NSData *data = [NSData dataWithContentsOfFile:file];
316
317         [mail addAttachmentData:data mimeType:@"application/pdf" fileName:@"LodnyDennik.pdf"];
318         [self presentViewController:mail animated:YES completion:^(NSLog(@"Action Completed"));]
319     }
320     else{NSLog(@"Message cannot be sent");}
321 }
```

Obrázok 68: Funkcia `sendMail` - Lodný denník

ZÁVER

Táto práca sa zaoberala tvorbou aplikácie pre operačný systém iOS. Aplikáciu s názvom Lodný denník, ktorá má implementované všetky potrebné veci pre fungovanie sa podarilo naprogramovať a otestovať. Na začiatku tejto práce boli popísané základné princípy pracovania s jazykom Objective-C a základnými frameworkami. Následne v praktickej časti je popísaná každá dôležitá časť programu. Výstupom aplikácie je súbor s príponou pdf, ktorý obsahuje všetky dôležité údaje o plavbe. Tento súbor je možné veľmi jednoducho odoslať mailom. Aplikácia je vo finálnej podobe stabilná, plne funkčná a zároveň pripravená na otestovanie v reálnych podmienkach.

ZOZNAM POUŽITEJ LITERATURY

- [1] *Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone*. Vyd. 1. Brno: ComputerPress, 2010, 550 s. ISBN 978-80-251-2654-7.
- [2] Cocoa Frameworks. APPLE. *Cocoa - OS X Technology Overview - Apple Developer* [online]. 2014 [cit. 2014-04-18]. Dostupné z: <https://developer.apple.com/technologies/mac/cocoa.html>
- [3] *CocoaTouch - iOS Technology Overview - Apple Developer* [online]. 2014 [cit. 2014-04-18]. Dostupné z: <https://developer.apple.com/technologies/ios/cocoa-touch.html>
- [4] MARK, Dave a Jeff LAMARCHE. *iPhone SDK: průvodce vývojem aplikací pro iPhone a iPod touch*. Vyd. 1. Brno: Computer Press, 2010, 480 s. ISBN 978-80-251-2820-6.
- [5] APPLE. *IOS7* [online]. 2014 [cit. 2014-05-24]. Dostupné z: <http://www.apple.com/ios/>
- [6] APPLE. *iPhone 5S* [online]. 2013 [cit. 2014-05-24]. Dostupné z: <http://www.apple.com/iphone-5s/>
- [7] APPLE. *Xcode* [online]. 2013 [cit. 2014-05-24]. Dostupné z: https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/About_Xcode/about.html
- [8] APPLE. *Storyboard* [online]. 2013 [cit. 2014-05-24]. Dostupné z: <https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/SecondTutorial.html>
- [9] APPLE. *Apple* [online]. 2014 [cit. 2014-05-24]. Dostupné z: <http://www.apple.com/>
- [10] PixelLove. *PixelLove* [online]. 2014 [cit. 2014-06-09]. Dostupné z: <http://www.pixellove.com/>

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

iOS Operačný systém pre mobilné zariadenia od firmy Apple.

OS X Operačný systém pre počítače firmy Apple.

GPS Globálny lokalizačný systém.

MAC Označenie pre počítače firmy Apple.

Int Integer.

MVC Model view controller.

WPS Wi-Fi-based positioning systém.

WiFi Súbor štandardov pre bezdrôtové lokálne siete.

SDK Software development kit.

UML Unified Modeling Language.

pdf Portable Document Format.

ZOZNAM OBRÁZKOV

Obrázok 1: Založenie nového projektu.....	12
Obrázok 2: Výber typu nového projektu	13
Obrázok 3: Trieda Zlomok [7].....	13
Obrázok 4: Koreňová trieda a podtrieda.....	14
Obrázok 5: Príklad kategórie	15
Obrázok 6: Zobrazenie MVC [3].....	16
Obrázok 7: Ukážka výsledku práce v Core Animation a s OpenGL ES [3].....	17
Obrázok 8: Pridanie frameworku do projektu	18
Obrázok 9: Inicializácia objektu CLLocationManager	19
Obrázok 10: Príklad použitia funkcie didUpdateLocations	19
Obrázok 11: Príklad použitia funkcie didFailWithError	20
Obrázok 12: Automatické generovanie outletu	21
Obrázok 13: Zamknutá obrazovka iOS [5].....	22
Obrázok 14: Základná obrazovka iOS [5]	22
Obrázok 15: Vrstvy iOS 7 [5].....	23
Obrázok 16: iPhone Control Center.....	23
Obrázok 17: Zobrazenie iPhone 5S [6].....	24
Obrázok 18: Zobrazenie Home button s TouchID [6].....	24
Obrázok 19: Prostredie Xcode.....	26
Obrázok 20: Možnosti nastavenia prostredia Xcode	27
Obrázok 21: Simulátor iOS.....	27
Obrázok 22: Diagram použitia aplikácie Lodný denník.....	28
Obrázok 23: Schematické rozdelenie prvkov Storyboard [9].....	29
Obrázok 24: Storyboard aplikácie Lodný denník [9]	29
Obrázok 25: Ikona aplikácie Lodný denník.....	30
Obrázok 26: Ikony v Tab Bar Controller [10]	30
Obrázok 27: Launch image aplikácie Lodný denník	31
Obrázok 28: Obrazovka Info	32
Obrázok 29: Obrazovka Prevodník.....	32
Obrázok 30: Prevodník - zadávanie.....	32
Obrázok 31: Obrazovka Databáza	33
Obrázok 32: Databáza - výber	33

Obrázok 33: Databáza - plavba.....	34
Obrázok 34: Databáza - etapa.....	34
Obrázok 35: Databáza - lodný denník	35
Obrázok 36: Obrazovka GPS.....	36
Obrázok 37: Použité outlety a akcie pre Info	36
Obrázok 38: Použitá globálna premenná v kóde pre Info	36
Obrázok 39: Funkcia viewWillAppear - Info	37
Obrázok 40: Funkcia prepareForSegue - Info	38
Obrázok 41: Funkcia viewWillAppear - Prevodník	39
Obrázok 42: Akcie pre skrytie klávesnice - Prevodník	39
Obrázok 43: Funkcia Vypocet - Prevodník	39
Obrázok 44: Funkcia Vypocet - riešenie výpočtu - Prevodník.....	40
Obrázok 45: Funkcia viewDidLoad - Prevodník	41
Obrázok 46: Funkcia numberOfRowsInSection - Prevodník.....	41
Obrázok 47: Funkcia cellForRowAtIndexPath - Prevodník.....	42
Obrázok 48: Funkcia willSelectRowAtIndexPath - Prevodník	42
Obrázok 49: Funkcia didSelectRowAtIndexPath - Prevodník	43
Obrázok 50: Funkcia didUpdateHeading - GPS.....	44
Obrázok 51: Funkcia didUpdateLocations - GPS.....	45
Obrázok 52: Funkcia didUpdateLocations - Riešenie rýchlosti a vzdialenosti - GPS	46
Obrázok 53: Funkcia getFilePath - Databáza	47
Obrázok 54: Funkcia setEditing - Databáza	47
Obrázok 55: Funkcia insertNewObject - Databáza	47
Obrázok 56: Funkcia commitEditingStyle - Databáza	47
Obrázok 57: Funkcia clickedButtonAtIndex - Databáza.....	48
Obrázok 58: Funkcia clickedButtonAtIndex - Generovanie databázy - Databáza.....	49
Obrázok 59: Funkcia clickedButtonAtIndex - Ukladanie databázy - Databáza.....	49
Obrázok 60: Funkcia viewWillAppear - Plavba.....	51
Obrázok 61: Funkcia SaveData - Plavba	52
Obrázok 62: Funkcia shouldChangeTextInRange - Etapa	52
Obrázok 63: Funkcia prepareForSegue - Etapa.....	53
Obrázok 64: Funkcia createTitlePage - Lodný denník.....	54
Obrázok 65: Funkcia printText - Lodný denník	54

Obrázok 66: Funkcie newPDFFileWithTable a drawImage - Lodný denník..... 54

Obrázok 67: Funkcia endOfPDF - Lodný denník..... 55

Obrázok 68: Funkcia sendMail - Lodný denník 55

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

Príloha 1: Súbor Databaza.plist

PRÍLOHA 1: SÚBOR DATABAZA.PLIST

Key	Type	Value
▼ Root	Array	(1 item)
▼ Item 0	Dictionary	(3 items)
▼ Etapa	Dictionary	(10 items)
atmosfer_javy	String	Hmla
ciel_plavby	String	
more	String	Nepatrne zvlnené
motor	String	2000
plachty	String	
pocasio	String	Oblaky pokrývajú 2/4 oblohy
poznamka	String	18:43 - spustenie motora
teplota	String	32
viditelnost	String	Mierne kouřmo
vietor	String	Slabý vietor
▼ Lodne_data	Array	(1 item)
▼ Item 0	Dictionary	(19 items)
atmosfer_javy	String	3
cas	String	18
cas_log	String	09-06 o 18:43
ciel_plavby	String	
den	String	9. pondelok
kurz	String	235
latitude	String	-122.18°
longitude	String	37.398°
mesiac	String	Jún
more	String	3
motor	String	2000
plachty	String	
pocasio	String	2
poznamka	String	18:43 - spustenie motora
rychlost	String	67.80
teplota	String	32
viditelnost	String	5
vietor	String	2
vzdialenost	String	749
▼ Plavba	Dictionary	(12 items)
baterie	String	
chladenie	String	
kapitan	String	
meno_lode	String	
moto	String	
nazov	String	Šibeník -> Atény
odplavane_mile	String	6
olej	String	
palivo	String	
▶ posadka	Array	(0 items)
typ_lode	String	
voda	String	