VÝVOJ MOBILNÍCH APLIKACÍ PRO IOS

určeno pro vzdělávání v akreditovaných studijních programech

ROSTISLAV FOJTÍK

Číslo operačního programu: CZ.1.07
Název operačního programu:
Vzdělávání pro konkurenceschopnost
Opatření: 7.2
Číslo oblasti podpory: 7.2.2

Inovace výuky informatických předmětů ve studijních programech Ostravské univerzity

Registrační číslo projektu: CZ.1.07/2.2.00/28.0245

OSTRAVA 2013

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

Recenzenti**:** ing. David Bražina, Ph.D.

Název: Vývoj mobilních aplikací pro iOS
Autor: Rostislav Fojtík
Vydání: první, 2014
Počet stran: 151

Studijní materiál pro distanční kurz: Vývoj mobilních aplikací pro iOS

Jazyková korektura nebyla provedena, za jazykovou stránku odpovídá autor.

ISBN

Obsah

Vysvětlivky k používaným symbolům

Průvodce studiem – vstup autora do textu, specifický způsob, kterým se studentem komunikuje, povzbuzuje jej, doplňuje text o další informace

Příklad – objasnění nebo konkretizování problematiky na příkladu ze života, z praxe, ze společenské reality, apod.

Pojmy k zapamatování.

Shrnutí – shrnutí předcházející látky, shrnutí kapitoly.

Literatura – použitá ve studijním materiálu, pro doplnění a rozšíření poznatků.

Kontrolní otázky a úkoly – prověřují, do jaké míry studující text a problematiku pochopil, zapamatoval si podstatné a důležité informace a zda je dokáže aplikovat při řešení problémů.

Úkoly k textu – je potřeba je splnit neprodleně, neboť pomáhají dobrému zvládnutí následující látky.

Korespondenční úkoly – při jejich plnění postupuje studující podle pokynů s notnou dávkou vlastní iniciativy. Úkoly se průběžně evidují a hodnotí v průběhu celého kurzu.

Úkoly k zamyšlení.

Část pro zájemce – přináší látku a úkoly rozšiřující úroveň základního kurzu. Pasáže a úkoly jsou dobrovolné.

Úvod

Tento učební materiál slouží pro výuku základů tvorby mobilních aplikací v operačním systému iOS. Čtenář se seznámí se základy programovacího jazyka Objective-C a naučí se ovládat vývojový nástroj Xcode. Po absolvování kurzu a přečtení textu bude čtenář schopen vytvořit jednoduché aplikace pro iPhone nebo iPad.

Studijní text je zaměřen na objasnění tvorby aplikací pro mobilní zařízení se systémem iOS. Základem je charakteristika a možnosti využití objektově orientovaného programovacího jazyka Objective-C.

Po absolvování textu budete:

- znát základy programovacího jazyka Objective-C
- znát základy objektově orientovaného programování
- umět pracovat s vývojovým nástrojem Xcode
- umět vytvářet jednoduché mobilní aplikace
- vědět, jak správně navrhovat architekturu mobilních aplikací

Získáte:

- základní znalosti a dovednosti pro tvorbu jednoduchých mobilních aplikací pracujících v operačním systému iOS
- schopnost naprogramovat aplikace, které využívají ovládací prvky, které poskytuje Framework Cocoa Touch
- schopnost využívat ve svých mobilních aplikací další frameworky, jako je například MapKit

1. CHARAKTERISTIKA A STRUKTURA IOS

V této kapitole se dozvíte:

cílem této lekce je ukázat základní charakteristiku operačního systému iOS pro mobilní zařízení.

Po absolvování lekce budete:

- znát základní strukturu iOS
- umět pracovat s počítači s operačním systémem Mac OS X

Klíčová slova této kapitoly:

Apple, iOS, Mac OS X, iPhone, iPad, sandboxing, push notifikace, lokání notifikace

Čas ke studiu: 2 hodiny

Pro vývoj nativních aplikací určených pro operační systém iOS je nejvhodnější používat nástroje firmy Apple. Proto je důležité se seznámit s ovládáním počítačů s operačním systémem Mac OS X. Tento systém má v mnoha ohledech jinou filozofii ovládání než rozšířenější systém MS Windows.

Velkou výhodou počítačů firmy Apple je skutečnost, že firma vyrábí nejen software, ale hardware. Operační systémy Mac OS X a iOS nejsou volně v prodeji. Systémy jsou instalovány jen na zařízení firmy Apple. Proto může být software přesněji vyladěn pro konkrétní hardwarové konfigurace a jsou využití správné ovladače.

Firma Apple vyrábí například následující hardwarové zařízení:

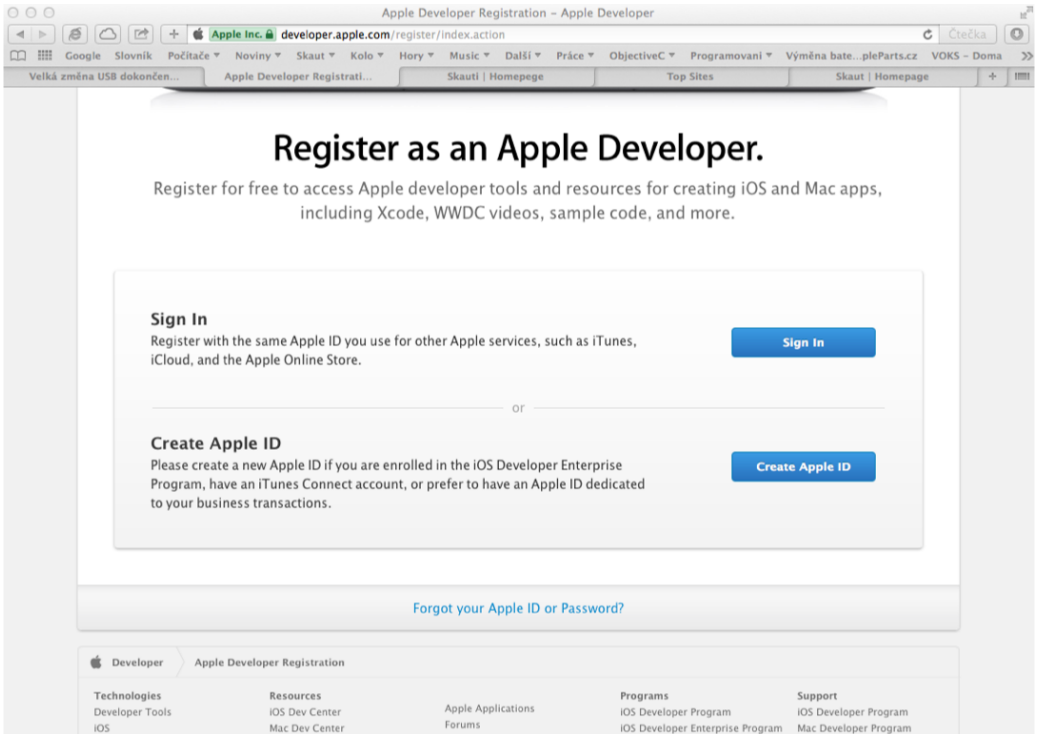
- stolní počítače iMac, Mac mini, Mac Pro
- notebooky MacBook Pro a MacBook Air
- iPad
- iPhone
- iPod
- Apple TV, Apple Airport Time Capsule, klávesnice, myši, monitory...

V oblasti software se můžeme setkat s následujícími produkty:

- operační systém Mac OS X
- operační systém iOS pro mobilní zařízení
- aplikace pro desktop i mobilní zařízení (iWork, Aperture, iPhoto, iMovie, GarageBand, Safari, iTunes, iBooks...)

Pro vývoj mobilních aplikací pro operační systém iOS potřebujeme následující:

- počítač s operačním systémem Mac OS X
- vývojové prostředí Xcode s jazykem Objective-C
- pro šíření aplikací je nutný developer účet u firmy Apple



Obrázek 1 - zřízení developerského účtu

Operační systém Mac OS X

Operační systém je úzce spjat s počítači firmy Apple a je jim plně přizpůsoben. Jeho jádro je odvozeno unixového jádra typu XNU. Při jeho vývoji byly rovněž využity vlastnosti a knihovny operačního systému Next Step. Nový uživatel Mac OS X si musí zvyknout na některé rozdílnosti oproti například MS Windows. Prostředí je samozřejmě grafické a umožňuje práci i běžným uživatelům. Na plochu si můžeme umístit dokumenty, zástupce aplikací a podobně.

Ke spuštění aplikací můžeme využít Dock, který umožňuje animované zvětšování ikon a který si můžeme schovávat nebo jej umístit do spodní nebo postranní části plochy. Které aplikace budou v Docku vidět si může uživatel nastavit podle svých požadavků.

Aplikace můžeme spouštět přímo ze složky Aplikace, do které se obvykle nahrávají. V Mac OS X nejsou registry (databáze nastavení) jako u MS Windows. U mnoha aplikací při odinstalaci často stačí smazat příslušnou aplikaci v adresáři.

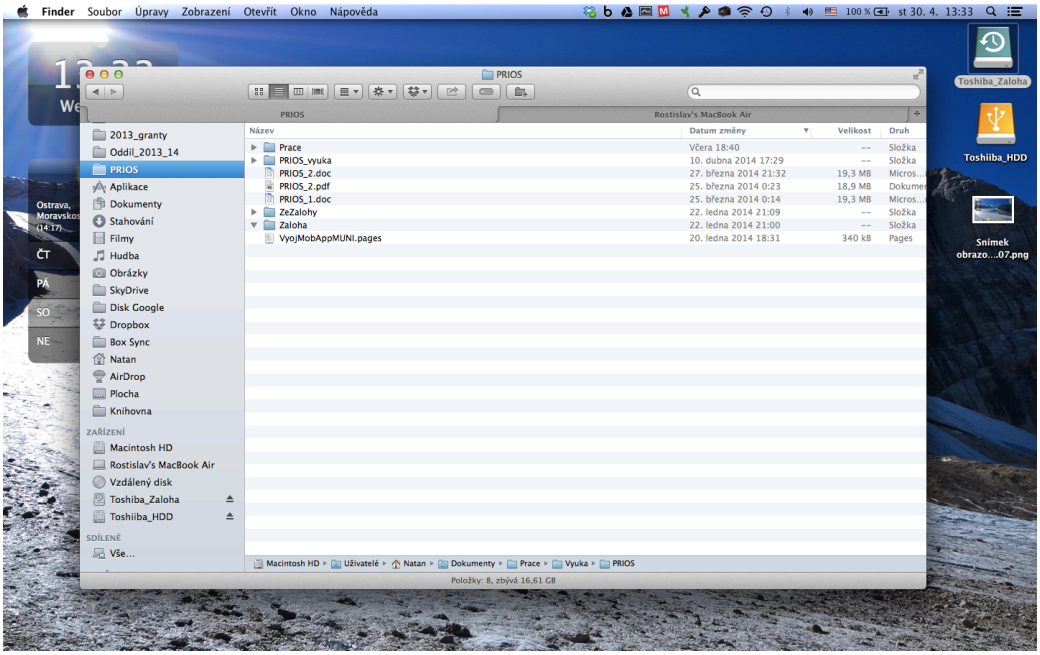
Třetí způsob jak spouštět aplikace je přes vyhledávací nástroj Spotlight. Tento nástroj prohledává nejen názvy aplikací a dokumentů, ale rovněž hledá zadané klíčové slovo uvnitř dokumentů. Díky spolehlivosti a rychlosti tohoto nástroje si uživatel často ani nemusí pamatovat, kde se dokumenty nacházejí.

Pro práci se soubory a adresáři se používá souborový manager, který se jmenuje Finder. Ten umožňuje pracovat se záložkami a do levé části si umísťovat zástupce důležitých a často využívaných složek.

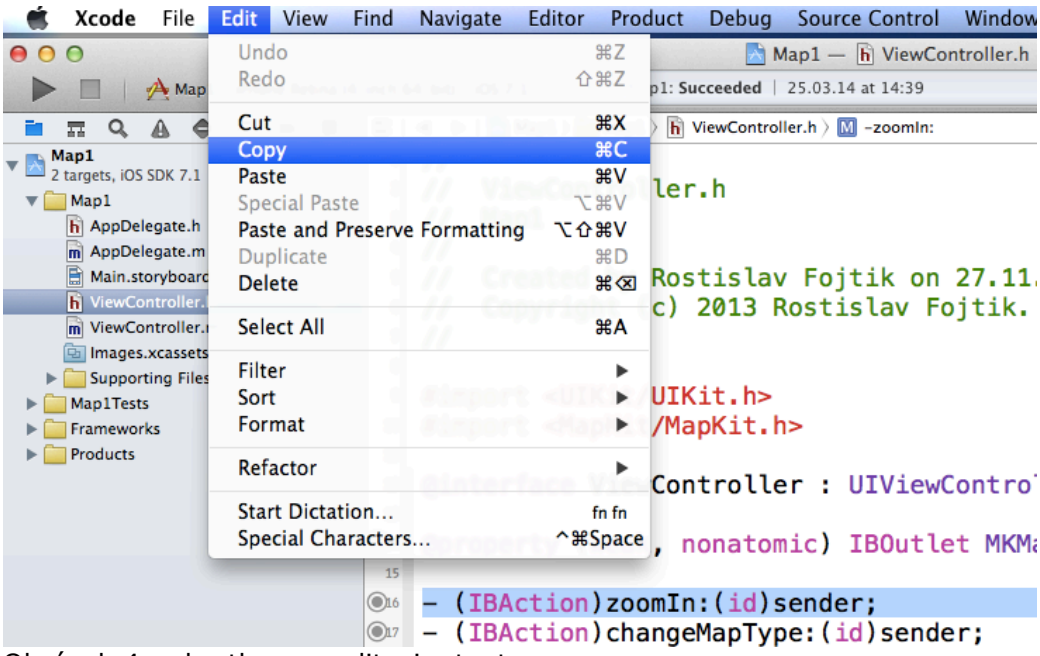
Začátečníkům dělá často obtíže uvědomit si některé rozdílnosti v ovládání:

- Pruh s menu (nabídkami aplikace) není ukotven k oknu aplikace, ale vždy v horní části obrazovky. Je proto nutné dávat pozor, se kterou aplikaci pracujeme a zda nabídka patří k dané aplikaci.
- Klávesové zkratky pro kopírování a podobně jsou jiné než u MS Windows. Kopírování je Cmd+C, vyjmutí Cmd+X, vložení Cmd+V. Klávesa Ctrl namá takové uplatnění jako u MS Windows.
- Ve Finderu se aplikace nebo dokument neotevře pouhým označením a stiskem klávesy Enter (tato volba slouží k přejmenování soboru). Otevření je nutné udělat klikem tlačítka myši nebo klávesovou zkratkou Cmd+O.
- Barevná kolečka u jednotlivých oken znamenají jiné akce než tlačítku o oken v MS Windows. Červené kolečko zavírá okno aplikace, ale aplikace je obvykle dále spuštěná. Žluté kolečko minimalizuje okno. Zelené kolečko maximalizuje okno, ale ne přes celou obrazovku, ale do velikosti, která je potřebná k zobrazení celého obsahu okna. K maximalizaci přes celou obrazovku (včetně překrytí pruhu menu) slouží tlačítko v pravém horním rohu okna. Okno pak zabírá celou samostatnou obrazovku.

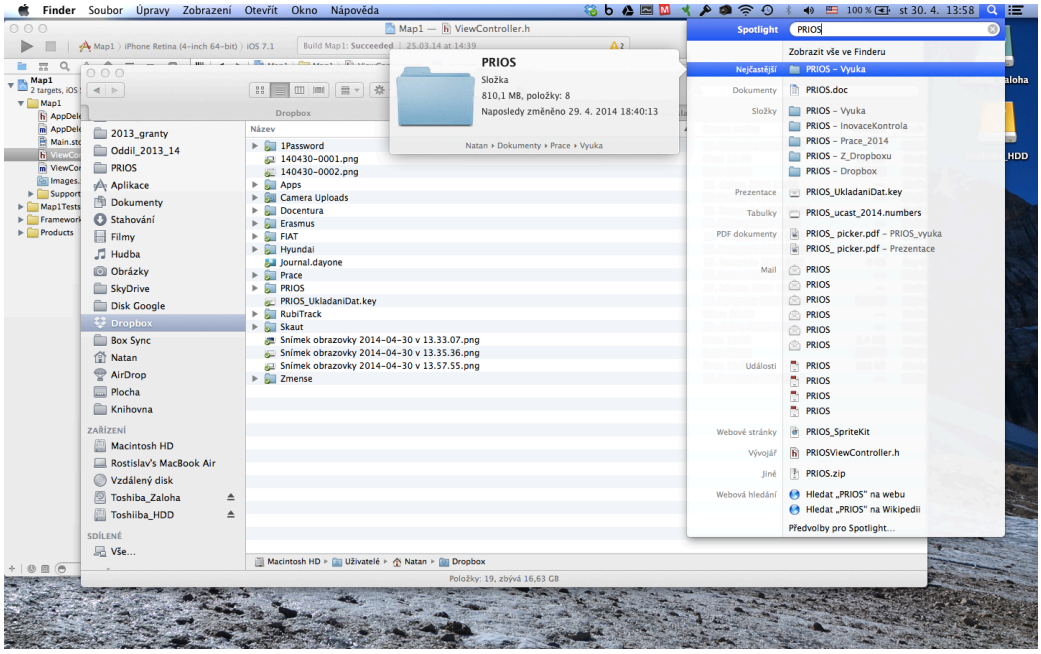
Obrázek 2 - plocha s Dockem na otevírání aplikací



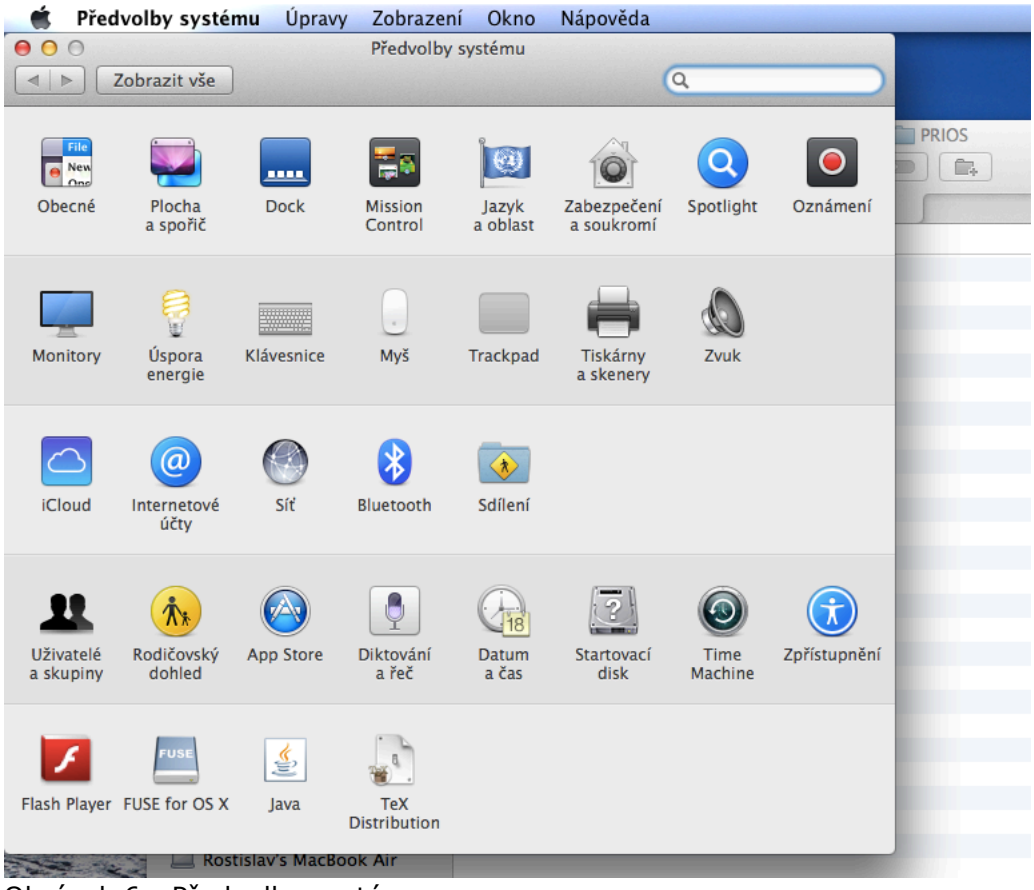
Obrázek 3 - souborový manager Finder



Obrázek 4 - zkratky pro editaci v textu



Obrázek 5 - hledání přes nástroj Spotlight



Obrázek 6 - Předvolby systému

Operační systém iOS

Operační systém iOS je určen pro mobilní zařízení typu iPhone nebo iPad. Pomocí mnoha služeb je úzce spjat i s desktopovým systémem Mac OS X, ale může rovněž existovat na něm zcela nezávisle. Tento mobilní operační systém je částečně založen na

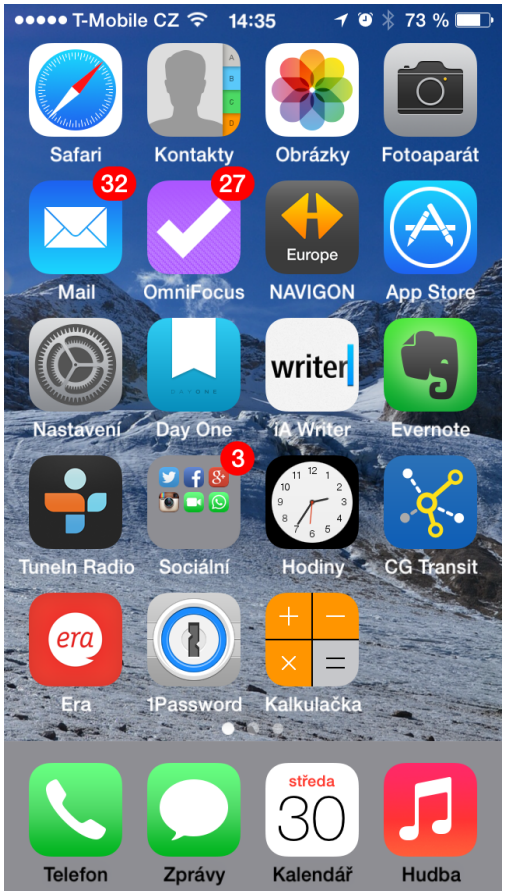
UNIXu a jedná se zjednodušený a odlehčený systém Mac OS X. Systém je rozšířen o multidotykové ovládání. Důležitou vrstvou systému je Cocoa Touch.

Důležitou vlastností systému je *Sandboxing* - práce s dokumenty. Každá aplikace má svůj vyhrazený prostor, do kterého si ukládá veškeré dokumenty. Výhodou je zvýšení bezpečnosti při práci s dokumenty, ke kterým můžeme přistupovat jen z určené aplikace. Ostatní aplikace do tohoto vyhrazeného prostoru nemají přístup. Pro běžné uživatele je rovněž výhodou jednodušší hledání souborů, které jsou jen u své aplikace. Systém iOS nenabízí plný přístup k souborovému systému, což ovšem může být i určitou nevýhodou. Další nevýhodou je, že (až na výjimky, např. Obrázky) nelze jeden dokument prohlížet v různých aplikacích. Máme-li například film, který budeme chtít pouštět v různých video přehrávačích, musíme jej nakopírovat do Sandboxu každé aplikace.

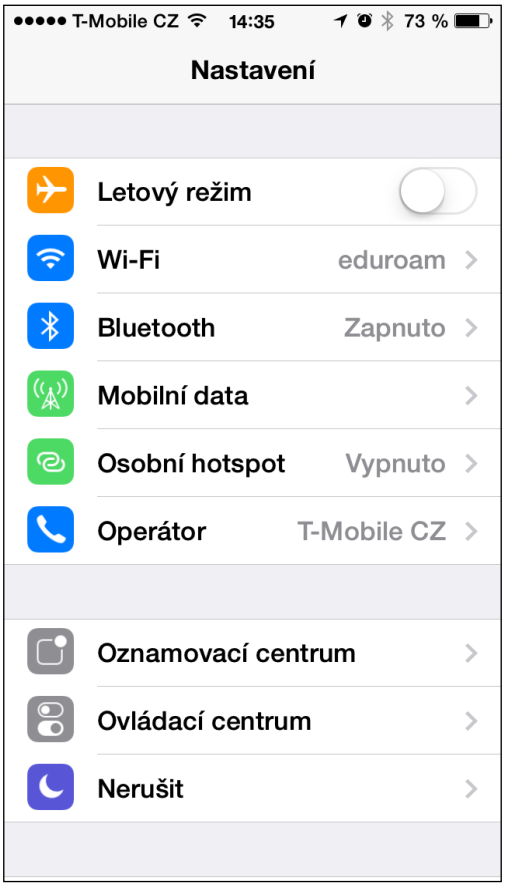
Ovládání aplikací je plně přizpůsobeno dotykovým gestům a ovládání prsty. Aplikace vždy zabírá celou obrazovku a nelze mít na obrazovce mobilního zařízení více aplikací najednou.

Systém iOS podporuje multitasking. Oproti desktopovým systémům má ovšem některá omezení, která jsou většinou z důvodů šetření energie. Většina aplikací nemůže plnohodnotně běžet na pozadí, ale může využívat tzv. *push notifikací* nebo *lokálních notifikací.* Notifikace mohou uživatele upozornit na určité informace aplikací. Push notifikace jsou iniciovány servery výrobce aplikace, které následně komunikují se servery firmy Apple. Lokální notifikace nevyžadují žádné připojení k serverům a jsou ukládány do systému. Pro vyvolání notifikací nemusí příslušné aplikace být aktuálně spuštěny.

Ke komunikaci s desktopovým systémem obvykle slouží aplikace iTunes, která zajišťuje synchronizaci dat. Synchronizace se může provádět na lokální disk nebo lze využít iCloud.

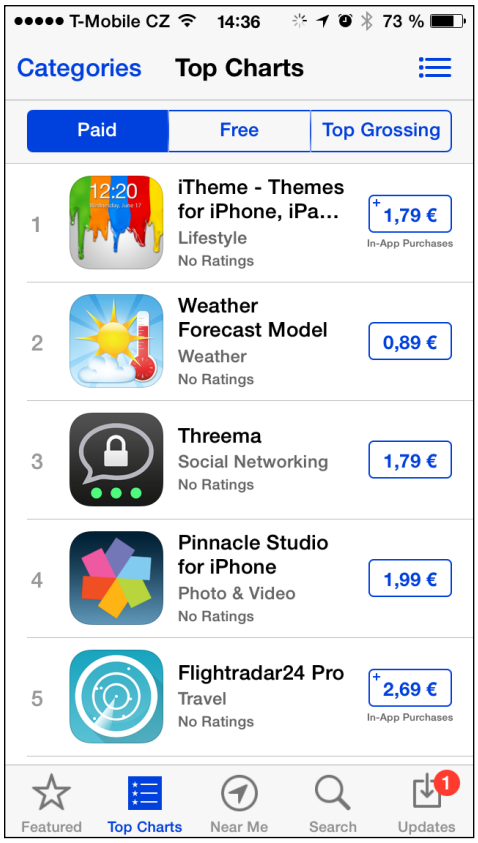


Obrázek 7 - základní obrazovka iOS na iPhone

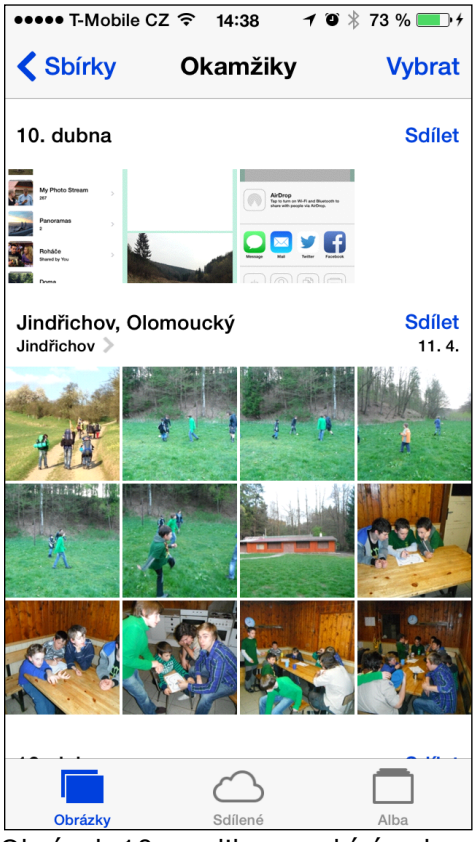


Obrázek 8 - nastavení většiny aplikací a systému se provádí ve společném

centru nastavení



Obrázek 9 - instalace aplikací se provádí prostřednictvím App Store



Obrázek 10 - aplikace zabírá celou plochu obrazovky

Kontrolní otázky

Co je to sandboxing a jaké má výhody a nevýhody?
Jaká mobilní zařízení využívají operační systém iOS?
Jak se jmenuje uložiště firmy Apple, které může sloužit k ukládání a synchronizaci dat?
K čemu slouží Spotlight v operačním systému Mac OS X?

Shrnutí kapitoly

Operační systém iOS je využíván firmou Apple pro mobilní zařízení iPhone, iPad nebo iPod Touch. Na desktopových počítačích firma používá operační systém Mac OS X.

2. ZÁKLADY PROGRAMOVACÍHO JAZYKA OBJECTIVE-C

V této kapitole se dozvíte:

cílem této lekce je ukázat základní charakteristiku objektově orientovaného programování a jeho implementace v programovacím jazyku Objective-C.

Po absolvování lekce budete:

- vědět, co je to třída a objekt
- umět vytvořit třídu a objekt v programovacím jazyku Objective-C
- umět využívat architekturu Model-View-Controller

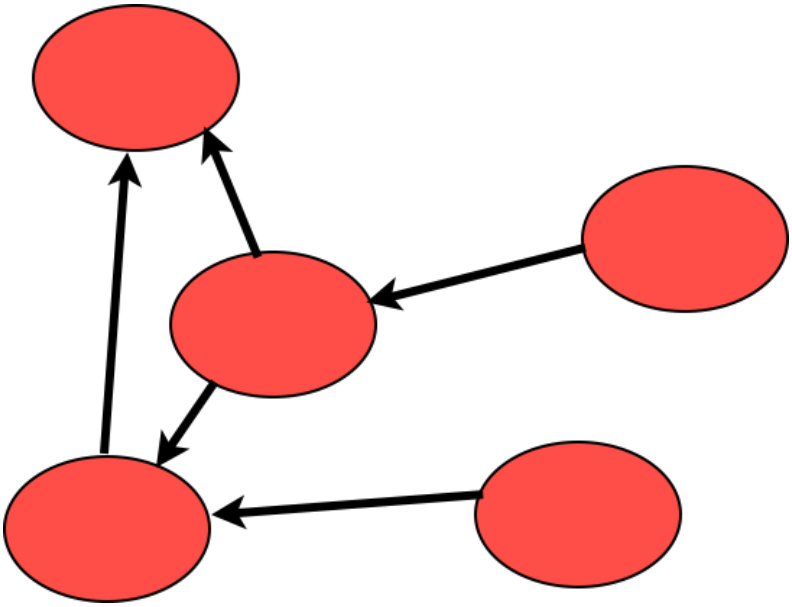
Klíčová slova této kapitoly:

programovací jazyk, vývojový nástroj, objektově orientované programování, třída, objekt, Objective-C, Model-View-Controller

Čas ke studiu: 3 hodiny

Objektově orientované programování využívá pojem třída a objekt. Třídu můžeme chápat jako obecný koncept a objekt jako konkrétní instanci třídy. Například obecný koncept *Osoba* budeme definovat jako třídu, která obsahuje metody (zprávy) a vlastnosti obecné osoby. Konkrétní osobu chápeme jako objekt třídy. Základem objektově orientovaného programování je *třída* (**class**). Typ třída se podobá struktuře v jazyce C. Může však navíc obsahovat i funkce nazývané metodami (zprávami) – princip zapouzdření. Zapouzdření kromě spojení členských dat a členských funkcí (metod) umožňuje jasně odlišit vlastnosti dané třídy, které mohou být používány i mimo definici třídy od rysů, které lze využívat jen uvnitř třídy - rozlišení přístupových práv.

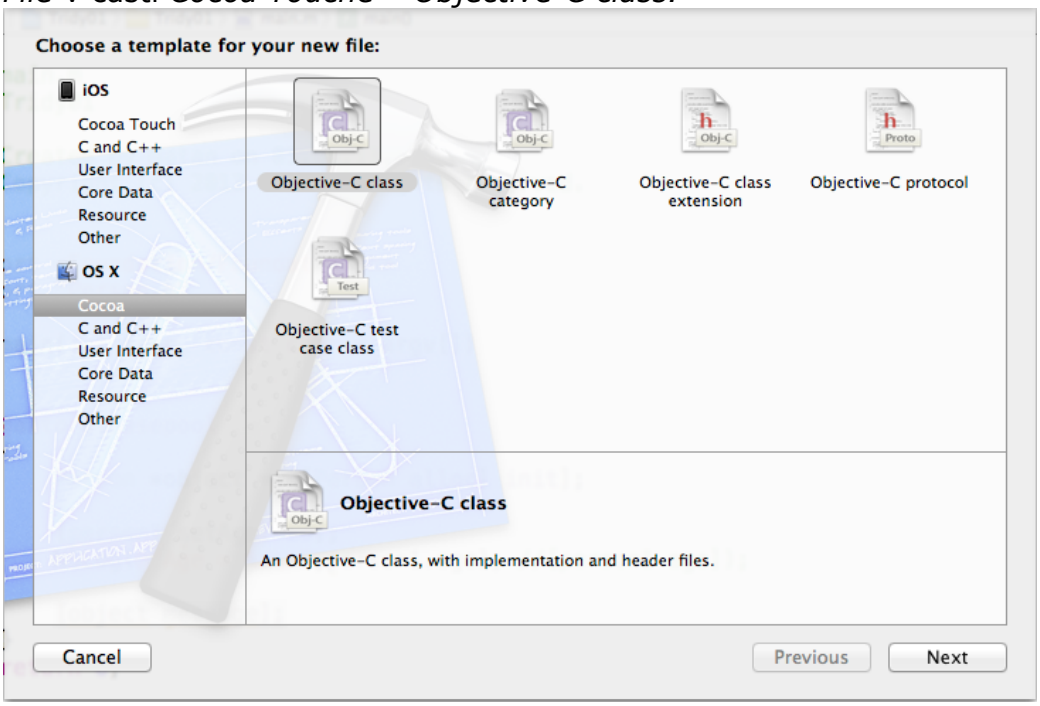
Jednotlivé objekty si zasílají zprávy (volají se metody) a mezi sebou komunikují.



Obrázek 11 - objekty si posílají zprávy

Programovací jazyk Objective-C vznikl rozšíření jazyka C objektově orientované vlastnosti. Tvůrci jazyka Brad Cox a Tom Love se nechali dost ovlivnit jazykem Smalltalk, což je vidět rovněž na způsobu zápisu kódu.

Novou třídu prostředí Xcodou vytvoříme v menu *File – New – File* v části *Cocoa Touche – Objective-C class.*

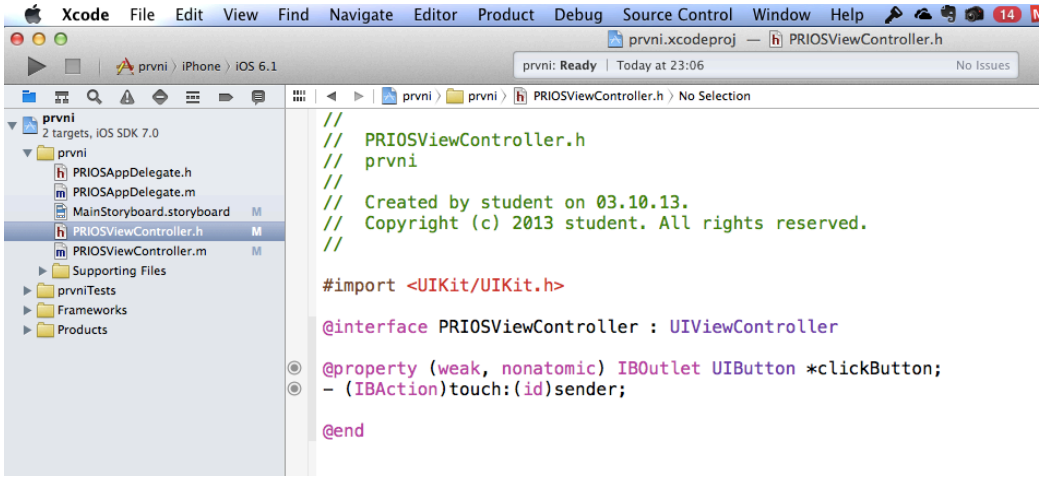


Obrázek 12 - vytvoření nové třídy

Třída v jazyku Objective-C je rozdělená do dvou souborů. První soubor má příponu *h* a obsahuje interface třídy:

```
//
//  PRIOSViewController.h
//  prvni
//
```

```
//  Created by student on 03.10.13.
//  Copyright (c) 2013 student. All rights reserved.
//
#import <UIKit/UIKit.h>

@interface PRIOSViewController : UIViewController

@property (weak, nonatomic) IBOutlet UIButton
*clickButton;
- (IBAction)touch:(id)sender;

@end
```

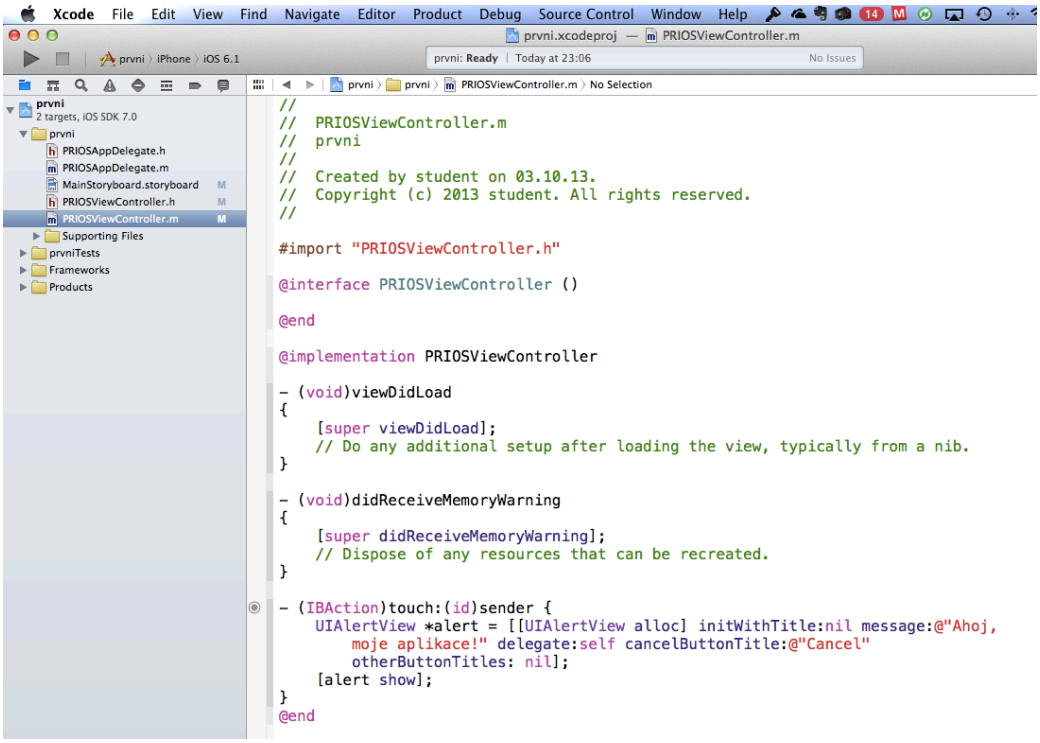


Obrázek 13 - soubor obsahující interface třídy

Implementace třídy a všech jejich metod (zpráv) se nachází v souboru s příponou *m*. Oba soubory mají stejný název, který se shoduje s názvem třídy.

```
//
//  PRIOSViewController.m
//  prvni
//
//  Created by student on 03.10.13.
//  Copyright (c) 2013 student. All rights reserved.
//
#import "PRIOSViewController.h"

@interface PRIOSViewController ()

@end

@implementation PRIOSViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
// Do any additional setup after loading the view,
// typically from a nib.
}

- (void)didReceiveMemoryWarning
{
```

```
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)touch:(id)sender {
    UIAlertView *alert = [[UIAlertView alloc]
initWithTitle:nil message:@"Ahoj, moje aplikace!"
delegate:self cancelButtonTitle:@"Cancel"
otherButtonTitles: nil];
    [alert show];
}
@end
```



Obrázek 14 - soubor s implementaci třídy

Příklad definování třídy Person. Nejprve se podívejme na soubor *Person.h*, který obsahuje vlastnosti (atributy) třídy a hlavičky metod (zpráv). V jazyku Objectvie-C je definována třída *NSObject*, která slouží jako univerzální předek pro nové třídy.

```
#import <Foundation/Foundation.h>

@interface Person : NSObject
{
    NSString *name;
    int age;
}

- (id) initWithName:(NSString *)newName age:(int)newAge;

@end
```

V souboru *Person.m* jsou definovány jednotlivé metody.

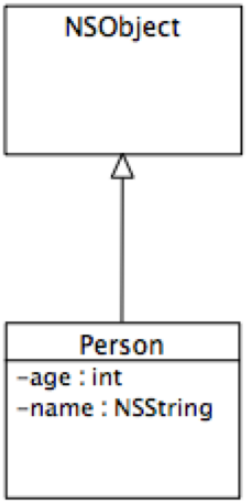
```
#import "Person.h"

@implementation Person
- (id) initWithName:(NSString *)newName age:(int) newAge
{
    self = [super init];
    if (self != nil)
    {
        name = newName;
        age = newAge;
    }
    return self;
}
//další případné definice ...
@end
```

V hlavním modulu můžeme definovat konkrétní objekt třídy *Person*.

```
#import <Foundation/Foundation.h>
#import "Person.h"

int main(int argc, const char * argv[])
{

  @autoreleasepool {
  Person * p = [[Person alloc] initWithName:@"Karel"
age:9];

  }
  return 0;
}
```

Zasílání zpráv (volání metod) se provádí následujícím způsobem:

```
//zpráva bez argumentů
[object message];

//zpráva s jedním argumentem
[object message : argument];

//zpráva se více argumenty
[object message : arg1 agument2 : arg2];

//zpráva vrací hodnotu
int value = [object message];
```



Obrázek 15 - diagram tříd, jednoduchá dědičnost

Vlastnosti jazyka Objective-C:

- nerozlišuje public a private zprávy (metody)
- nerozlišuje mezi virtuálními a nevirtuálními zprávami (metodami)
- nepoužívá destruktor
- nepoužívá const u zpráv
- nový preprocesor
- symbol @ pro prvky jazyka Objective-C
- static binding, dynamic binding
- jazyk nepoužívá reference, ale jen ukazatele
- nemá šablony, namespace

Protocol

Jazyk Objective-C využívá protokoly (Protocol), které jsou vlastně seznamem metod sdílených třídami. Jedná se vlastně o konstrukci *interface* využívaných v jazyku Java.

Protocol obsahuje deklarace metod a využívá klíčové slovo *@protocol*. Metody mohou být:

@require – povinné, je nutné je ve třídách implementovat

@optional – nepovinné

```
//  FirstProtocol.h
#import <Foundation/Foundation.h>

@protocol FirstProtocol <NSObject>
- (NSString *) methodFirstProtocol;
@end


//  SecondProtocol.h
#import <Foundation/Foundation.h>

@protocol SecondProtocol <NSObject>
- (NSString *) methodSecondProtocol;
@end

//  MyClass.h
```

```
#import <Foundation/Foundation.h>
#import "FirstProtocol.h"
#import "SecondProtocol.h"

@interface MyClass : NSObject <FirstProtocol,
SecondProtocol>
//...
@end

//  MyClass.m
#import "MyClass.h"

@implementation MyClass

- (NSString *) methodFirstProtocol
{
    return (@"methodFirstProtocol");
}

- (NSString *) methodSecondProtocol
{
    return  @"methodSecondProtocol";
}
@end
```

Obrázek 16 - chybějící definice metody z protokolu vyvolá upozornění

Pro vytváření mobilních aplikací pro iOS nabízí firma Apple framework Cocoa Touch, který nabízí prostředky pro jednodušší tvorbu. Framework zároveň podporuje implementaci vhodné architektury programu. Jako první se jedná o architekturu nazvanou Model – View – Controller (MVC), která rozděluje aplikaci na tři relativně samostatné části:

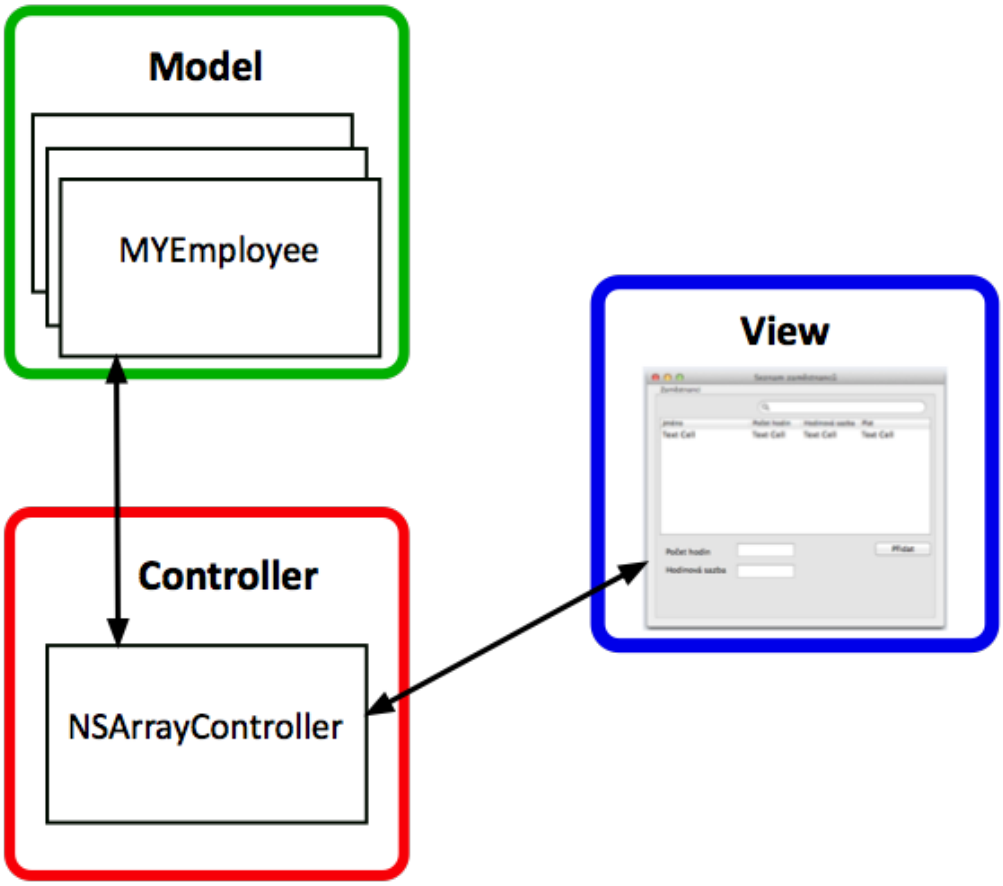
1. Model – popisuje datovou část aplikace
2. View – část uživatelského rozhrání (vzhledu aplikace)
3. Controller – se zabývá řídící logikou aplikace

Základní důvody rozdělení aplikace na tyto tři samostatné části jsou:

- větší přehlednost a logičtější uspořádání kódu,
- možnost nahradit jednu část aplikace, aniž bychom museli upravovat další části (například možnost měnit vzhled aplikace bez nutnosti zasahovat do řídící logiky a datového modelu),
- jednodušší práce v týmech.

Na následujícím obrázku je ukázka uspořádání jednoduché aplikace podle MVC. Program simuluje jednoduchou databázi zaměstnanců. V části Model je definována třída MYEmployee, která popisuje základní vlastnosti a schopnosti zaměstnance. V části View

je pouze definován vzhled aplikace. Popisuje, ze kterých oken se aplikace skládá a jaké ovládací prvky obsahuje. Část Controller obsahuje řídící logiku ve třídě NSArrayController, ve které je nadefinováno, jak se pracuje s jednotlivými objekty (zaměstnanci).



Obrázek 17 - Architektura Model - View - Controler

Kontrolní otázky

Jaké soubory je potřeba vytvořit při sestavovní nové třídy v jazyce Objective-C?

Shrnutí kapitoly

Objetive-C patří mezi objektově orientované programovací jazyky.
Třída vyjadřuje obecný koncept. Objekt je již konkrétní jev, instance dané třídy.
Základní architektura aplikace by měla vycházet z návrhu Model – View – Controller.

3. XCODE A PRVNÍ APLIKACE

V této kapitole se dozvíte:

cílem této lekce vytvořit první aplikaci a seznámit se s vývojovými nástroji.

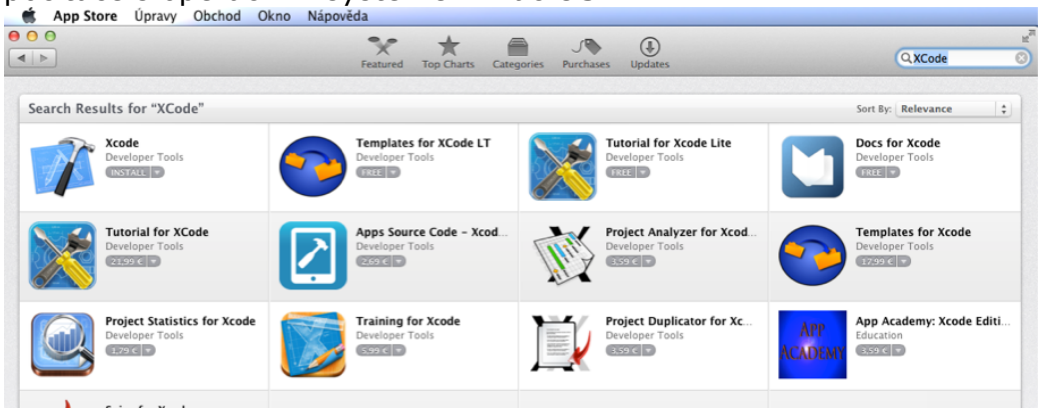
Po absolvování lekce budete:

- umět pracovat s vývojovým nástrojem Xcode,
- umět vytvářet jednoduché aplikace.

Klíčová slova této kapitoly:
akce, Objective-C, outlet, program, Xcode

Čas ke studiu: 3 hodiny

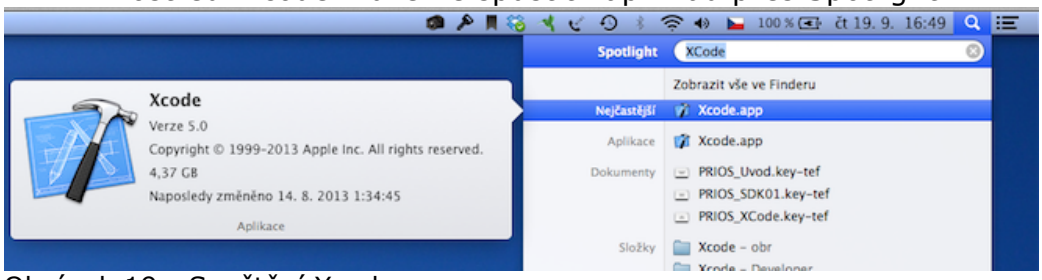
K vývoji mobilních aplikací pro operační systém iOS je potřeba zvolit vhodný programovací jazyk a vývojový nástroj. Firma Apple pro vývojáře nabízí vývojové prostředí Xcode a příslušné SDK, které lze zdarma stáhnout z App Storu. Prostředí lze nainstalovat pouze na počítače s operačním systémem Mac OS X.



Obrázek 18 - Vývojové prostředí Xcode v App Store

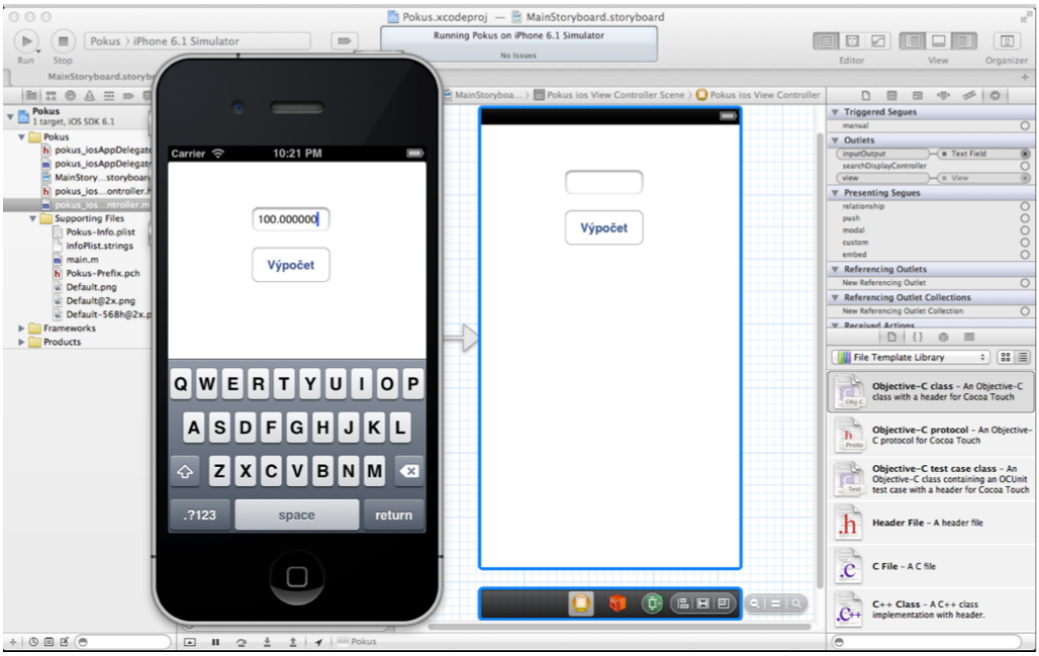
Vývojové prostředí obsahuje překladač jazyka C/C++ a objektově orientovaného programovacího jazyka Objective-C.

Prostředí Xcode můžeme spustit například přes Spotlight.



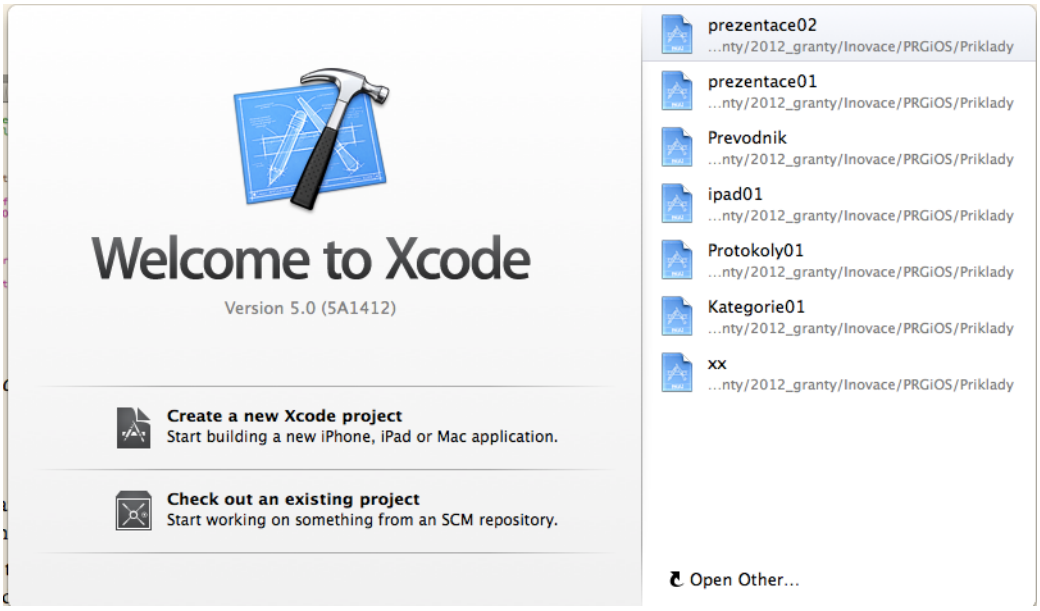
Obrázek 19 - Spuštění Xcode

K testování vytvořených aplikací můžeme používat reálná mobilní zařízení (vyžaduje developer účet) nebo softwarový simulátor.



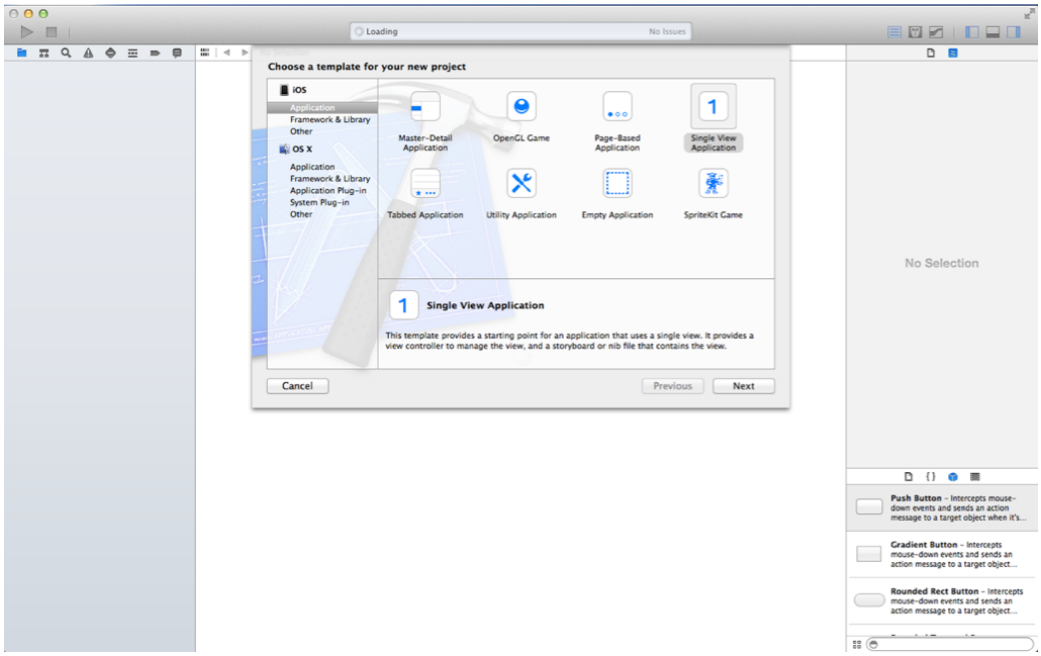
Obrázek 20 - spuštěný simulátor iPhonu (starší verze)

Po spuštění Xcode se objeví dialogové okno, ve kterém si vybereme, zda chceme vytvořit novou aplikaci nebo chceme otevřít již existující projekt.



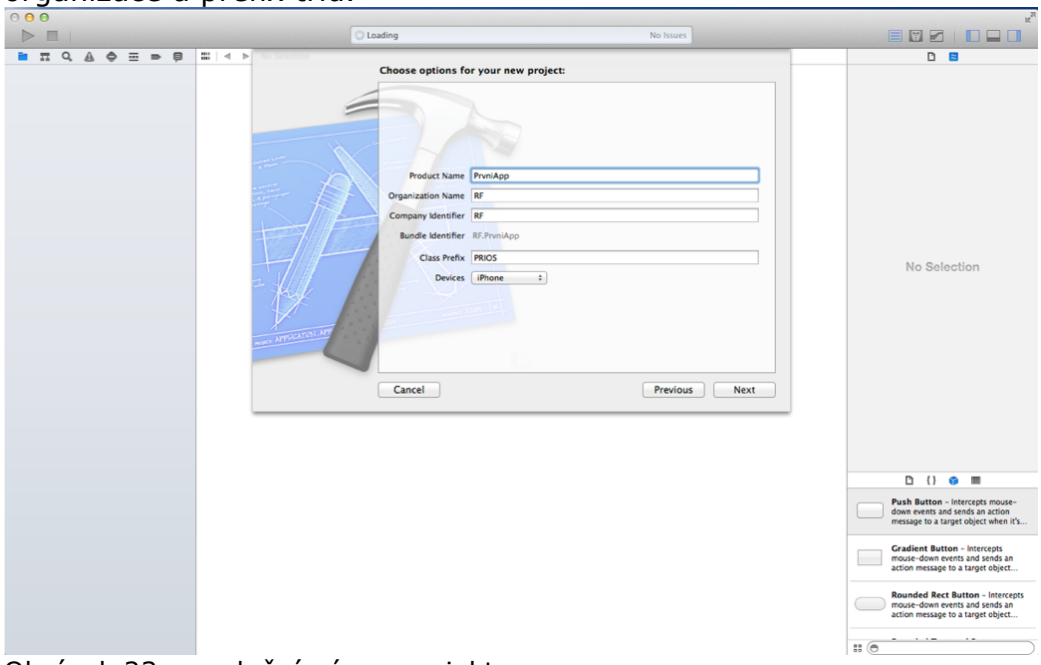
Obrázek 21 - Otevření Xcode

Následuje výběr typu aplikace. Pro tvorbu mobilních aplikací je potřeba vytvořit projekt pro iOS a nejprve vytvoříme aplikaci typu *Single View Application*.



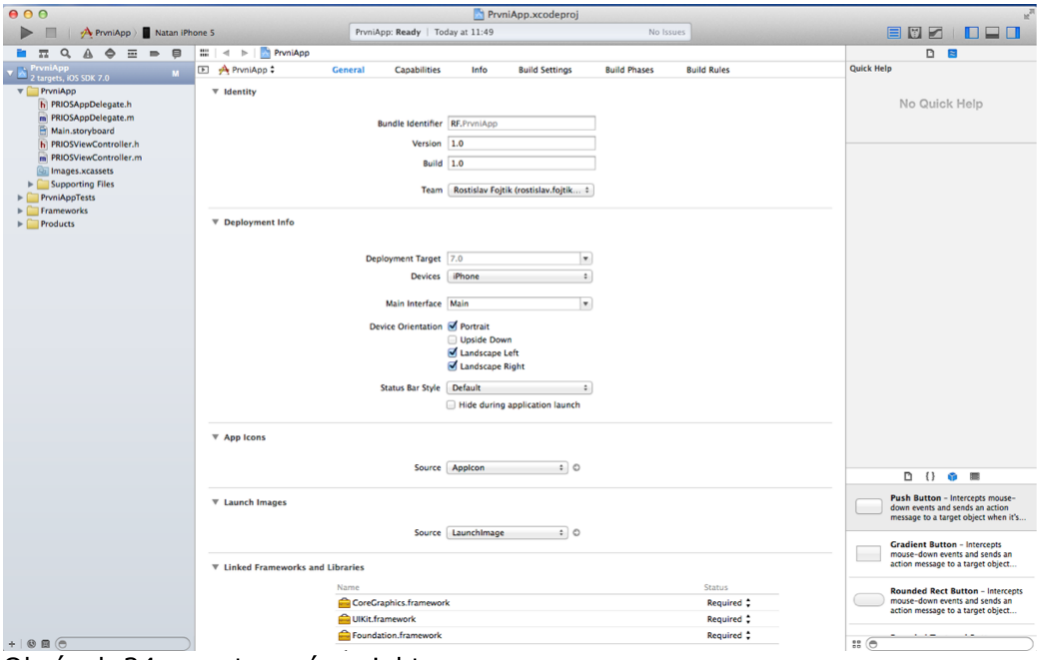
Obrázek 22 - výběr typu projektu

Dále je potřeba vyplnit název projektu, případně název organizace a prefix tříd.



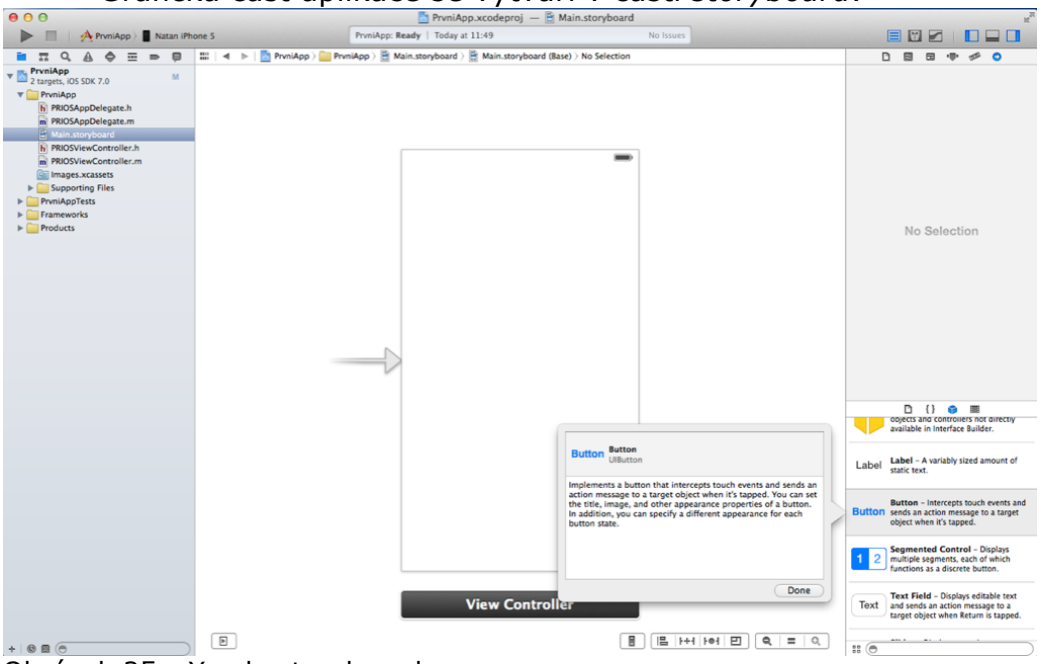
Obrázek 23 - vyplnění názvu projektu

V hlavním nastavení aplikace je potřeba zvolit zařízení, ve kterém aplikace má pracovat (iPhone nebo iPad).



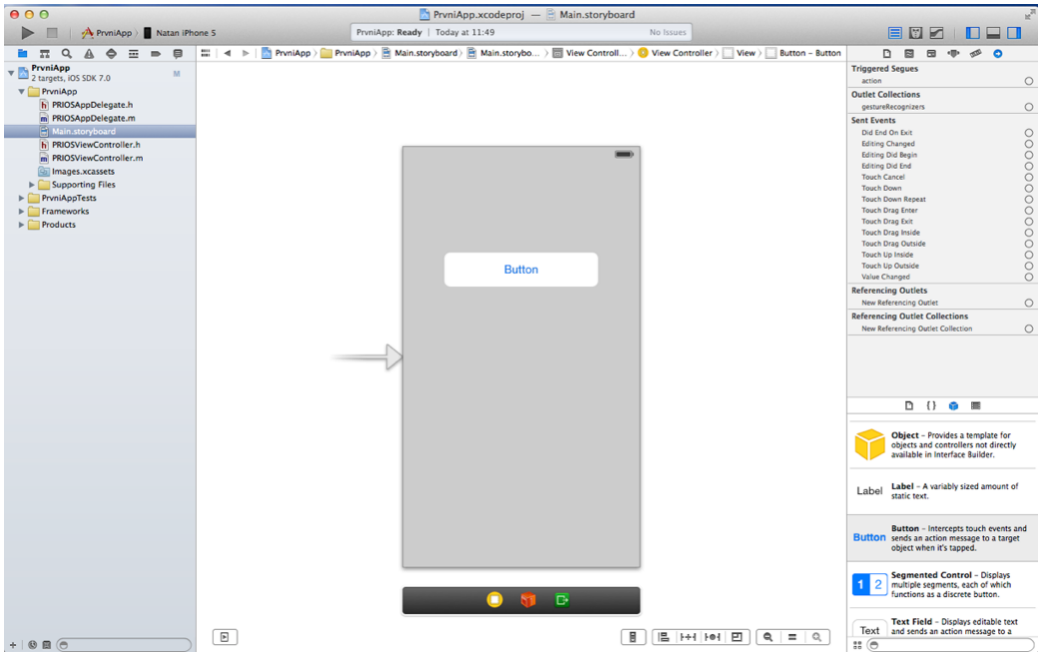
Obrázek 24 - nastavení projektu

Grafická část aplikace se vytváří v části *storyboard*.



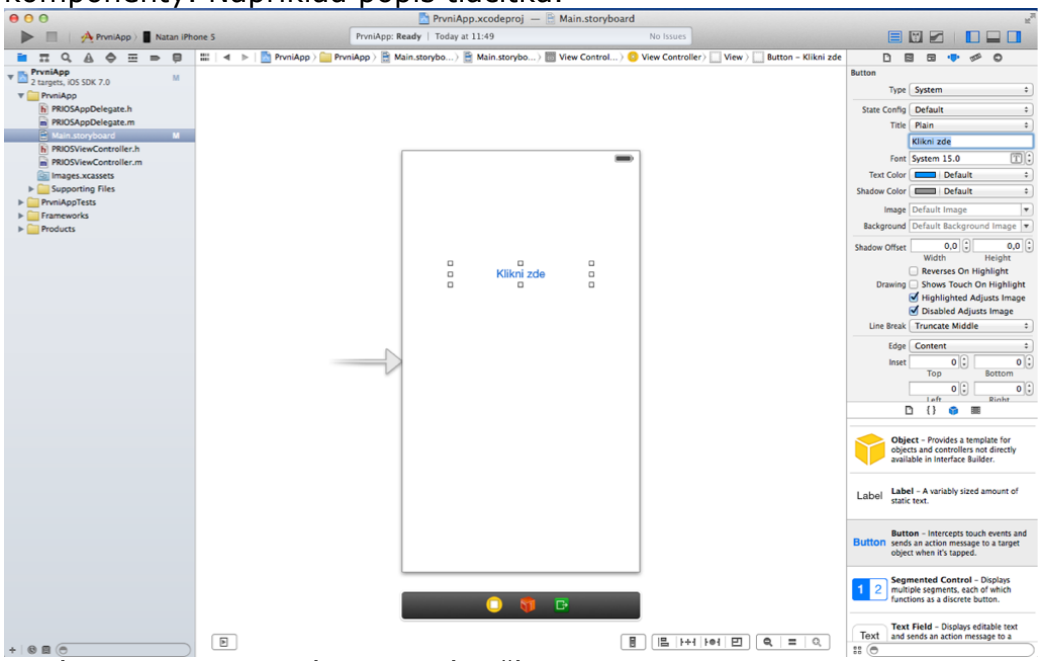
Obrázek 25 - Xcode storyboard

V seznamu grafických komponent si vybereme tlačítko a umístíme jej do prostředí aplikace. Pomoci myši můžeme upravit velikost a umístění tlačítka.



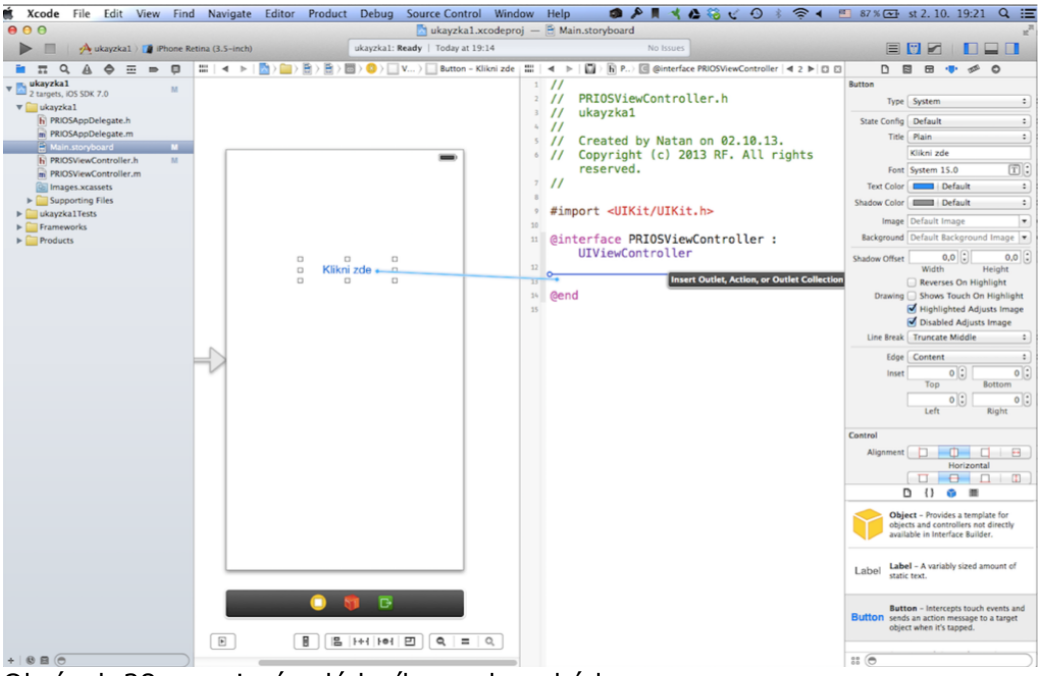
Obrázek 26 - Vložení ovládacího prvku do prostředí aplikace

V pravém horním rohu Xcode si nechte zobrazit *Attributes inspectors*, ve kterém můžeme nastavit vlastnosti konkrétní komponenty. Například popis tlačítka.



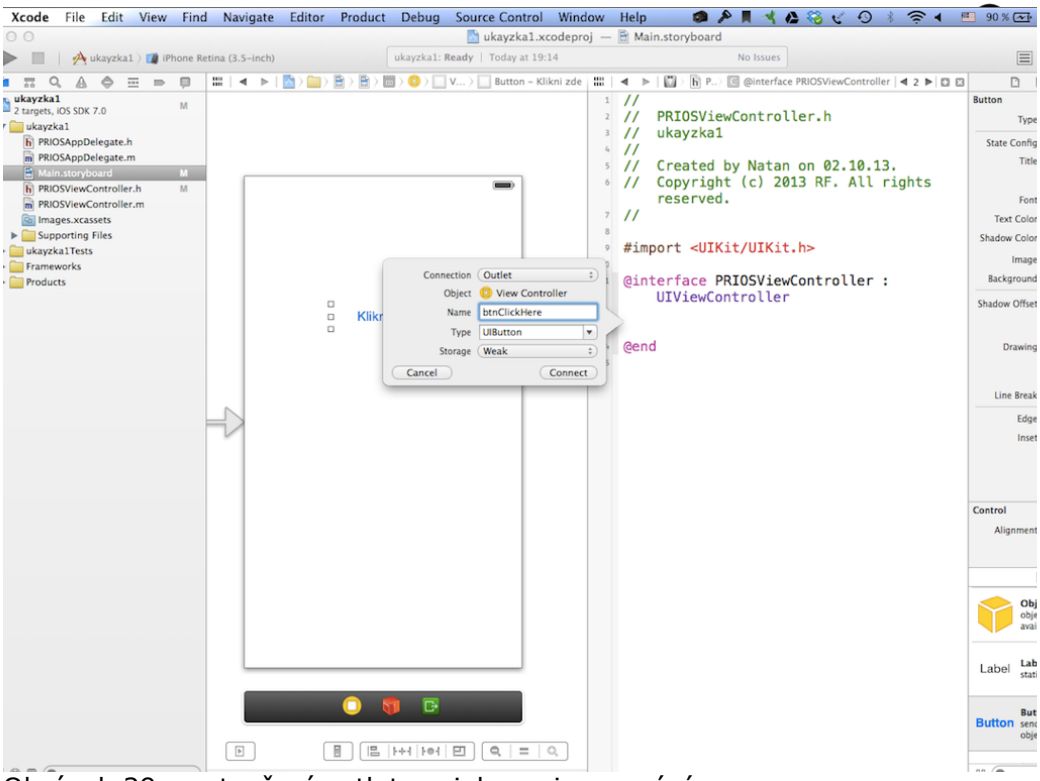
Obrázek 27 - nastavení vlastností tlačítka

V pravé horní části klikněte na ikonku *Show the Assistant Editor*. Prostřední část Xcode se rozdělí na dvě části. V levé její části je stále vidět vzhled budoucí aplikace (storyboard) a v pravé se objeví zdrojový kód. Zobrazte si kód souboru *ViewController.h*. Najeďte myší na tlačítko ve vytvářené aplikaci. Zmáčkněte pravé tlačítko myši a přesuňte se do části s kódem. Objeví se modrá vodicí úsečka. Stejného efektu můžete dosáhnout pomocí levého tlačítka myši a současného držení tlačítka Ctrl.



Obrázek 28 - spojení ovládacího prvku s kódem

Nejprve vytvoříme outlet, tedy spojení ovládacího prvku s kódem. Doplníme název outletu. *Outlet* je instanční proměnná, která je deklarována pomocí klíčového slova *IBOutlet*. Tato instanční proměnná se spojí s objektem ve storyboardu.



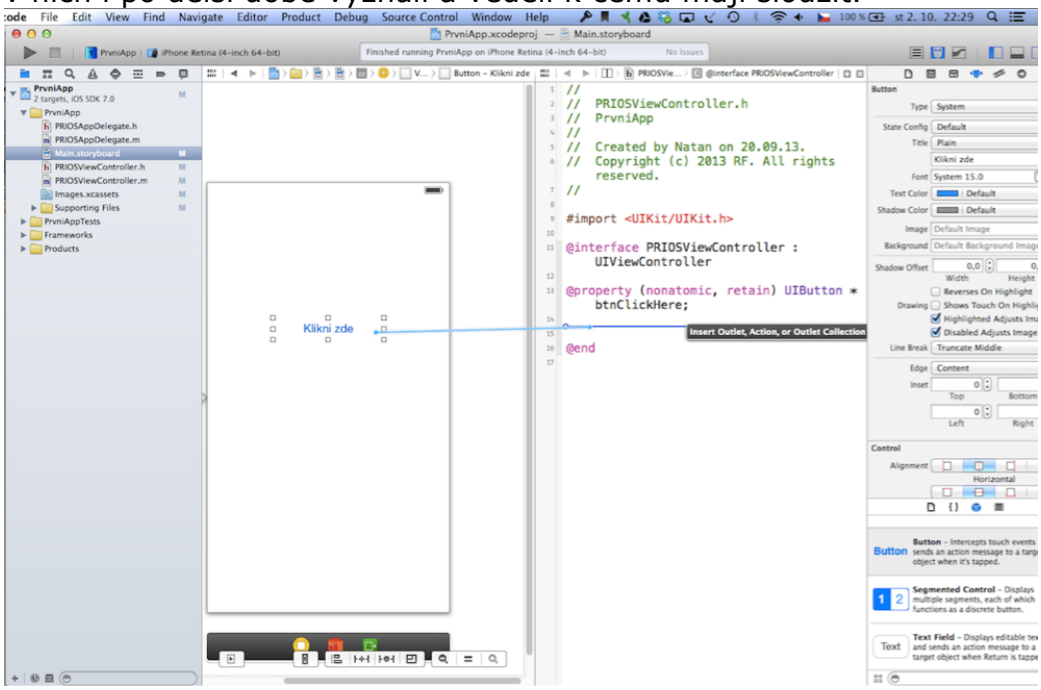
Obrázek 29 - vytvoření outletu a jeho pojmenování

Outlety zajišťují propojení ovládacích prvků s kódem. Patří k třídě *IBOutlet*.

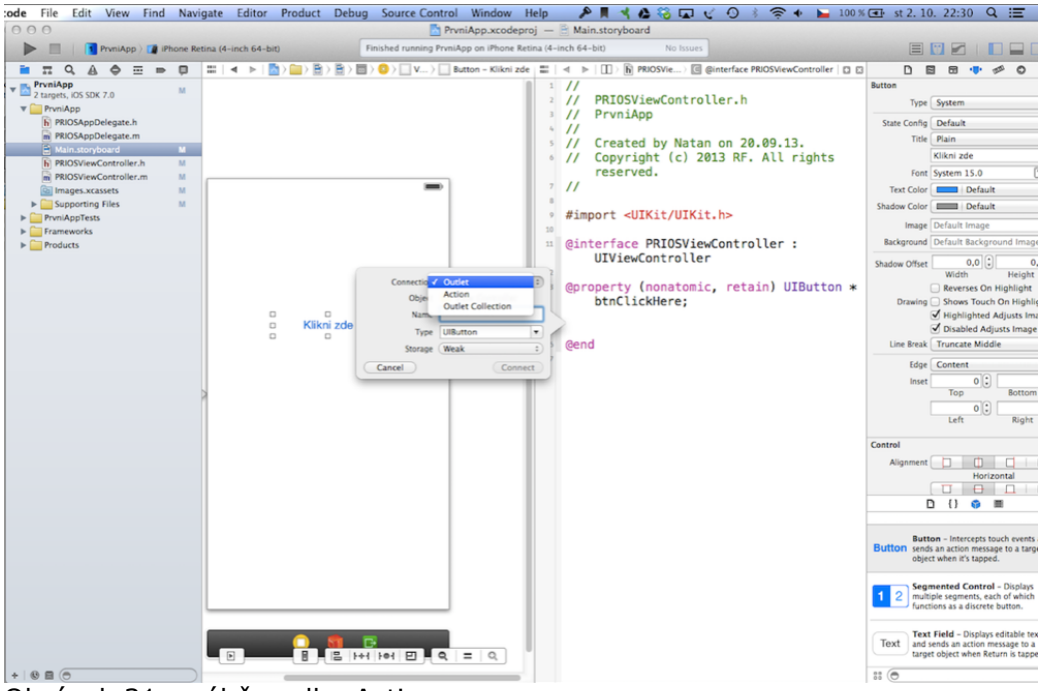
```
@property (nonatomic, retain) IBOutlet UIButton
*btnClickHere;
```

Protože tlačítko bude provádět a vyvolávat nějakou činnost (v našem příkladu spustí alert – upozornění), je potřeba vytvořit akci. Opět pomocí držení pravého tlačítka myši spojíme náš ovládací prvek s kódem. Místo volby Outlet vybereme Action a akci pojmenujeme.

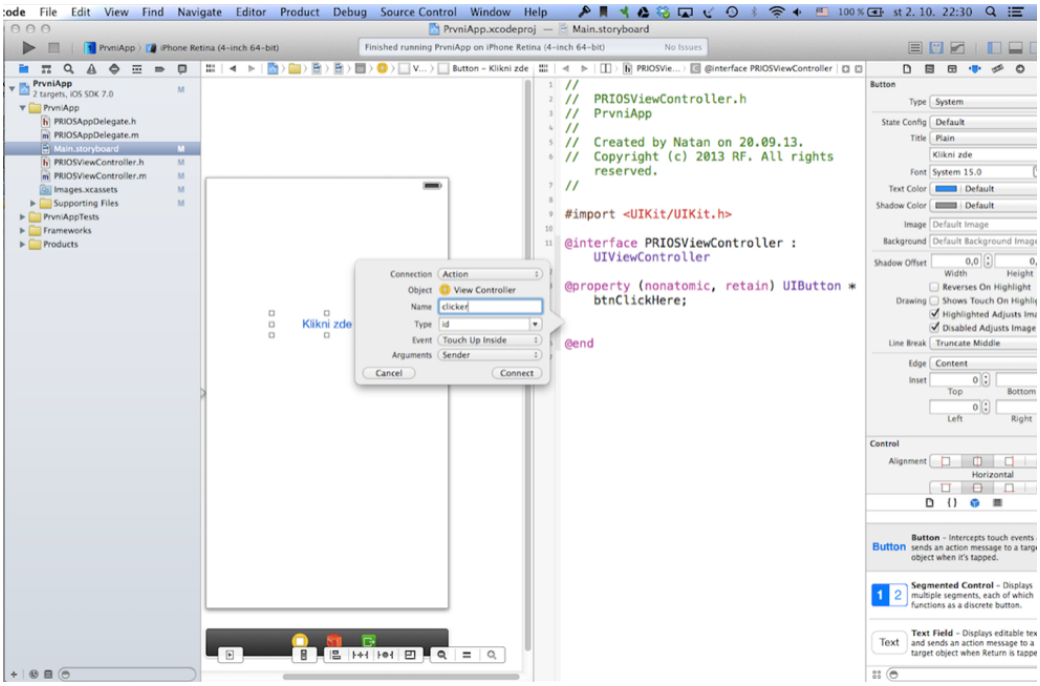
Pozor! Všechny názvy outletů i akcí definujte tak, aby jste se v nich i po delší době vyznali a věděli k čemu mají sloužit.



Obrázek 30 - vytvoření akce



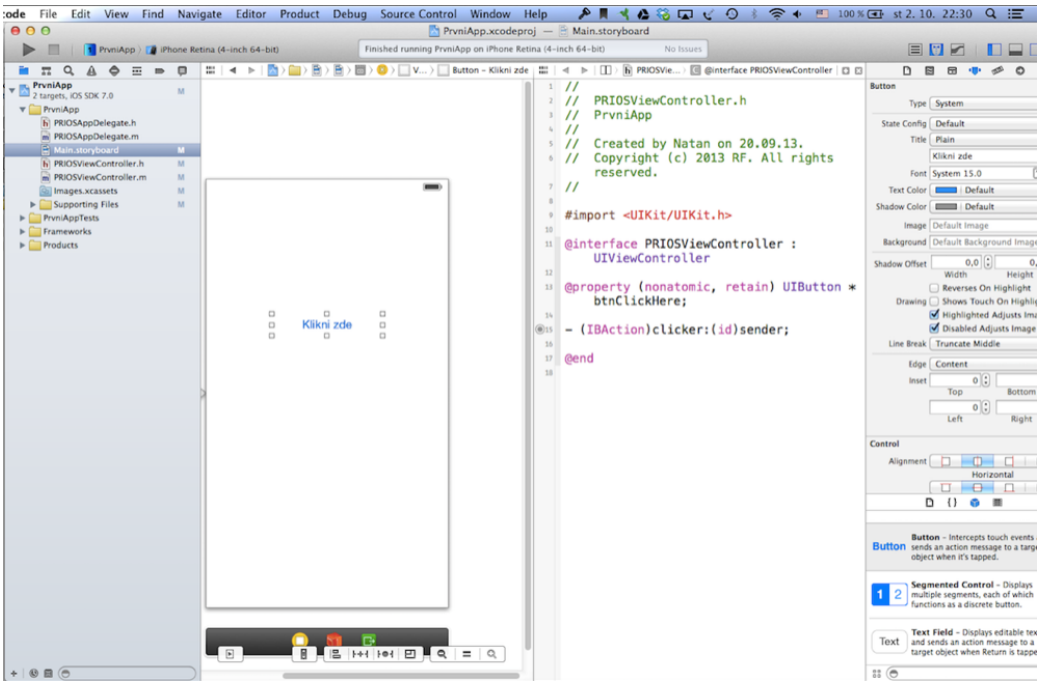
Obrázek 31 - výběr volby Action



Obrázek 32 - pojmenování konkrétní akce

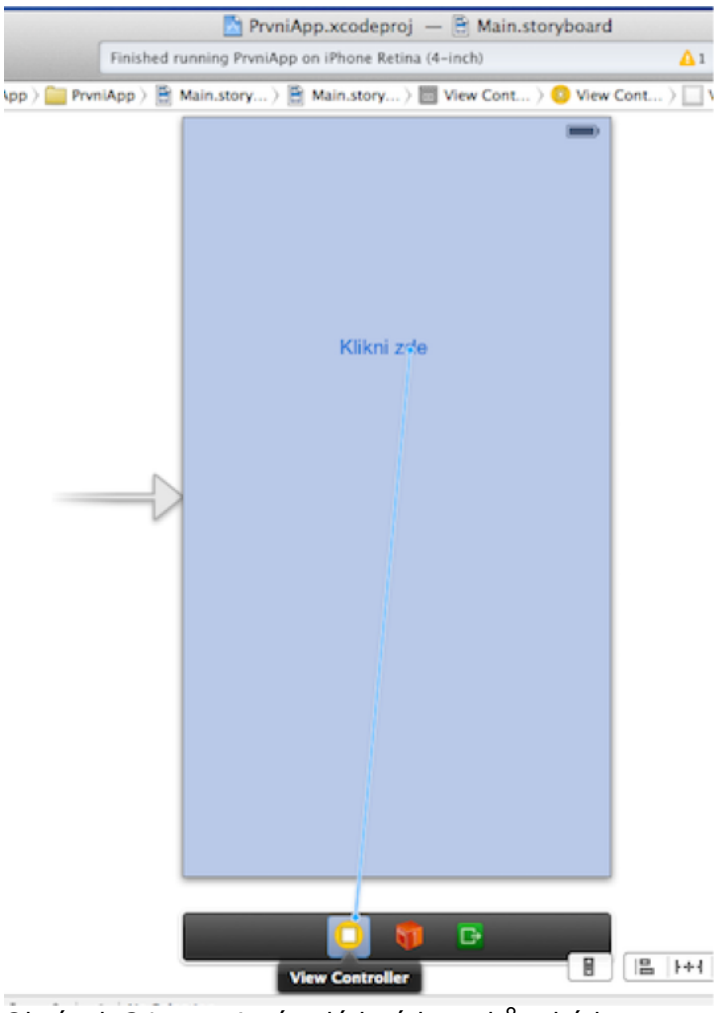
Akce jsou metody (zprávy), které jsou součástí třídy řízení. Odvozují se ze třídy *IBAction*.

- `(IBAction)clicker:(id)sender;`

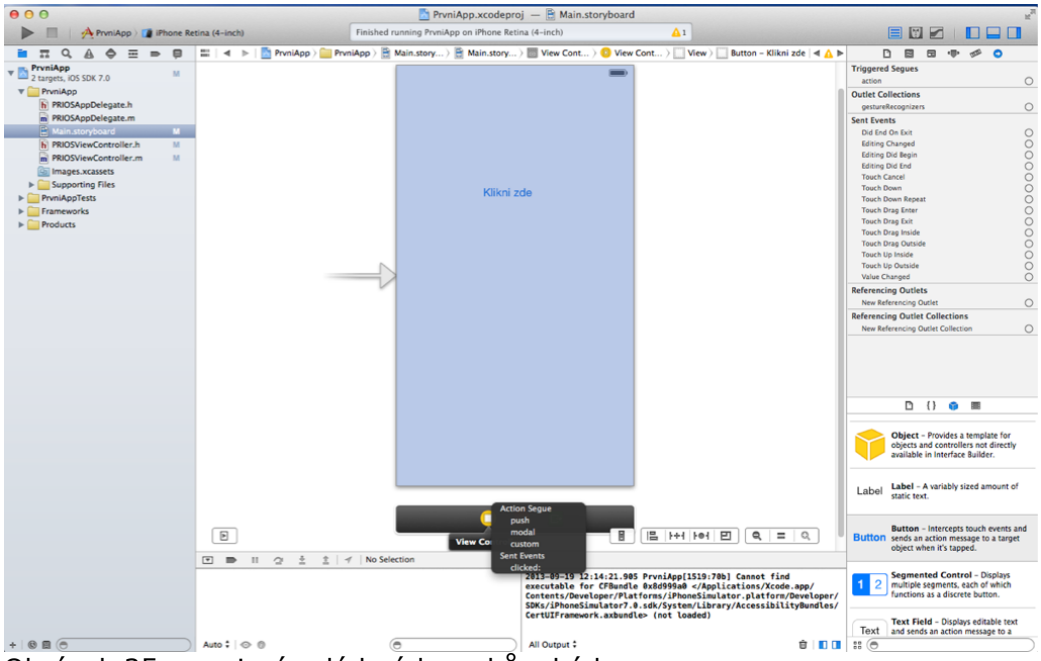


Obrázek 33 - outlet a akce v kódu

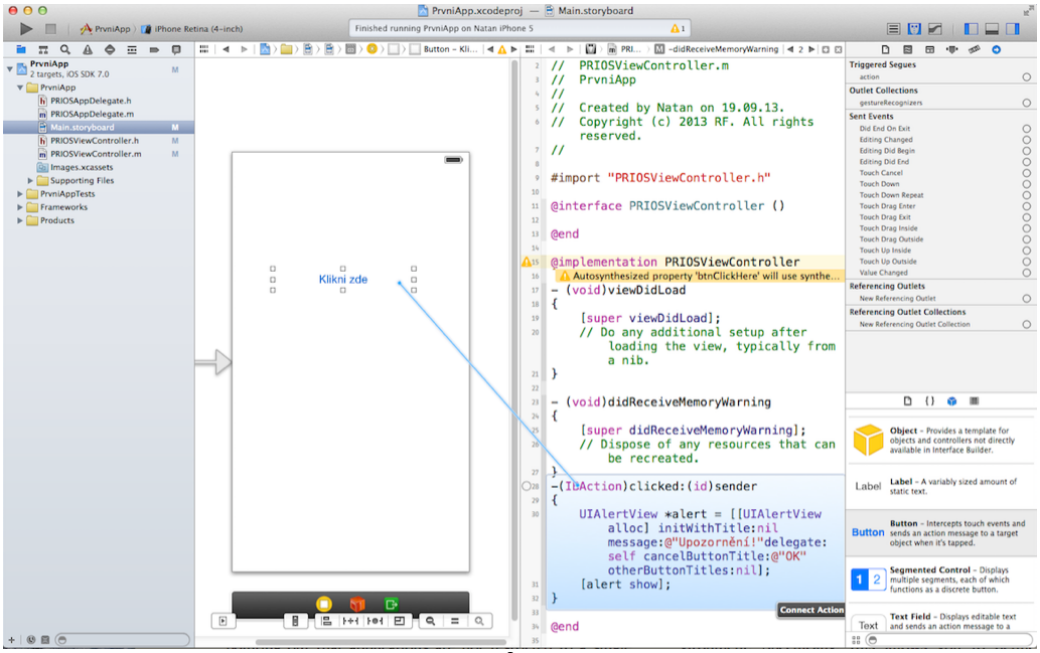
Další obrázky ukazují jiný způsob spojení ovládacích prvků s kódem.



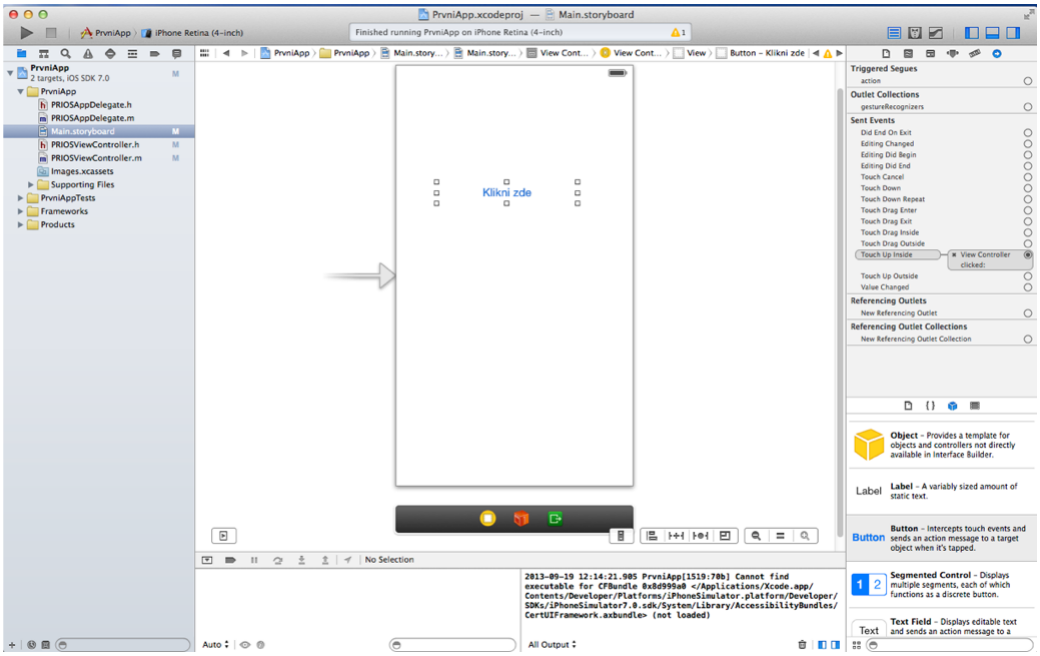
Obrázek 34 - spojení ovládacích prvků s kódem



Obrázek 35 - spojení ovládacích prvků s kódem

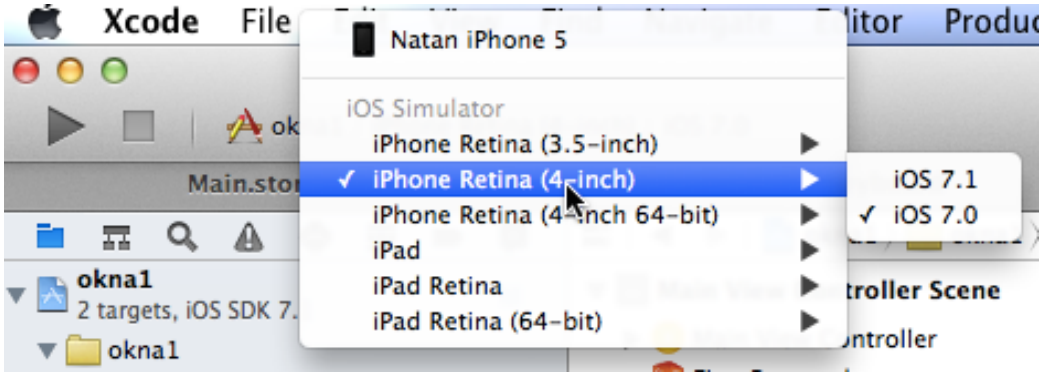


Obrázek 36 - spojení ovládacích prvků s kódem

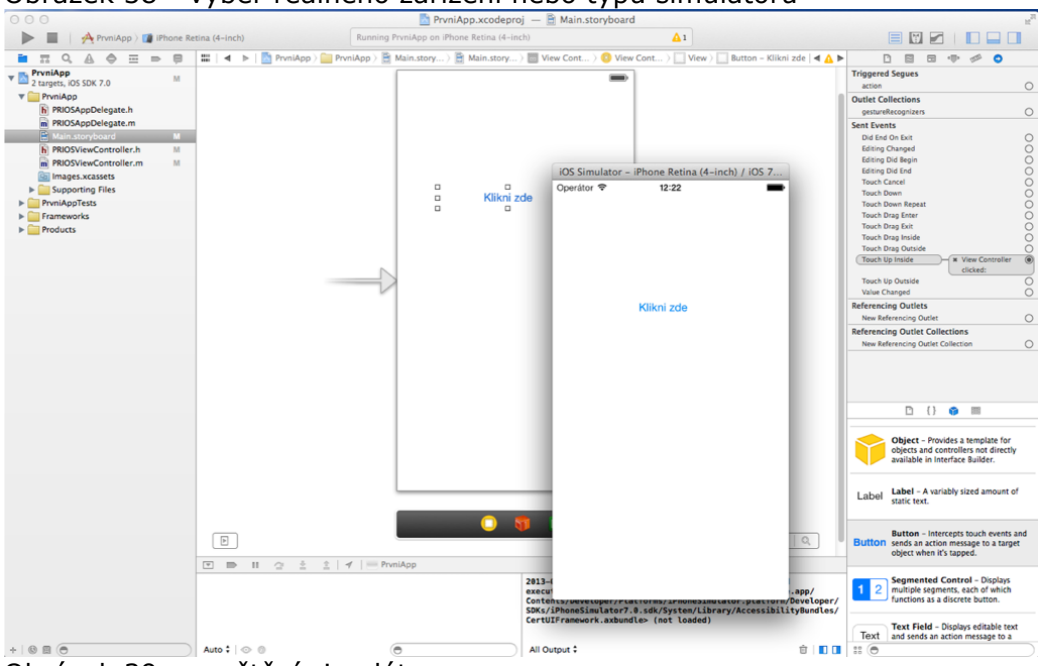


Obrázek 37 - kontrola spojení ovládacího prvku s kódem

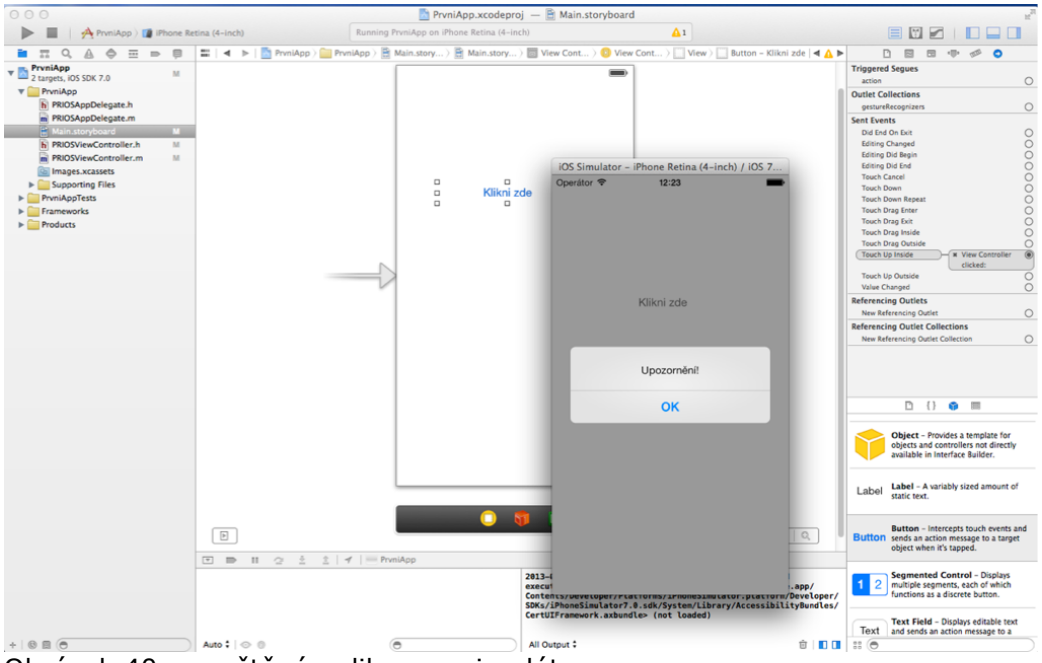
Abychom si mohli zkontrolovat, jak se bude vytvořená aplikace chovat, obsahuje vývojové prostředí rovněž simulátor zařízení iPhone nebo iPad. Pokud má programátor zaplacený vývojářský účet, může své aplikace přímo zkoušet na konkrétních fyzických mobilních zařízeních.



Obrázek 38 - výběr reálného zařízení nebo typu simulátoru



Obrázek 39 - spuštění simulátoru



Obrázek 40 - spuštěná aplikace v simulátoru

Kontrolní otázka

K čemu slouží outlet?

Shrnutí kapitoly

Programovací jazyk Objetive-C slouží k vývoji aplikací pro iOS.
Základní architektura aplikace by měla vycházet z návrhu Model – View – Controller.
Grafické rozhraní aplikace se vytváří ve storyboardu.
Outlet slouží k propojení instanční proměnné s komponentou v grafické části aplikace.
Akce je metoda (zpráva) je součástí třídy řízení.

4. ÚVOD DO COCOA TOUCH

V této kapitole se dozvíte:

Základní informace o frameworku Coca Touch.

Po absolvování lekce budete:

- znát strukturu a základní části frameworku Cocoa Touch,
- umět vytvářet jednoduché aplikace.

Klíčová slova této kapitoly:
framework, Cocoa, Cocoa Touch

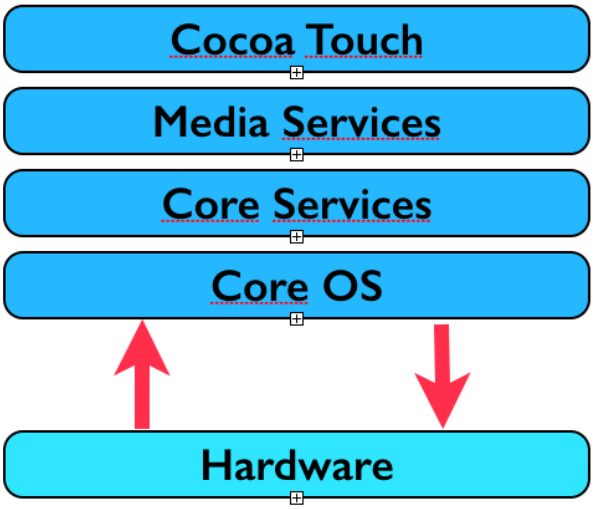
Čas ke studiu: 2 hodiny

Pro vývoj aplikací je potřeba využít příslušné SDK a frameworky. Zatímco pro operační systém Mac OS X se využívá pro podporu vývoje aplikací Cocoa, tak pro systém iOS je to Cocoa Touch.
Cocoa spojuje frameworky:

- Foundation
- Application Kit (AppKit)

Cocoa Touch obsahuje:

- Foundation
- UIKit (místo AppKit)



Obrázek 41 - architektura

- Core Animation
  - OpenGL ES
  - Core Image
  - Resolution Independent
  - Quartz 2D
  - AirPrint
  - Photo Library
- Core Audio
  - Video Playback and Capture
  - AirPlay
  - Media Library

    - Core Audio and OpenAL
- Core Data
    - SQLite
    - Sdílení dat mezi aplikacemi
    - Přístup ke kontaktům a fotografiím
    - Přístup ke kalendáři
    - XML
    - HTML5

Obrázek 42 - Cocoa a Model-View-Controller

Pro vývoj aplikací pro iOS můžeme využívat například následující frameworky a knihovny:
Cocoa Touch Layer
- UIKit.framework
- MapKit.framework
- Push Notification Service
- MessageUI.framework
- AddressUI.framework
- GameKit.framework
- iAd.framework
- EventKit.framework
- Accounts.framework
- Social.framework

iOS Media Layer
- CoreVideo.framework
- CoreText.framework
- ImageIO.framework
- AssetsLibrary.framework
- CoreGraphics.framework
- CoreImage.framework
- OpenGLES.framework
- GLKit.framework
- NewsstandKit.framework
- iOS Audio Support
- AVFoundation.framework
- Core Audio Frameworks
- Open Audio Library

- MediaPlayer.framework
- CoreMIDI.framework

iOS Core Services Layer

- AddressBook.framework
- CFNetwork.framework
- CoreData.framework
- CoreFoundation.framework
- CoreMedia.framework
- CoreTelephony.framework
- EvenKit.framework
- Foundation.framework
- CoreLocation.framework
- MobileCoreServices.framework
- StoreKit.framework
- SQLite library
- SystemConfiguration.framework
- QuickLook.framework

iOS Core OS Layer

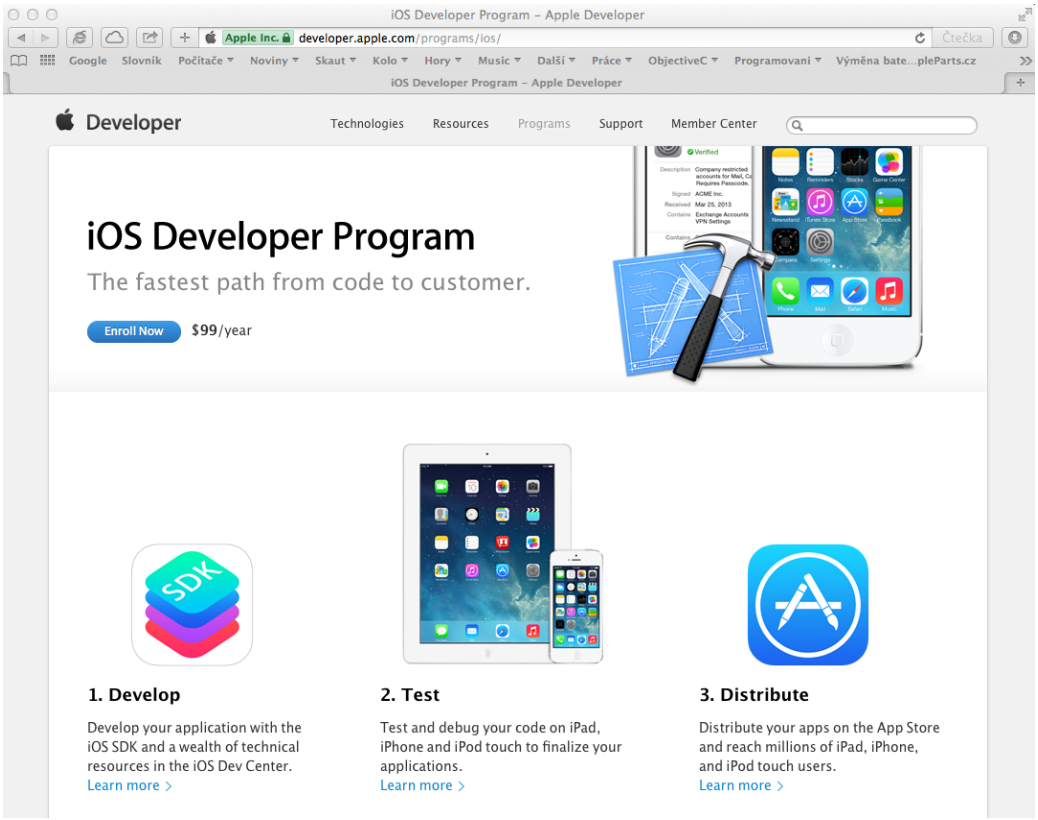
- Accelerate.framework
- ExternalAccessory.framework
- Security.framework
- System

Práce s pamětí

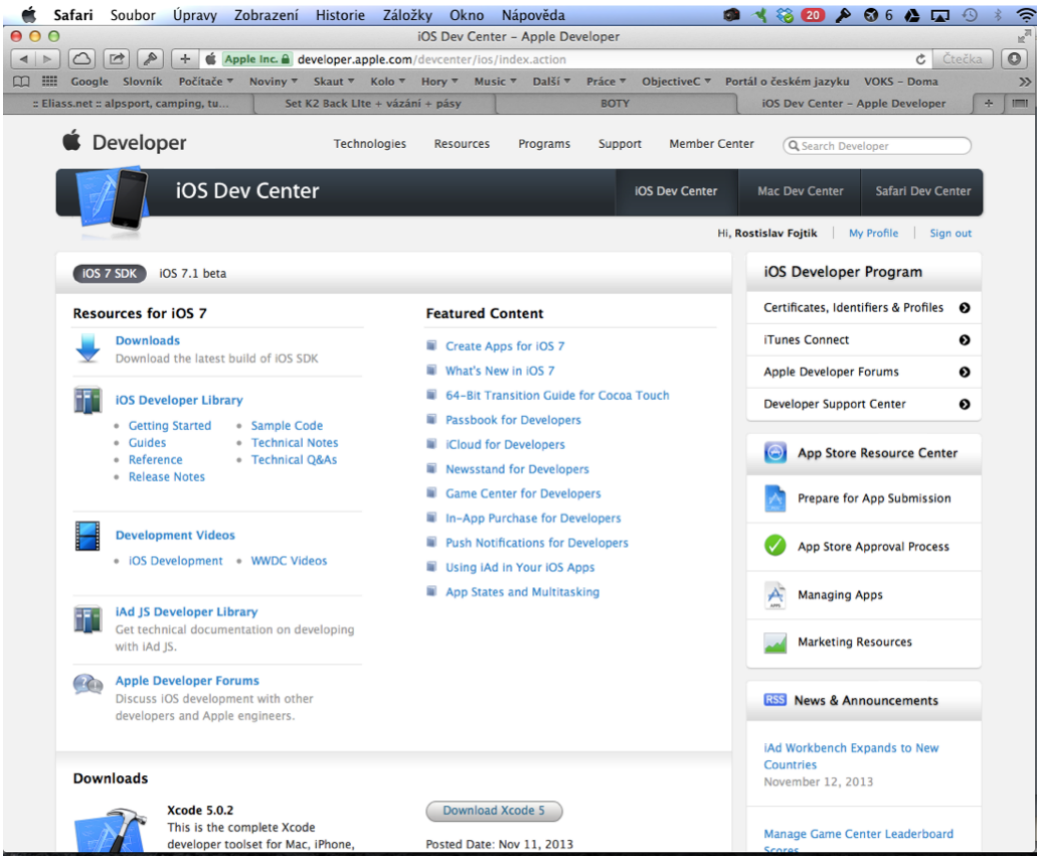
Aplikace v operačním systému iOS nevyužívají při alokaci a uvolňování paměti *Garbage Collection*, ale tzv. počítání referencí ARC (Automatic Reference Counting). Je to z důvodů menší náročnosti, které šetří výkon mobilního zařízení a tím také energie z baterií.

Developer účet

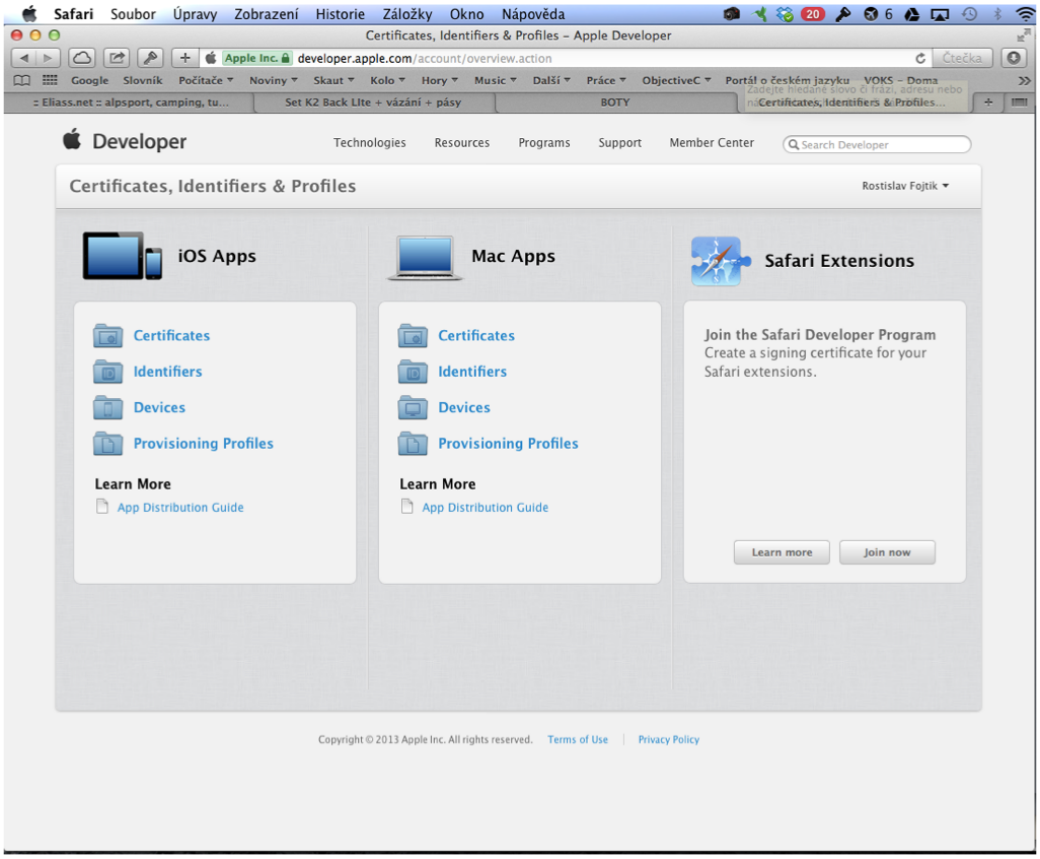
Vývojář, který chce své aplikace dále šířit, si musí zaregistrovat na stránce https://developer.apple.com vývojářský účet, který stojí $99 ročně.



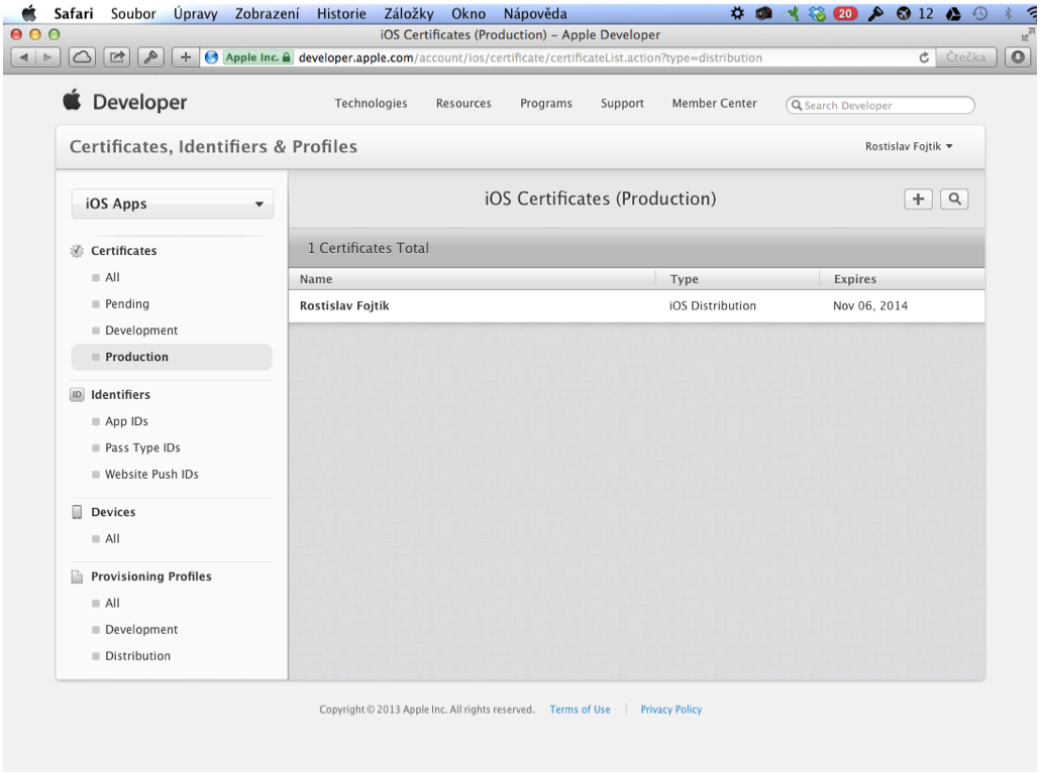
Obrázek 43 - stránky pro vývojáře iOS aplikací



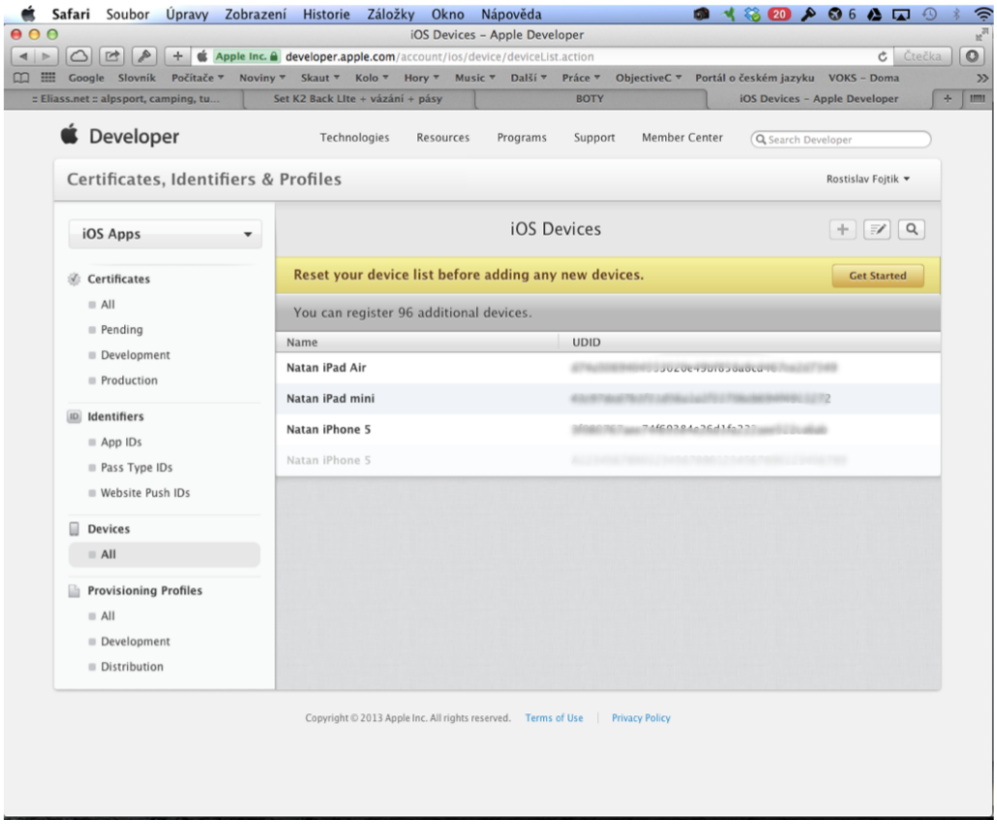
Obrázek 44 - přihlášení do vývojářského centra



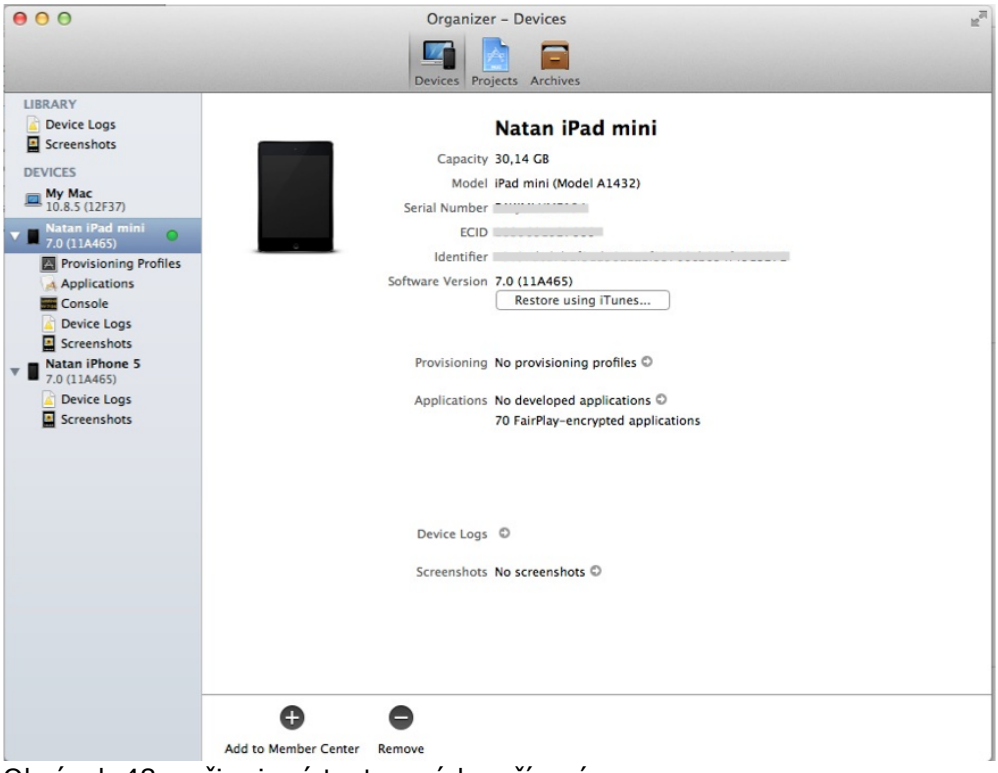
Obrázek 45 – certifikace



Obrázek 46 - připojení testovacích zařízení



Obrázek 47 - seznam testovacích zařízení



Obrázek 48 - připojené testovacích zařízení

Shrnutí kapitoly

Pro vývoj aplikací v iOS je připraven Framework Cocoa Touch.

5. APLIKACE PŘEVODNÍK

V této kapitole se dozvíte:

cílem této lekce vytvořit jednoduchou aplikaci pro převod jednotek rychlosti a seznámit se s vývojovými nástroji.

Po absolvování lekce budete:

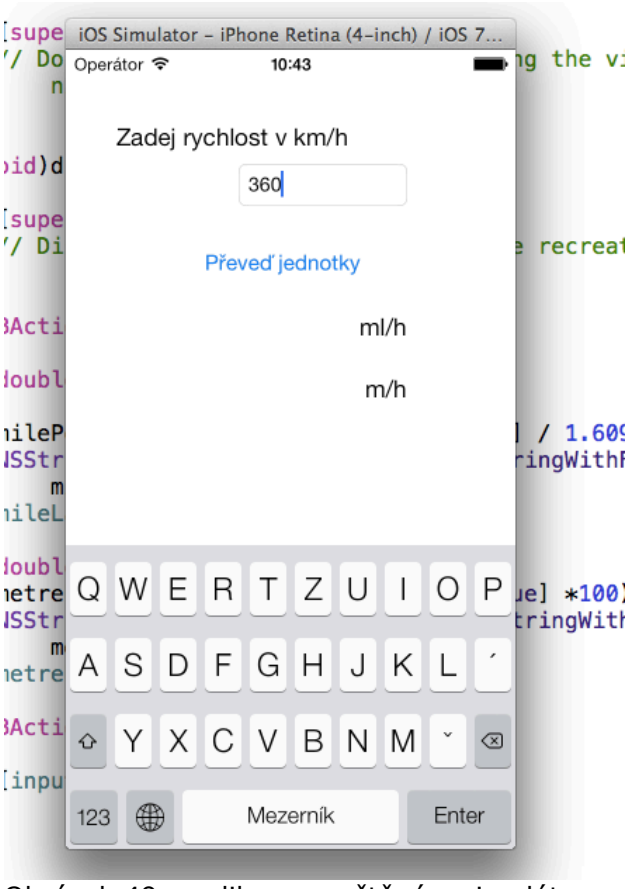
- umět pracovat s vývojovým nástrojem Xcode
- umět vytvářet jednoduché aplikace

Klíčová slova této kapitoly:
akce, Objective-C, outlet, program, Xcode

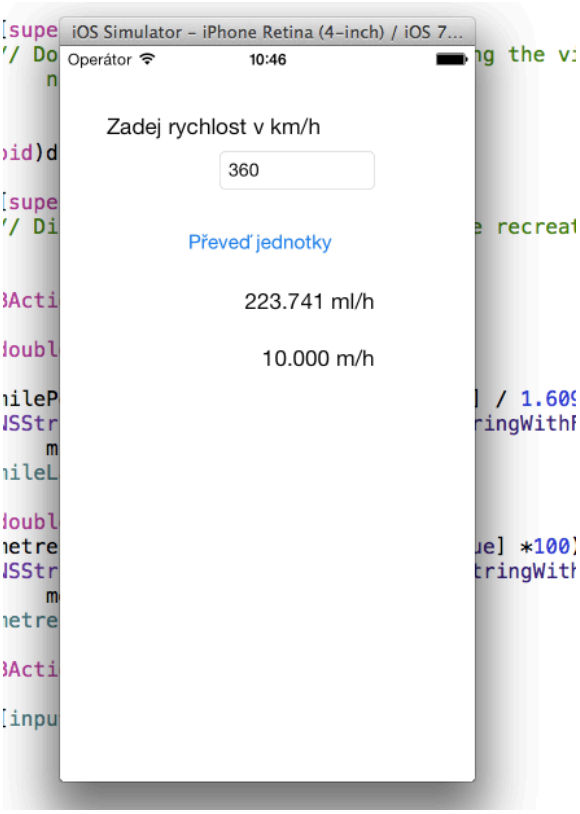
Čas ke studiu: 3 hodiny

Zadání příkladu:

Vytvořme jednoduchou aplikaci pro iPhone, která bude sloužit k převodu jednotek rychlosti. Uživatel pomocí softwarové klávesnice zadá do editačního pole rychlost v km/h a po stisknutí tlačítka, aplikace vypíše danou rychlost v mílích a metrech za hodinu. Při dotyku mimo editační pole se softwarová klávesnice skryje.

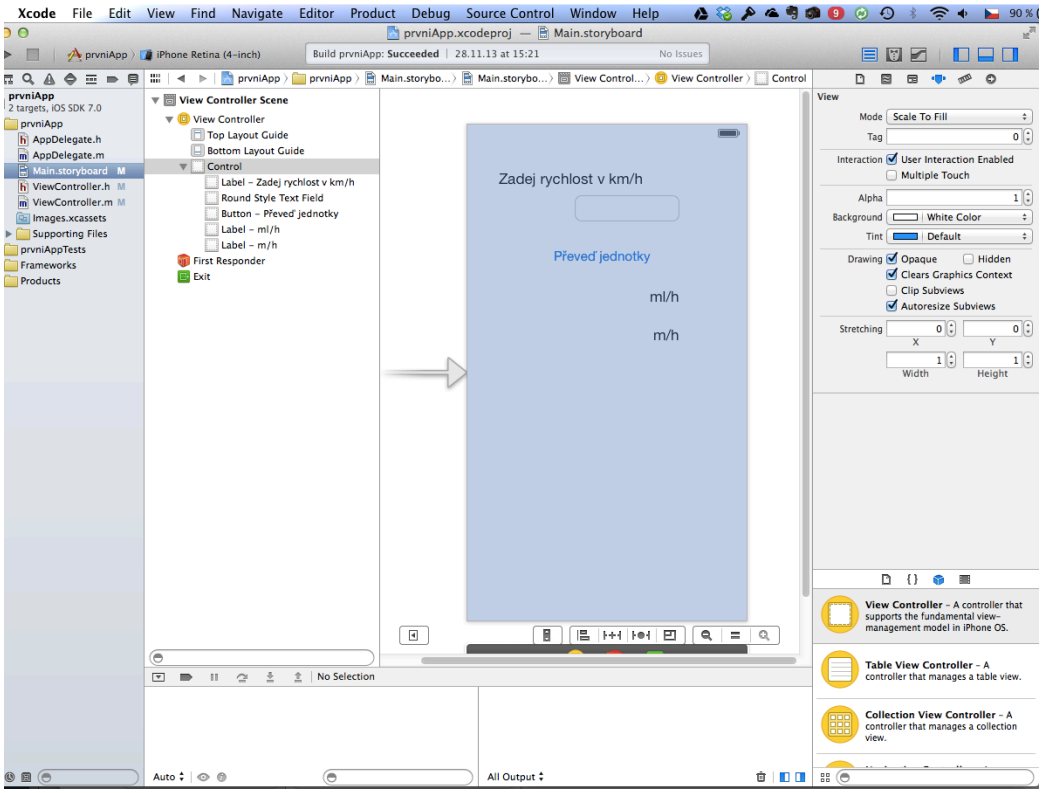


Obrázek 49 - aplikace spuštěná v simulátoru



Obrázek 50 - převedení jednotek a skrytí klávesnice v aplikaci

Vytvořme nový projekt jako *Single View Application*. Pomocí storyboardu vytvořme vzhled aplikace. Na plochu aplikace umístíme tři prvky typu *Label*, které budou sloužit k výpisu informací. Dalším vizuálním prvkem bude *Text Field*, který bude sloužit k zadávání hodnoty rychlosti v km/h. Posledním prvkem bude *Button*, pomocí kterého vyvoláme převod jednotek. Umístění a velikost jednotlivých prvků můžeme nastavit graficky pomoci myši nebo textově v *Attributes ispector* a *Size ispector*.



Obrázek 51 - uživatelské rozhraní aplikace

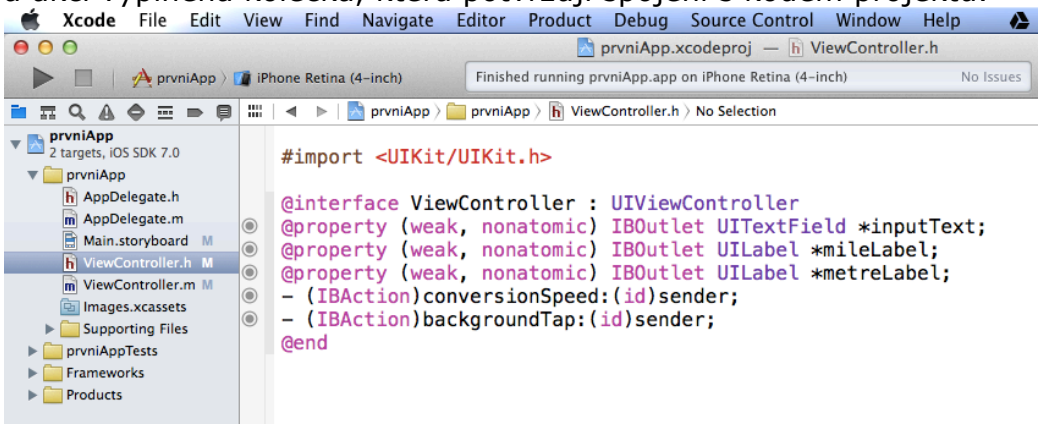
Dále je potřeba vytvořit outlety pro ovládací prvky, které se budou během aplikace měnit nebo na které se potřebujeme v kódu odkazovat.

```
@property (weak, nonatomic) IBOutlet UITextField
*inputText;
@property (weak, nonatomic) IBOutlet UILabel *mileLabel;
@property (weak, nonatomic) IBOutlet UILabel *metreLabel;
```

V aplikaci je potřeba definovat dvě akce. První při kliknutí na tlačítko, druhá při kliknuti na plochu aplikace pro skrytí klávesnice.

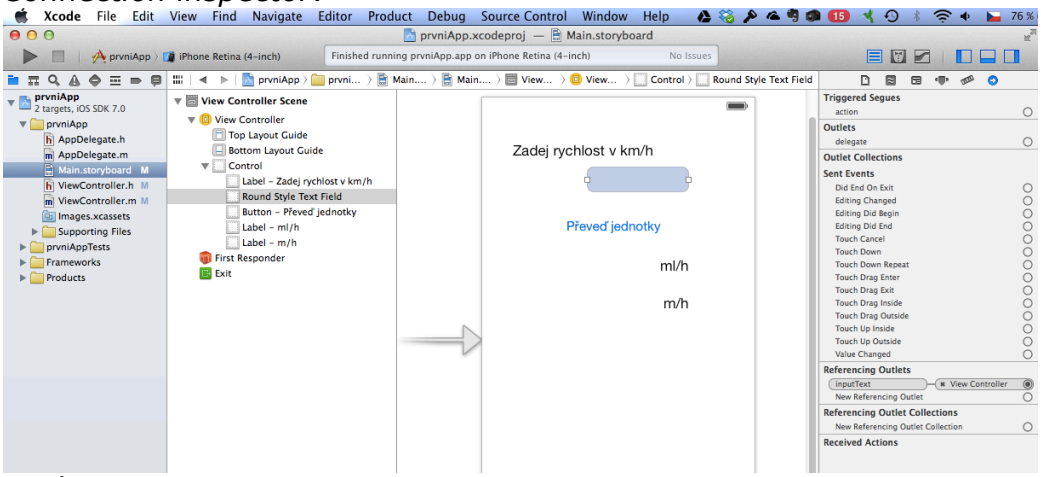
```
- (IBAction)conversionSpeed:(id)sender;
- (IBAction)backgroundTap:(id)sender;
```

Pokud jsme postupovali správně, jsou vedle hlaviček outletů a akci vyplněná kolečka, která potvrzují spojení s kódem projektu.



Obrázek 52 - outlety a akce v souboru ViewController.h

Další kontrolu správného spojení můžeme vidět v části *Connection inspector*.

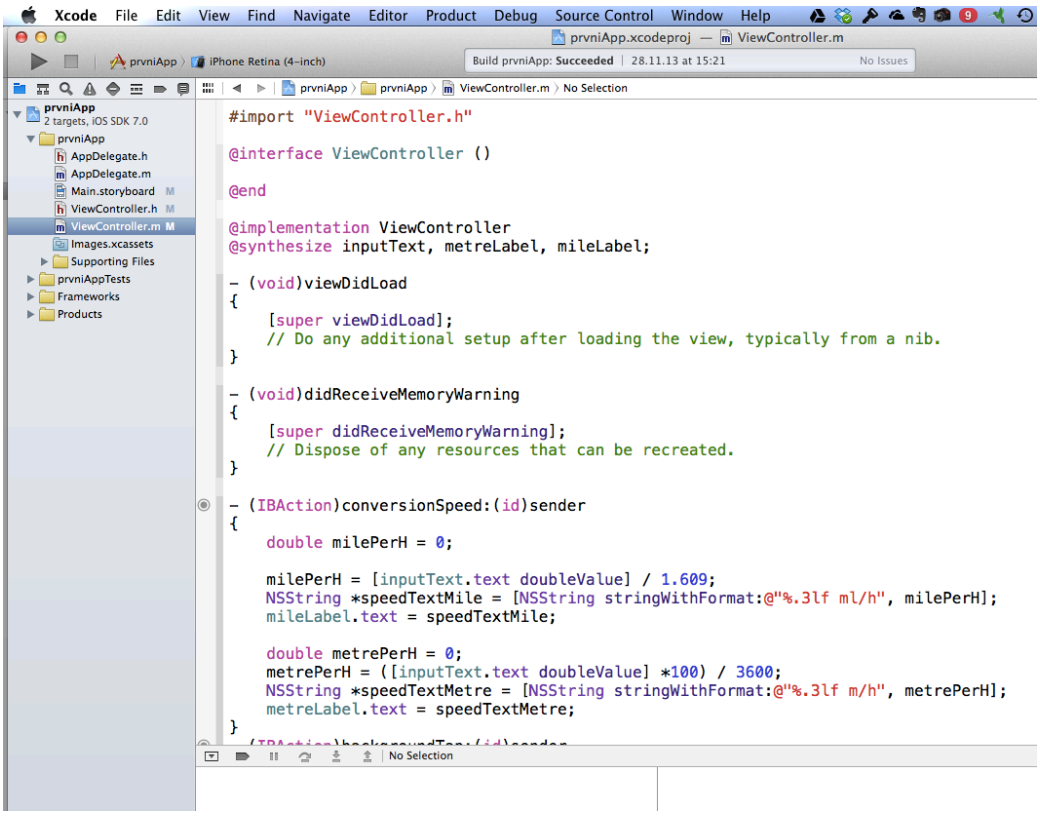


Obrázek 53 - Connection ispector

V souboru ViewController.m je potřeba dopsat kód jednotlivých akcí. Nejprve doplňte kód, který zajistí přímý přístup k outletům:

```
@synthesize inputText, metreLabel, mileLabel;
```

Následně je potřeba dopsat kód, který bude prováděn po stisknutí tlačítka v aplikaci a pak i kód pro skrytí klávesnice.



```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize inputText, metreLabel, mileLabel;

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)conversionSpeed:(id)sender
{
    double milePerH = 0;

    milePerH = [inputText.text doubleValue] / 1.609;
    NSString *speedTextMile = [NSString stringWithFormat:@"%.3lf ml/h", milePerH];
    mileLabel.text = speedTextMile;

    double metrePerH = 0;
    metrePerH = ([inputText.text doubleValue] *100) / 3600;
    NSString *speedTextMetre = [NSString stringWithFormat:@"%.3lf m/h", metrePerH];
    metreLabel.text = speedTextMetre;
}
```

Obrázek 54 - kód akcí v souboru ViewController.m

```
#import "ViewController.h"
@interface ViewController ()

@end
```

```
@implementation ViewController
@synthesize inputText, metreLabel, mileLabel;

- (void)viewDidLoad
{
    [super viewDidLoad];
// Do any additional setup after loading the view,
// typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

// akce po stiknutí tlačítka - převod jednotek
- (IBAction)conversionSpeed:(id)sender
{
    double milePerH = 0;

    milePerH = [inputText.text doubleValue] / 1.609;
    NSString *speedTextMile = [NSString
stringWithFormat:@"%.3lf ml/h", milePerH];
    mileLabel.text = speedTextMile;

    double metrePerH = 0;
    metrePerH = ([inputText.text doubleValue]*100) / 3600;
    NSString *speedTextMetre = [NSString
stringWithFormat:@"%.3lf m/h", metrePerH];
    metreLabel.text = speedTextMetre;
}

// akce po kliknuti na plochu aplikace - skrytí klávesnice
- (IBAction)backgroundTap:(id)sender
{
    [inputText resignFirstResponder];
}

@end
```

Všimněte si, že tato jednoduchá aplikace nedodržuje důsledně architekturu MVC. Data jsou vlastně jen čísla a výpočty převodu natolik jednoduché, že jsme je zapsali přímo do akce při stisknuti tlačítka. Příklad je jen ukázkový a slouží hlavně k seznámení s prostředím Xcode. U skutečných aplikací však tento přístup není vhodný, neboť znesnadní další úpravu a rozšíření kódu.

Shrnutí kapitoly

Aplikace se obvykle vyvíjí v prostředí Xcode a pomocí jazyka Objectvie-C. Aplikace jsou postaveny na architektuře Model-View-Controller.

6. PICKER

V této kapitole se dozvíte:

Cílem této lekce je seznámit se ovládacím prvkem Picker a základními možnostmi správy zdrojů dat pro aplikaci.

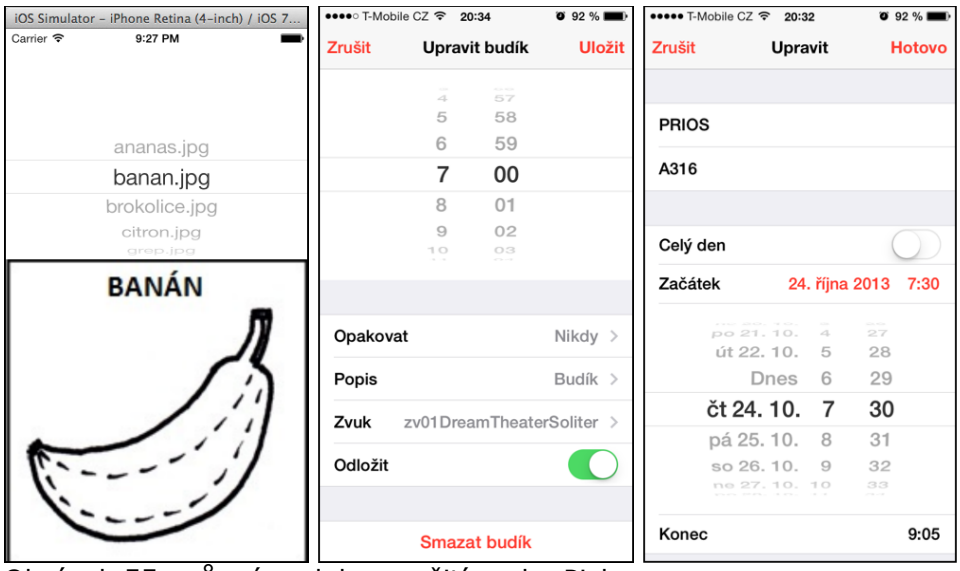
Po absolvování lekce budete:

- umět pracovat s ovládacím prvkem Picker
- umět vytvořit a využívat property list pro správu dat aplikace
- vědět k čemu slouží delegát a zdroj dat

Klíčová slova této kapitoly:
Delegát, Data Picker, Picker View, Property list

Čas ke studiu: 3 hodiny

Picker je ovládací prvek vhodný pro výběr. V desktopových systémech na PC se používá k podobným účelům prvek zvaný *Combo Box*. Ten však není příliš vhodný pro ovládání prsty na dotykovém displeji.

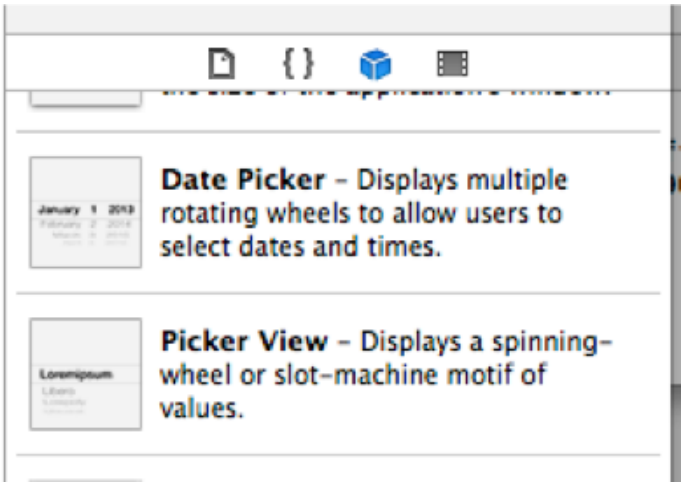


Obrázek 55 - různé podoby využití prvku Picker

Zadání příkladu:

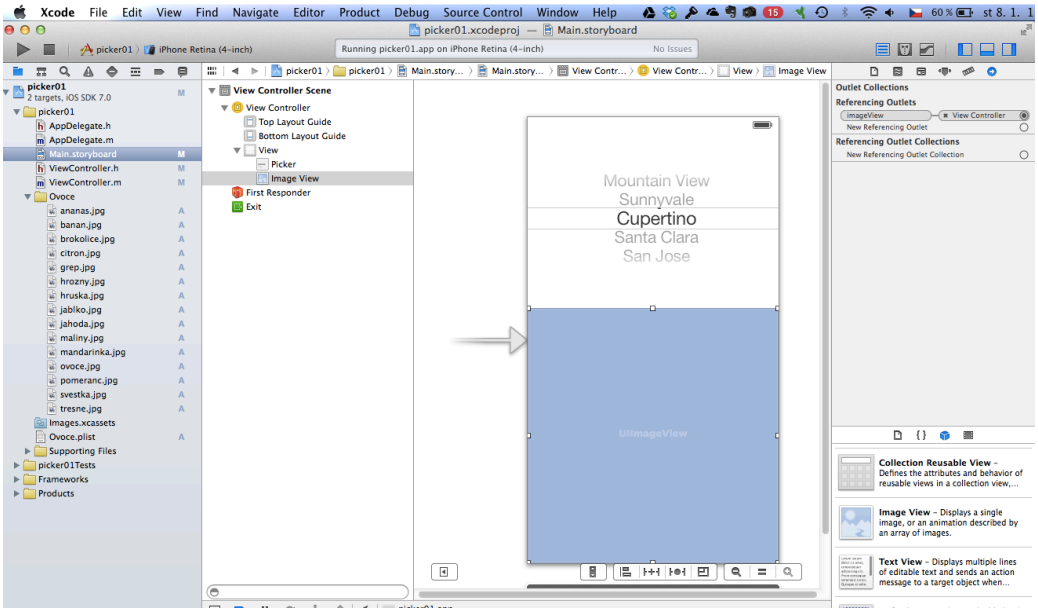
Vytvořte program, který pomocí prvku *Picker* bude zobrazovat vybraný obrázek ze sady předdefinovaných obrázků.

Nejprve vytvořme nový projekt na základě šablony *Single View Application*. *Picker* je otočný rámec pro výběr. Může mít dvojí podobu. První je předdefinovaná pro výběr časových informací (*Data Picker*), druhá je obecná (*Picker View*).



Obrázek 56 - Data Picker a Picker View

Nejprve vytvořme vzhled budoucí aplikace. Do prostředí vložme prvky *Picker View* a *Image View*. Vytvořme outlety pro oba ovládací prvky.



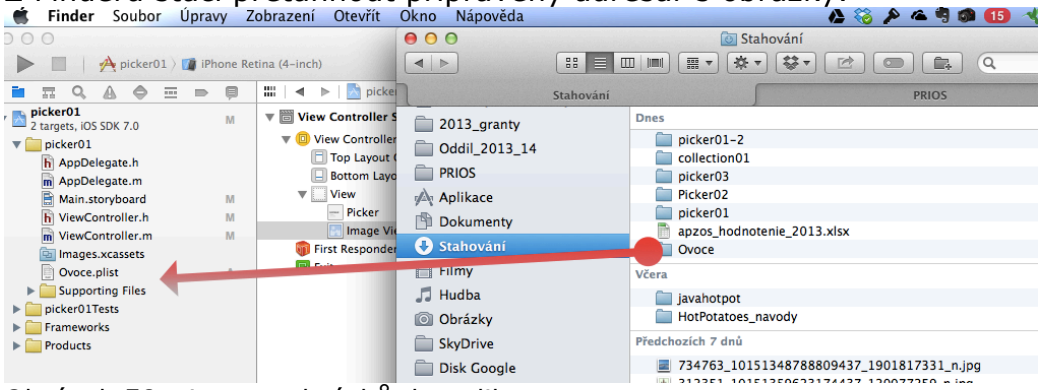
Obrázek 57 - tvorba vzhledu aplikace

Pro správnou funkci otočného rámce je potřeba stanovit delegáta a zdroj dat. Picker přenechává svému delegátu (UIPickerViewDelegate) provádění úkolů. Zdroj dat (UIPickerViewDataSource) předává Pickeru informaci, se kterými komponentami bude pracovat. UIPickerViewDelegate a UIPickerViewDataSource patří mezi *protokoly*, což jsou vlastně seznamy metod (zpráv), které má třída využívat. Je nutné je připojit v části interface k názvu třídy:

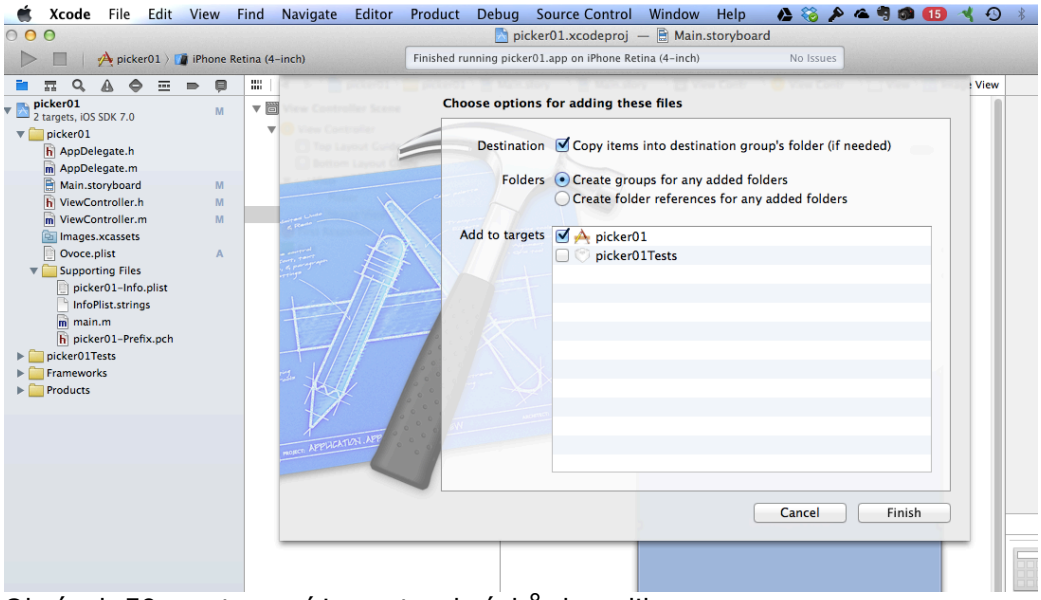
```
//  ViewController.h
#import <UIKit/UIKit.h>

@interface PRIOSViewController : UIViewController
<UIPickerViewDelegate, UIPickerViewDataSource>
{
```

```
    NSArray *images, *texts;
    NSString *selectedItem;
}

@property (weak, nonatomic) IBOutlet UIPickerView
*pickerViewImg;
@property (weak, nonatomic) IBOutlet UIImageView
*imageView;

@end
```

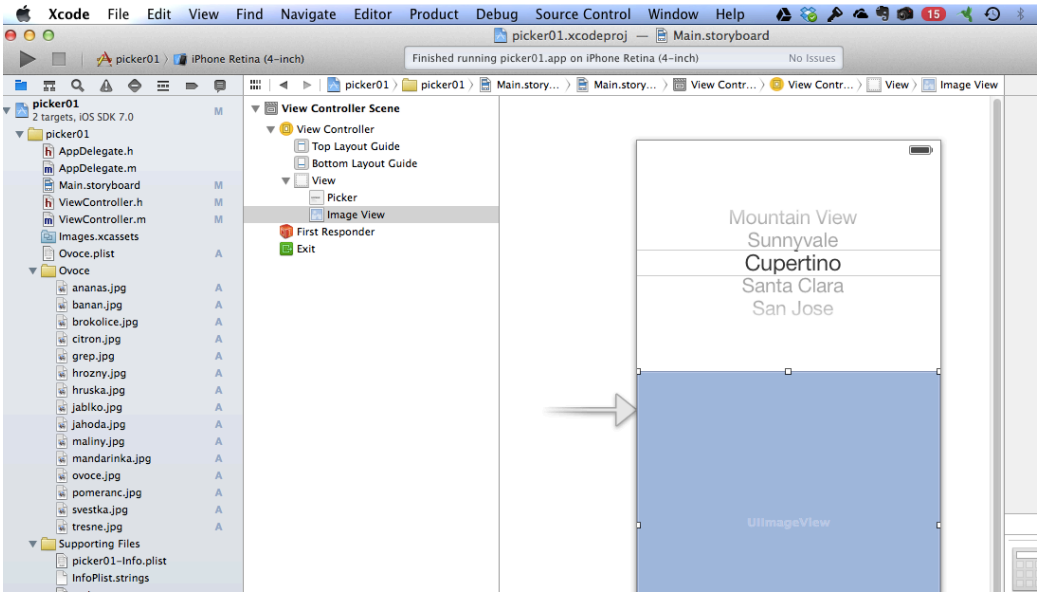
Nejprve do aplikace naimportuje předem připravené obrázky. Z Finderu stačí přetáhnout připravený adresář s obrázky.



Obrázek 58 - import obrázků do aplikace

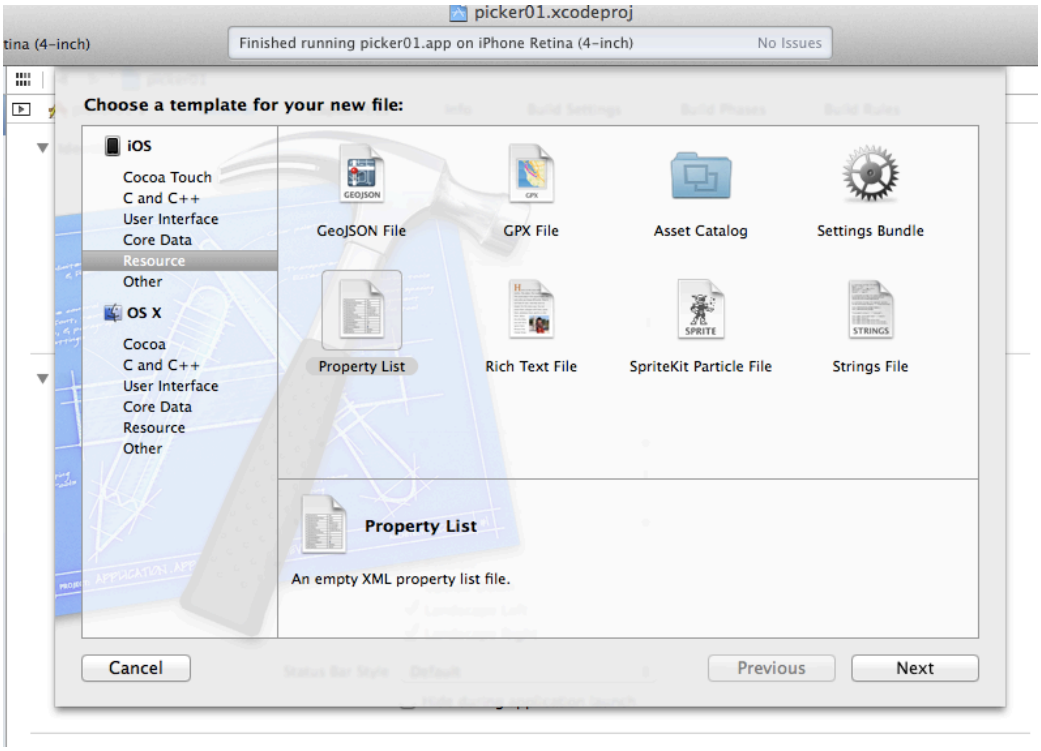


Obrázek 59 - potvrzení importu obrázků do aplikace



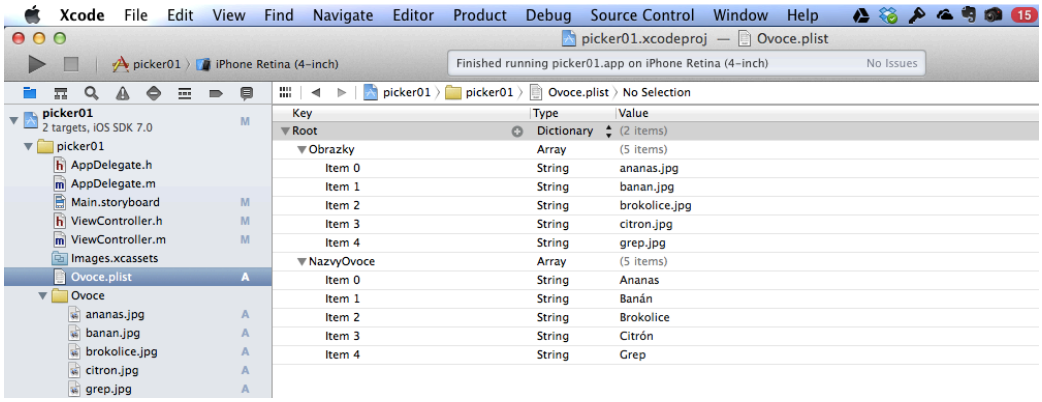
Obrázek 60 - obrázky se stanou součástí aplikace

Pro správu obrázků a údajů o nich si vytvoříme *Property list* přes menu *File – New – File*, volba *Resource – Property list*. Dokument nazvěte *Ovoce.plist*.



Obrázek 61 - vytvoření nového Property listu

Do Property listu si zaznamenáme názvy souboru a jejich popisy. Jedná se o strukturu slovník skládající se ze dvou polí, které obsahují textové řetězce.



Obrázek 62 - obsah Property listu

V souboru ViewController.m je potřeba v metodě (zprávě) viewDidLoad dopsat kód pro načtení informací v Property listu.

```
#import "ViewController.h"
@interface ViewController ()

@end

@implementation ViewController
@synthesize pickerViewImg, imageView;

- (void)viewDidLoad
{
    [super viewDidLoad];
    NSString *path = [[NSBundle mainBundle]
pathForResource:@"Ovoce" ofType:@"plist"];
    NSDictionary *dict = [[NSDictionary alloc]
initWithContentsOfFile:path];

    texts = [dict objectForKey:@"NazvyOvoce"];
    images = [dict objectForKey:@"Obrazky"];

    [pickerViewImg setDelegate:self];
    [pickerViewImg setDataSource:self];

    //inicializace prvniho obrazku
    selectedItem = [images objectAtIndex:0];
    NSString *imageName = [NSString
stringWithFormat:@"%@",selectedItem];
    UIImage *aImage = [UIImage imageNamed:imageName];
    [imageView setImage:aImage];
}
```

Následně je nutné definovat další metody, které zajisti správnou funkci prvku Picker View a které je nutné dodefinovat z důvodu využití delegáta UIPickerViewDelegate a zdroje dat UIPickerViewDataSource.

```
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

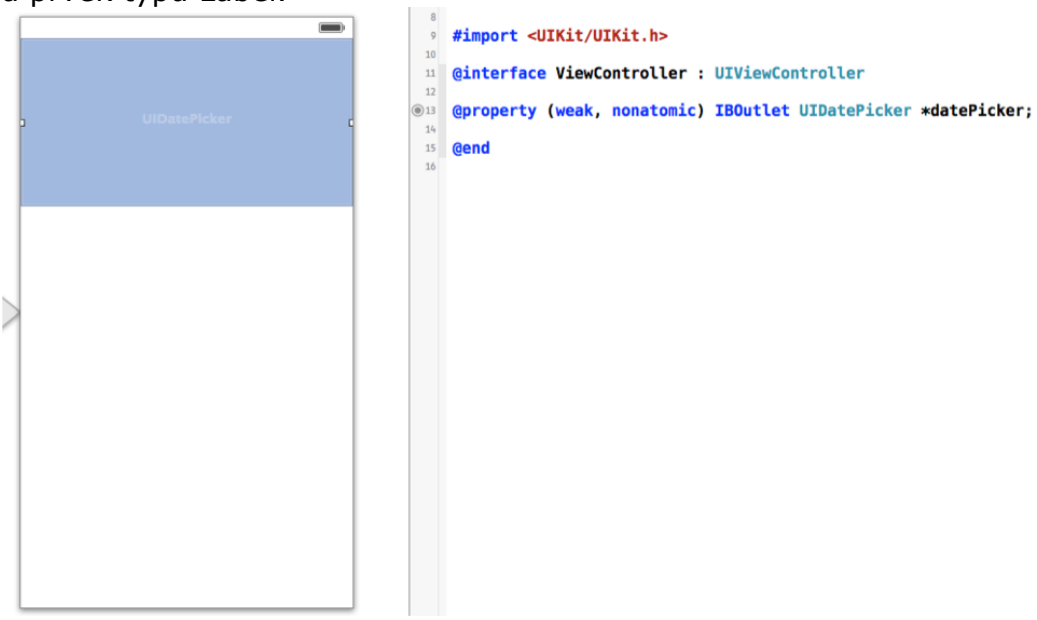
```
-(NSInteger)numberOfComponentsInPickerView:(UIPickerView
*)pickerView
{
    return 1;
}
-(NSInteger)pickerView:(UIPickerView *)pickerView
numberOfRowsInComponent:(NSInteger)component
{
    return [images count];
}
-(NSString *)pickerView:(UIPickerView *)pickerView
titleForRow:(NSInteger)row
forComponent:(NSInteger)component
{
    return [texts objectAtIndex:row];
}
-(void) pickerView:(UIPickerView *)pickerView
didSelectRow:(NSInteger)row
inComponent:(NSInteger)component
{
    selectedItem = [images objectAtIndex:row];
    NSString *imageName = [NSString
stringWithFormat:@"%@",selectedItem];
    UIImage *aImage = [UIImage imageNamed:imageName];
    [imageView setImage:aImage];
}
```

Příklad

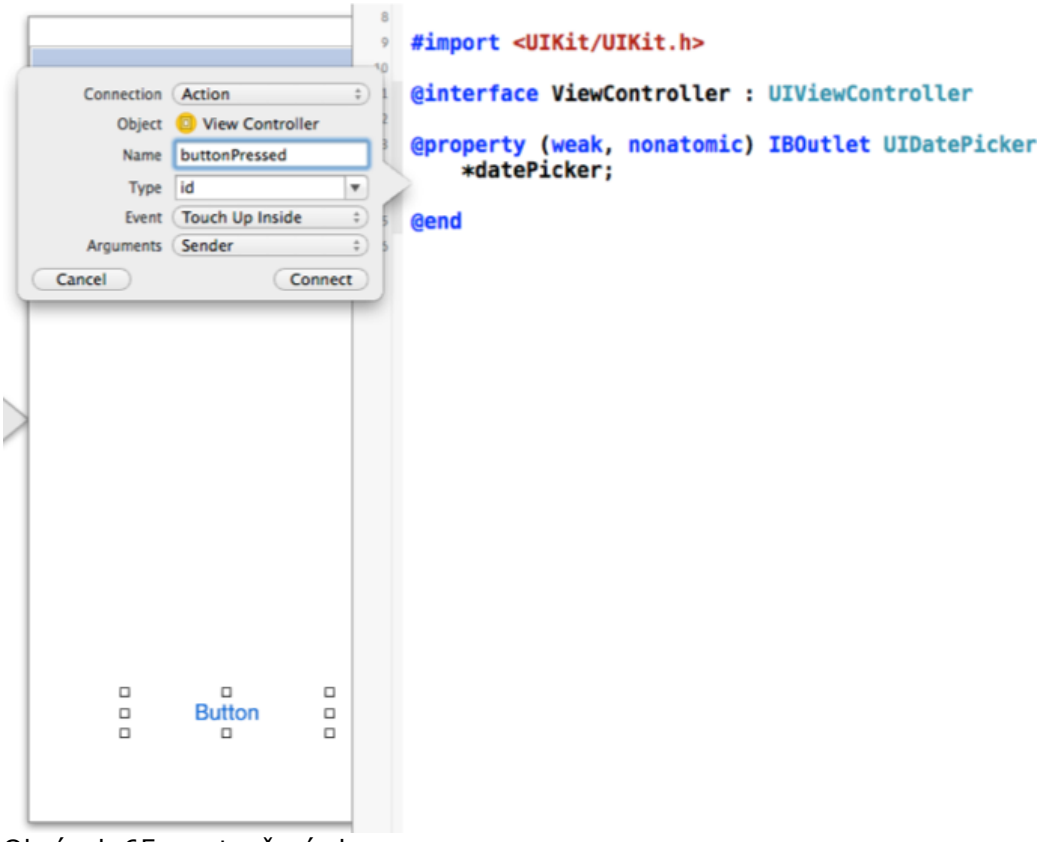
Vytvořte aplikaci, která využije prvek *Data Picker* a bude na ploše aplikace zobrazovat vybraný čas a datum (viz následující obrázek).

Obrázek 63 - vzhled aplikace

Pokuste se příklad vyřešit samostatně s pomocí následujících obrázků a nápovědy. Umístěte do aplikace Data Picker, tlačítko a prvek typu Label.



Obrázek 64 - umístění Data Pickeru do aplikace a vytvoření outletu



Obrázek 65 - vytvoření akce

Upravte kód souboru ViewController.h:

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

@interface ViewController : UIViewController
```

```
@property (weak, nonatomic) IBOutlet UIDatePicker
*datePicker;
@property (weak, nonatomic) IBOutlet UILabel *dateLabel;
- (IBAction)buttonPressed:(id)sender;

@end
```

Definujte jednotlivé zprávy v souboru ViewController.m:

```
//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize datePicker, dateLabel;

- (void)viewDidLoad
{
    [super viewDidLoad];
    NSDate *now =[[NSDate alloc] init];
    [datePicker setDate:now animated:NO];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}

- (IBAction)buttonPressed:(id)sender {
    NSDate *selected = [datePicker date];
    NSString *message = [[NSString alloc]
    initWithFormat:@"%@", selected];
    [dateLabel setText:message];
}
@end
```

Shrnutí kapitoly

Ovládací prvek Picker může být buď ve formě Data Picker pro výběr časových informací nebo obecnější Picker View. Pro správnou funkčnost je potřeba ke třídě připojit protokoly UIPickerViewDelegate a UIPickerViewDataSource.

Jednotlivá data nezapisujeme přímo do zdrojových kódů, ale do Property listu, ve kterém můžeme dopsat jednotlivé informace.

7. SCÉNY APLIKACE

V této kapitole se dozvíte:

Cílem této lekce seznámit se s možnostmi vytvářet v aplikaci více scén.

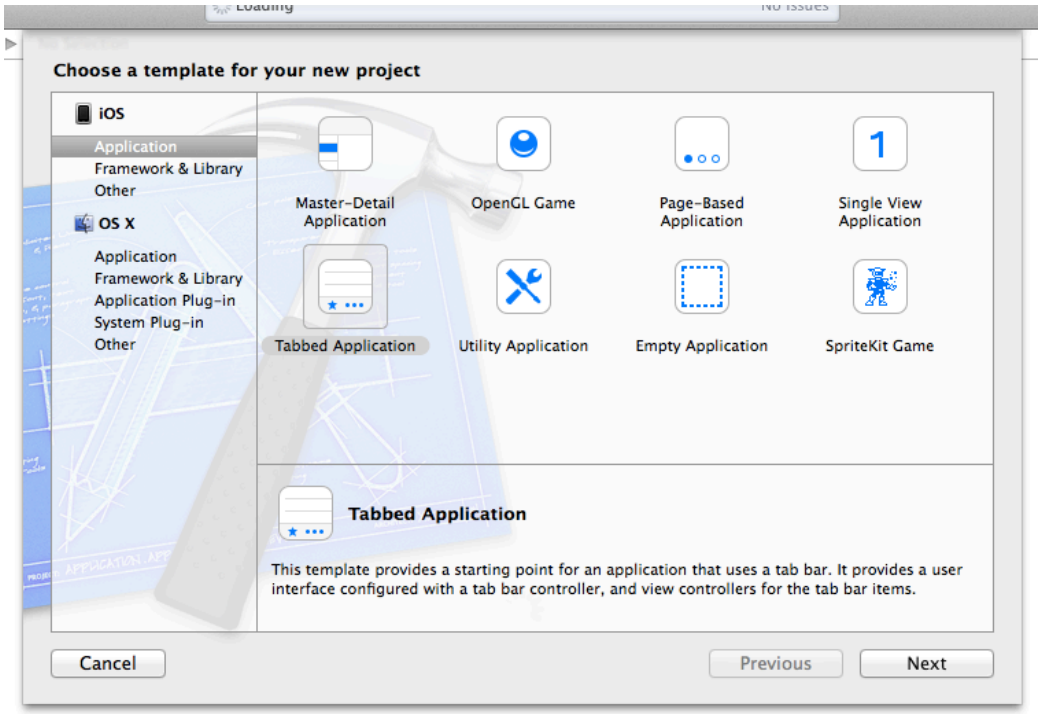
Po absolvování lekce budete:

- umět používat více scén v aplikaci
- umět využít šablonu projektu Tabbed Application

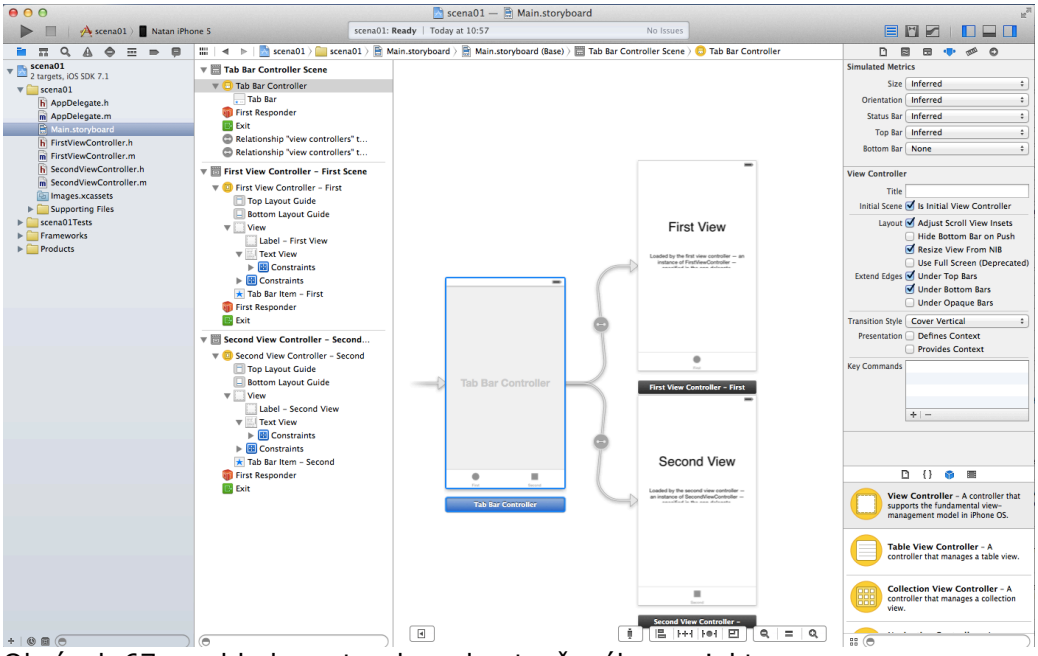
Klíčová slova této kapitoly:
scéna, Tabbed Application

Čas ke studiu: 3 hodiny

Následující příklad ukazuje užití více scén. K vytvoření projektu využijeme šablonu Tabbed Application. Díky šabloně nebude nutné napsat žádný kód.

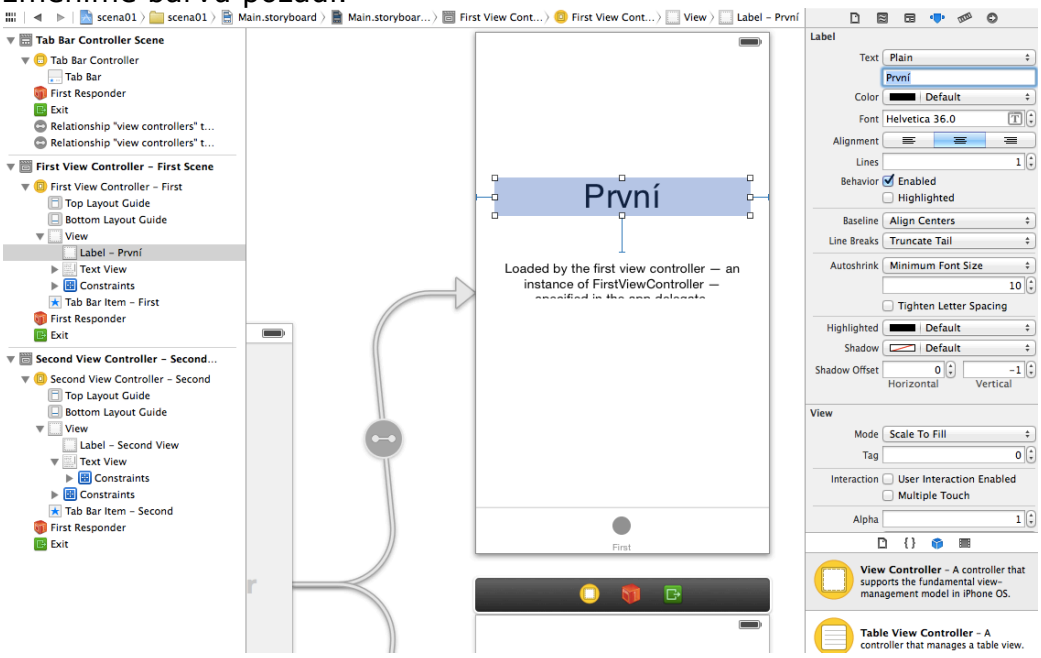


Obrázek 66 - vytvoříme nový projekt na základě šablony Tabbed Application

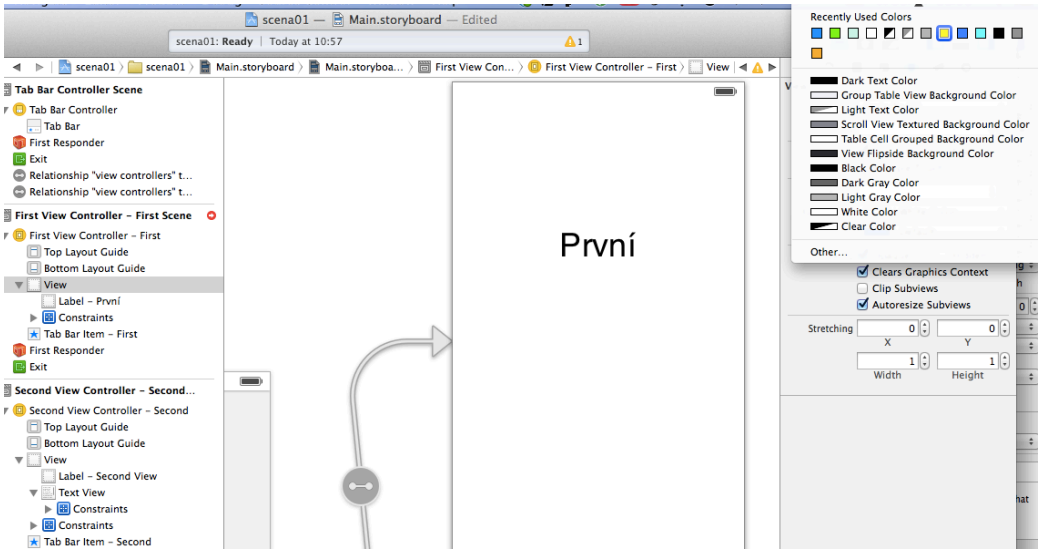


Obrázek 67 - pohled na storyboard vytvořeného projektu

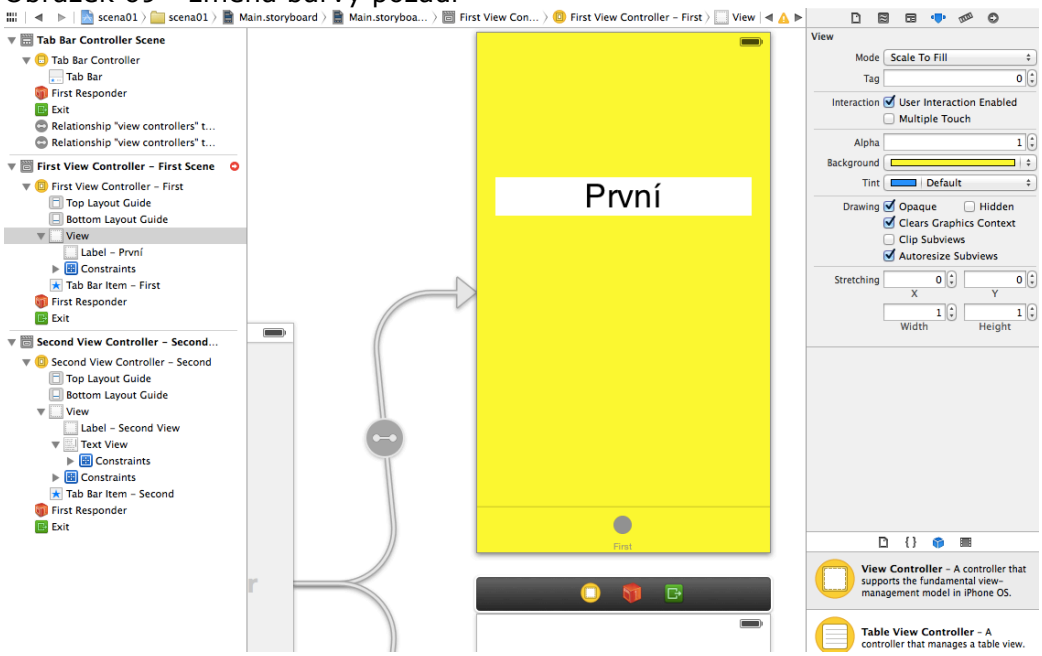
Nejprve změníme vzhled jednotlivých scén aplikace. Smažeme přebytečné komponenty (TextView), přepíšeme texty (Label) na jednotlivých pohledech – scénách aplikace a pro názornost si změníme barvu pozadí.



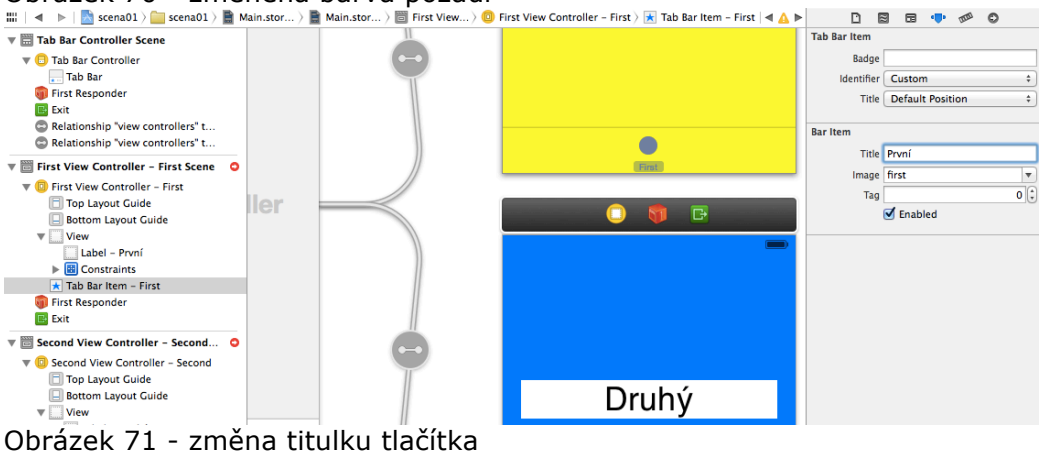
Obrázek 68 - přepis textu na prvním pohledu



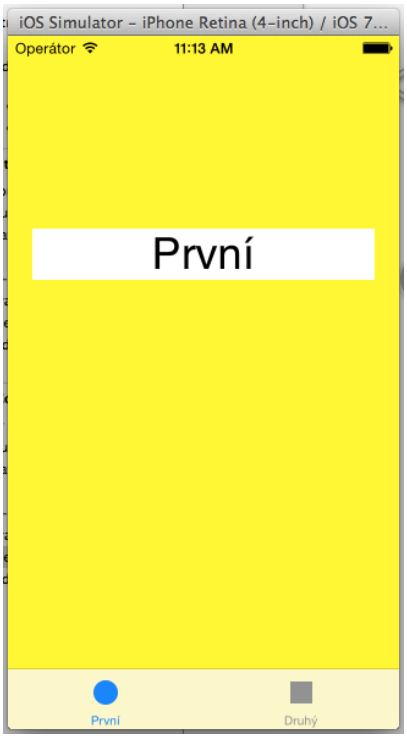
Obrázek 69 - změna barvy pozadí



Obrázek 70 - změněná barva pozadí



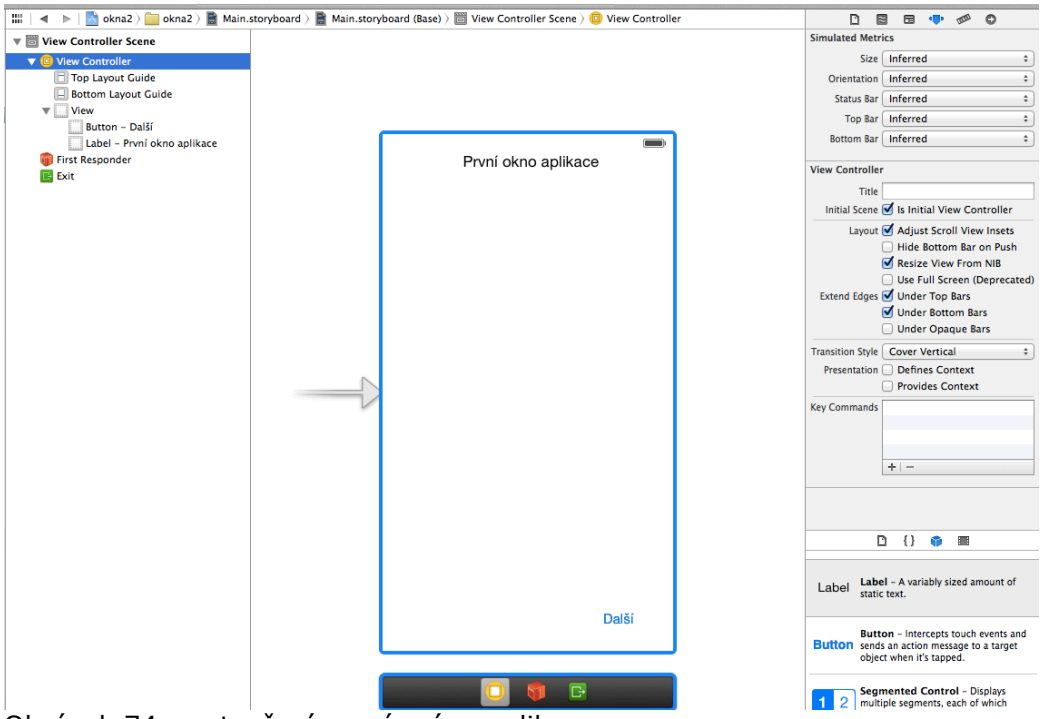
Obrázek 71 - změna titulku tlačítka



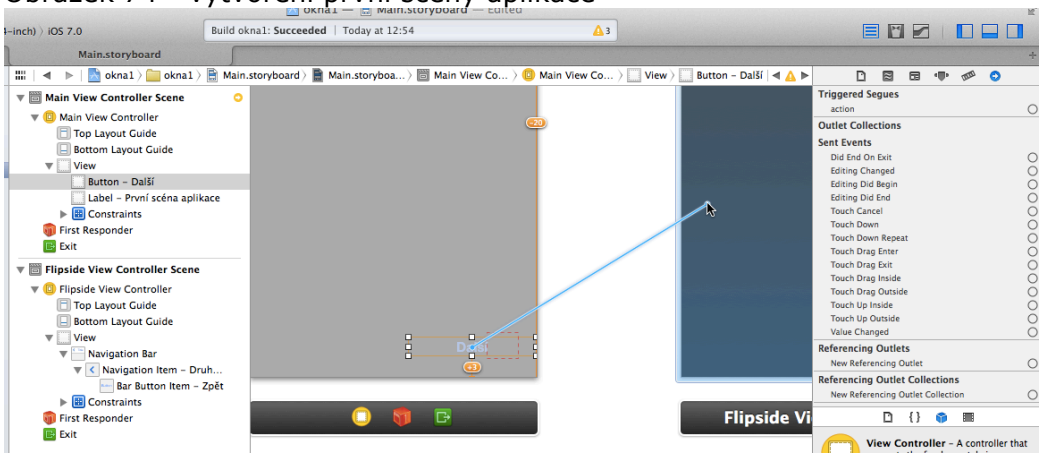
Obrázek 72 - pohled na první scénu spuštěné aplikace

Obrázek 73 - přepnutí aplikace do druhé scény

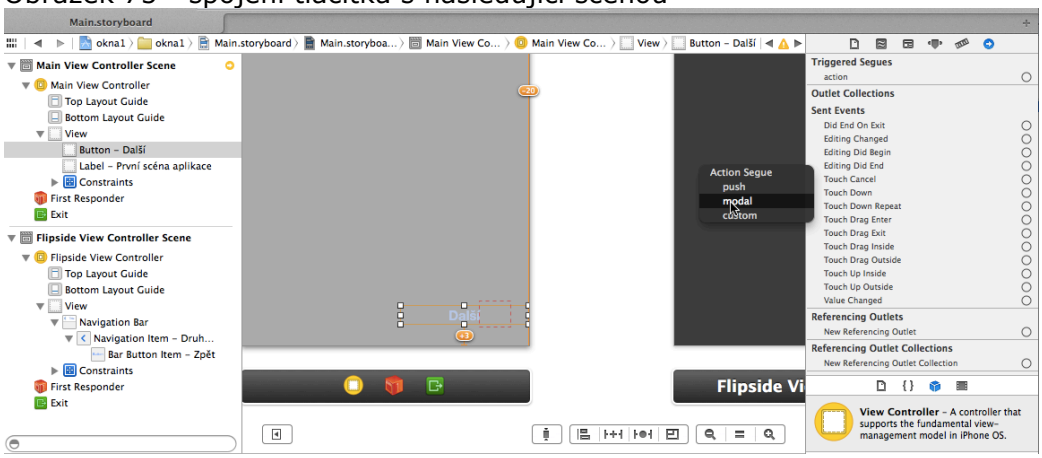
V následujícím ukázkovém příkladu si prohlédněte další způsob vytváření více scén v mobilní aplikaci. Vytvoříme si projekt na základě šablony Single View Application a k prvnímu ViewControlleru vytvoříme další scénu. Přepínání scén se bude dít pomocí tlačítek, které umístíme do scén.



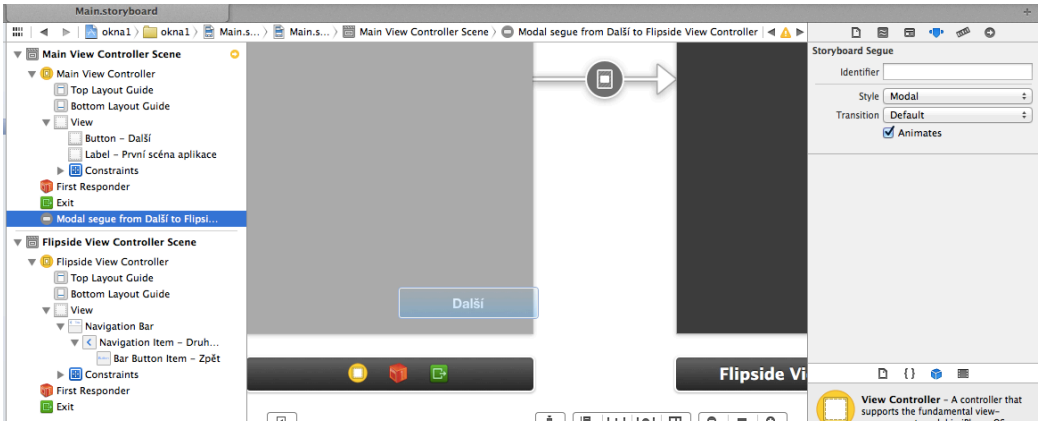
Obrázek 74 - vytvoření první scény aplikace



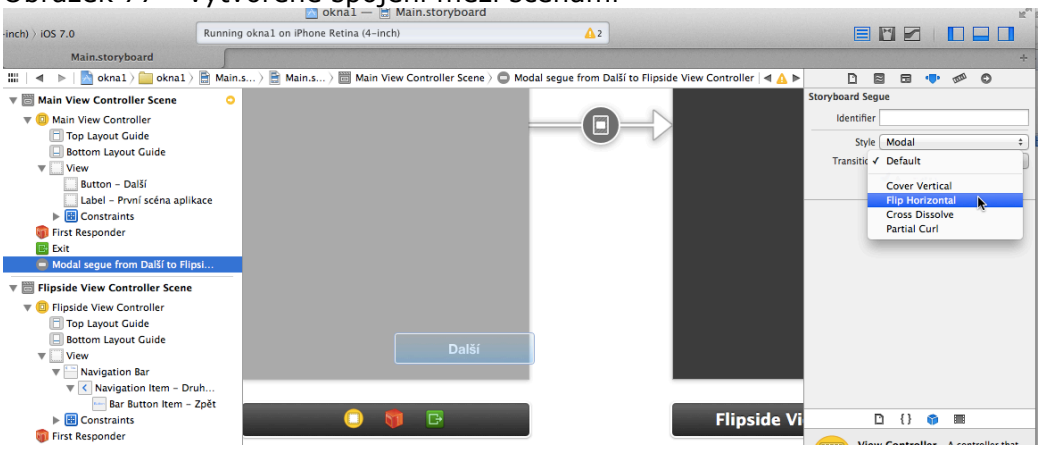
Obrázek 75 - spojení tlačítka s následující scénou



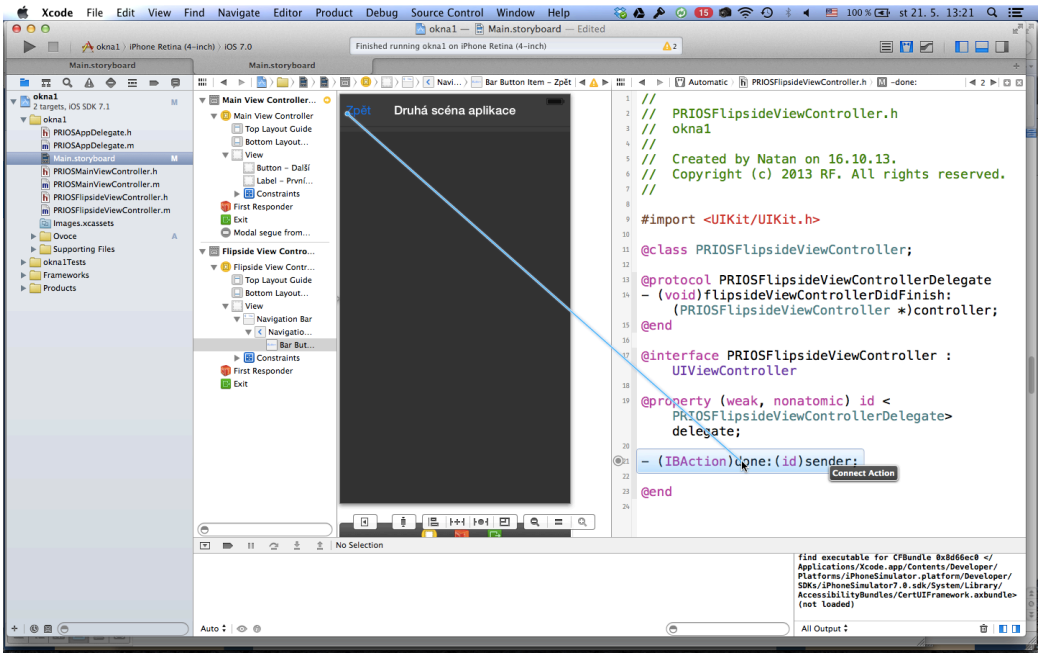
Obrázek 76 - výběr Action Seque



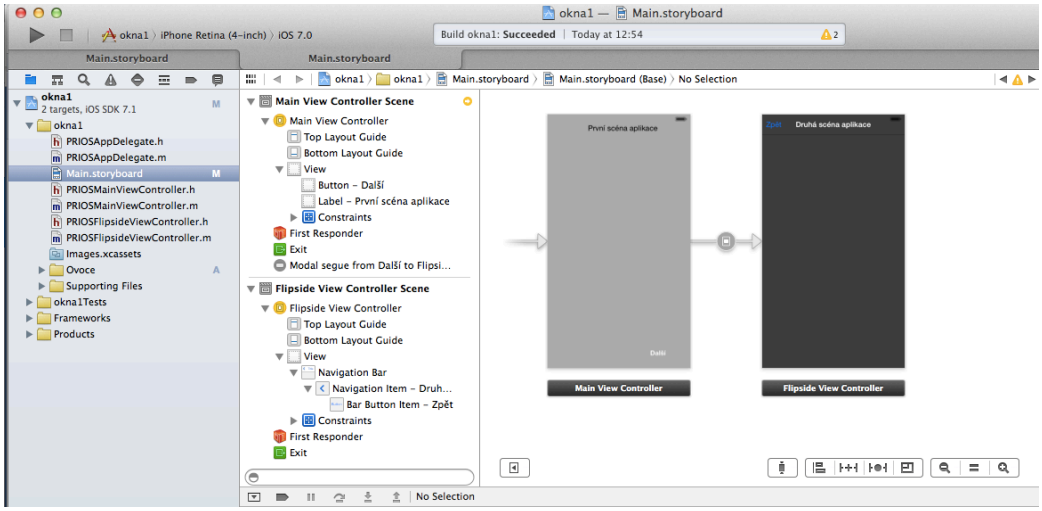
Obrázek 77 - vytvořené spojení mezi scénami



Obrázek 78 - nastavení animace přechodu mezi scénami



Obrázek 79 - vytvoření akce pro tlačítko Zpět



Obrázek 80 - výsledný návrh aplikace

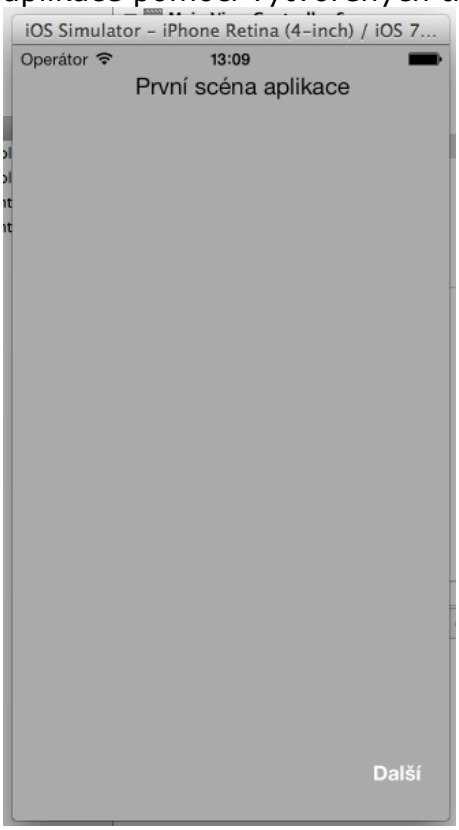
Prohledněte si zdrojové kódy aplikace:

```
//  PRIOSMainViewController.h
#import "PRIOSFlipsideViewController.h"

@interface PRIOSMainViewController : UIViewController
<PRIOSFlipsideViewControllerDelegate>

@end


//  PRIOSMainViewController.m
#import "PRIOSMainViewController.h"

@interface PRIOSMainViewController ()

@end

@implementation PRIOSMainViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
        // Do any additional setup after loading the view,
typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark - Flipside View

-
(void)flipsideViewControllerDidFinish:(PRIOSFlipsideViewCo
ntroller *)controller
{
```

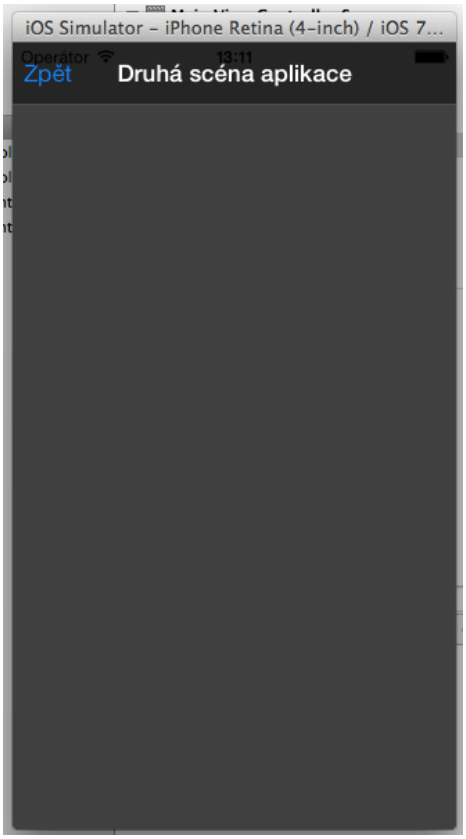
```
    [self dismissViewControllerAnimated:YES
completion:nil];
}

- (void)prepareForSegue:(UIStoryboardSegue *)segue
sender:(id)sender
{
    if ([[segue identifier]
isEqualToString:@"showAlternate"]) {
        [[segue destinationViewController]
setDelegate:self];
    }
}

@end


//  PRIOSFlipsideViewController.h
#import <UIKit/UIKit.h>

@class PRIOSFlipsideViewController;

@protocol PRIOSFlipsideViewControllerDelegate
- (void)flipsideViewControllerDidFinish:
(PRIOSFlipsideViewController *)controller;
@end

@interface PRIOSFlipsideViewController : UIViewController

@property (weak, nonatomic) id
<PRIOSFlipsideViewControllerDelegate> delegate;

- (IBAction)done:(id)sender;

@end

//  PRIOSFlipsideViewController.m
#import "PRIOSFlipsideViewController.h"

@interface PRIOSFlipsideViewController ()

@end

@implementation PRIOSFlipsideViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
```

```
}

#pragma mark - Actions

- (IBAction)done:(id)sender
{
    [self.delegate flipsideViewControllerDidFinish:self];
}

@end
```

Následující dva obrázky ukazují přepínání mezi scénami (okny) aplikace pomocí vytvořených tlačítek.



Obrázek 81 - první scéna (okno) spuštěné aplikace



Obrázek 82 - druhá scéna spuštěné aplikace

8. URČOVÁNÍ POLOHY A MAPY

V této kapitole se dozvíte:

Cílem této lekce je seznámit se možnostmi zjištění umístění přístroje a využití mapových podkladů ve vytvářených aplikací.

Po absolvování lekce budete:

- umět pracovat s frameworkem MapKit
- umět pracovat s frameworkem Core Location

Klíčová slova této kapitoly:
framework Core Location, mapy, MapKit, MKMapView

Čas ke studiu: 3 hodiny

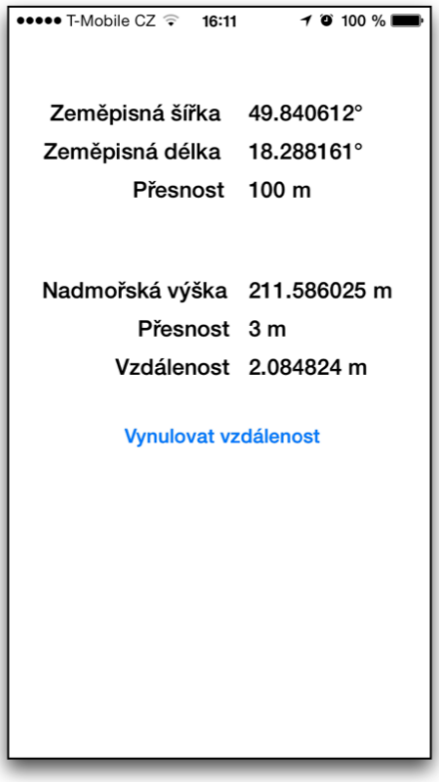
Přístroje iPhone a iPad mohou využívat služby pro lokaci umístění. Při tvorbě aplikací je možné využít frameworku Core Location. Umístění přístroje lze zjistit na základě tří základních služeb:

- GPS
- vysílače mobilních sítí
- WiFi

Přistroj si sám vybere tu nejpřesnější dostupnou službu.

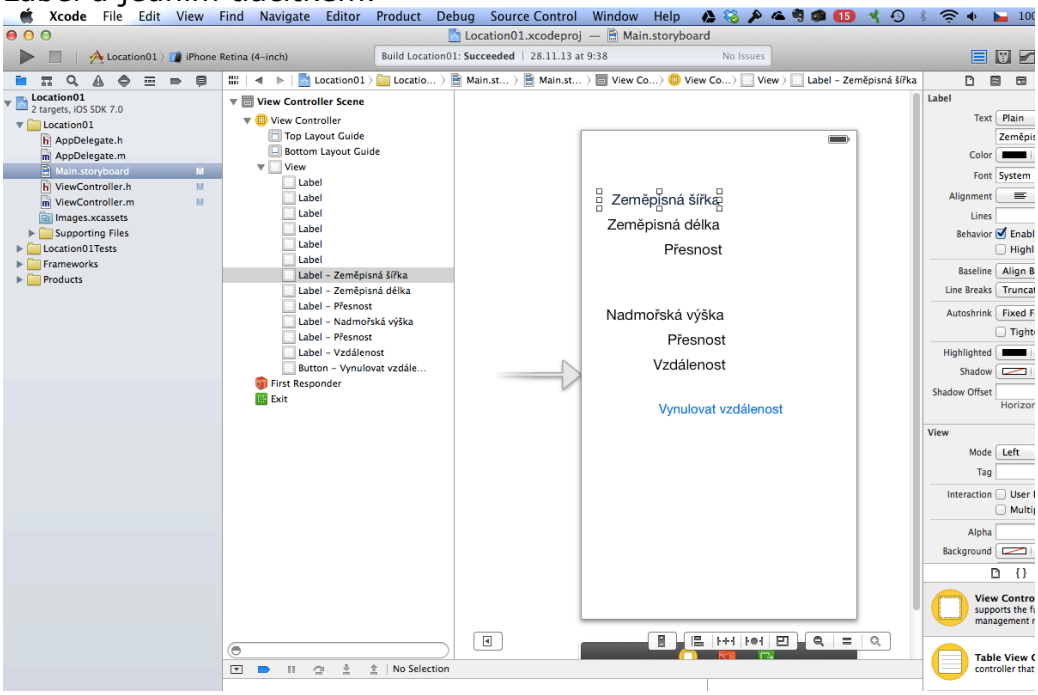
Příklad

Aplikace bude ukazovat základní zeměpisné údaje a vzdálenost, kterou uživatel s přístrojem urazil.



Obrázek 83 - spuštěná aplikace

Nejprve vytvořte vzhled aplikace. Vystačíme s prvky typu Label a jedním tlačítkem.



Obrázek 84 – návrh vzhledu aplikace

Vytvořme outlety a akci, importujme potřebný Framework a doplňme protokol v souboru ViewController.h:

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

@interface ViewController : UIViewController
<CLLocationManagerDelegate>

@property (weak, nonatomic) IBOutlet UILabel
*latitudeLabel;
@property (weak, nonatomic) IBOutlet UILabel
*longitudeLabel;
@property (weak, nonatomic) IBOutlet UILabel
*horizontalAccuracyLabel;
@property (weak, nonatomic) IBOutlet UILabel
*altitudeLabel;
@property (weak, nonatomic) IBOutlet UILabel
*verticalAccuracyLabel;
@property (weak, nonatomic) IBOutlet UILabel
*distanceLabel;

@property (retain,nonatomic) CLLocationManager
*locationManager;
@property (retain,nonatomic) CLLocation *startingPoint;

- (IBAction)resetDistance:(id)sender;
@end
```

Nadefinujme akci v souboru ViewController.m:

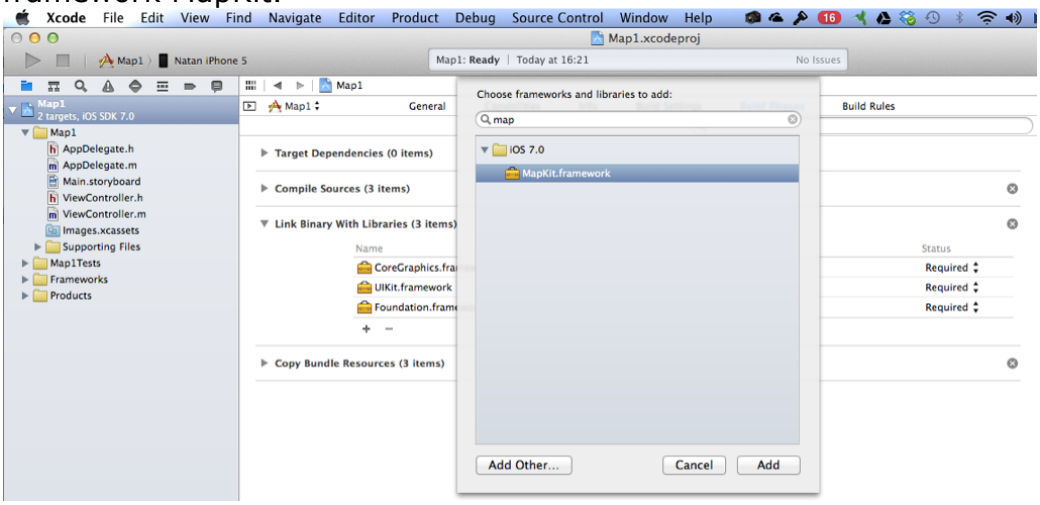
```
//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

@synthesize horizontalAccuracyLabel, latitudeLabel,
longitudeLabel, altitudeLabel, verticalAccuracyLabel,
distanceLabel;
@synthesize locationManager, startingPoint;

- (void)viewDidLoad
{
    [super viewDidLoad];
     locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate = self;
    locationManager.desiredAccuracy =
kCLLocationAccuracyBest;
    [locationManager startUpdatingLocation];
    startingPoint = nil;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}

- (IBAction)resetDistance:(id)sender
{
    startingPoint = nil;
}

#pragma mark -
#pragma mark CLLocationManagerDelegate

- (void) locationManager:(CLLocationManager *)manager
didUpdateToLocation:(CLLocation *)newLocation
fromLocation:(CLLocation *)oldLocation
{
    if (startingPoint == nil) startingPoint = newLocation;

    NSString *latitudeStr = [[NSString alloc]
initWithFormat:@"%lf°", newLocation.coordinate.latitude];
    latitudeLabel.text = latitudeStr;
    NSString *longitudeStr = [[NSString alloc]
initWithFormat:@"%lf°", newLocation.coordinate.longitude];
    longitudeLabel.text = longitudeStr;
    NSString *horizonatalAccuracyStr = [[NSString alloc]
initWithFormat:@"%.0lf m",
newLocation.horizontalAccuracy];
    horizontalAccuracyLabel.text = horizonatalAccuracyStr;
```

```
    NSString *altitudeStr = [[NSString alloc]
initWithFormat:@"%lf m", newLocation.altitude];
    altitudeLabel.text = altitudeStr;
    NSString *verticalAccuracyStr = [[NSString alloc]
initWithFormat:@"%.0lf m", newLocation.verticalAccuracy];
    verticalAccuracyLabel.text = verticalAccuracyStr;

    CLLocationDistance distance = [newLocation
getDistanceFrom:startingPoint];
    NSString *distanceStr = [[NSString alloc]
initWithFormat:@"%lf m", distance];
    distanceLabel.text = distanceStr;

}

@end
```

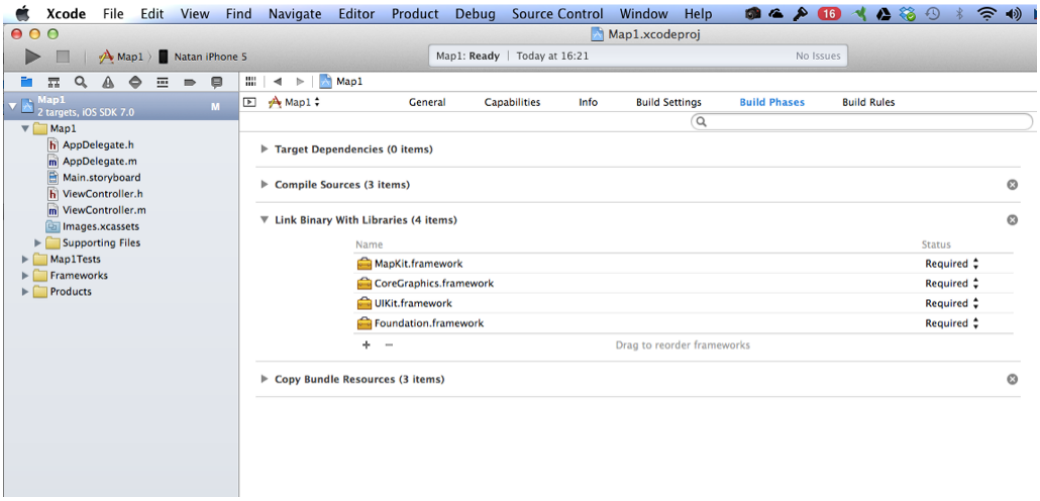
Příklad

Vytvořme aplikaci, která bude zobrazovat systémové mapy.

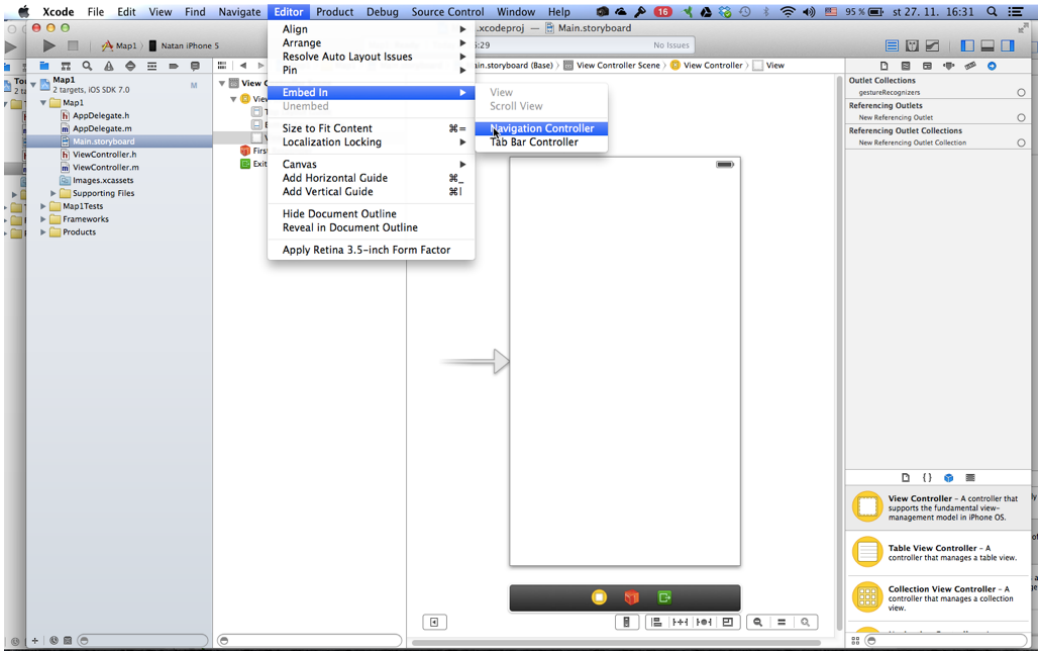
Nejprve vytvořme nový projekt a připojme k němu příslušný framework MapKit.



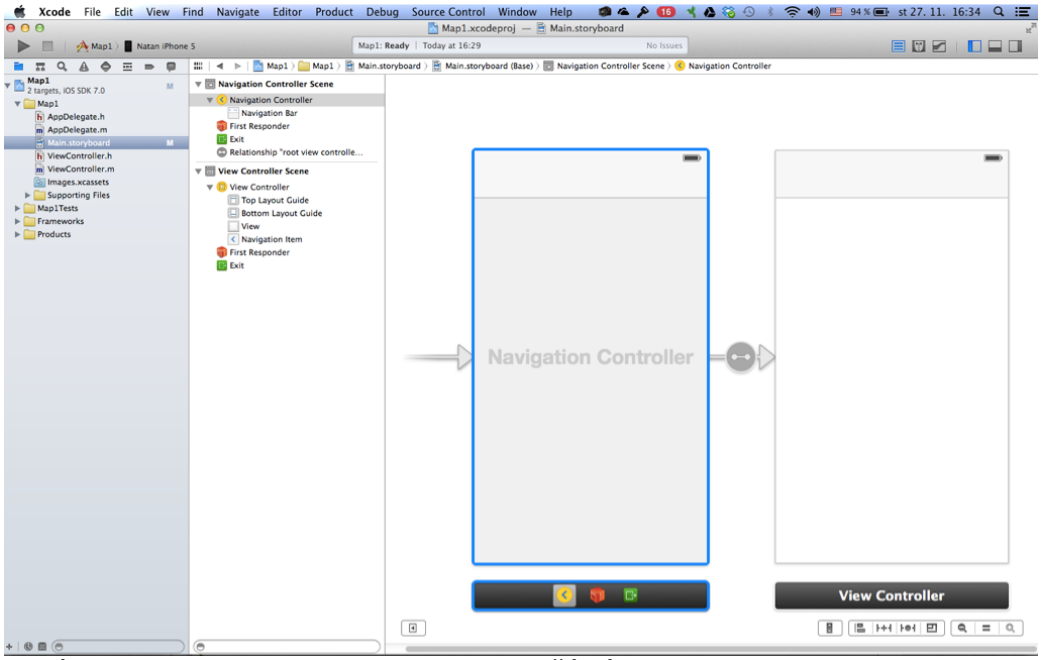
Obrázek 85 - připojení frameworku MapKit



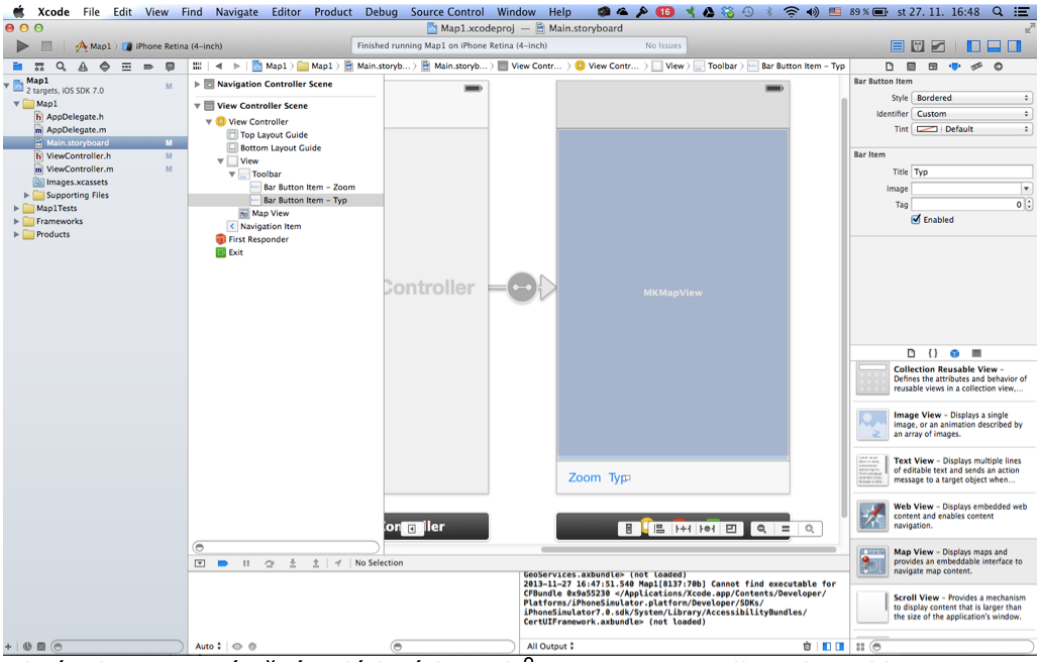
Obrázek 86 - framework MapKit je připojen k projektu



Obrázek 87 - připojení Navigator Controlleru



Obrázek 88 - Navigator Controller je součástí aplikace



Obrázek 89 - umístění ovládacích prvků Map View Toolbar do aplikace

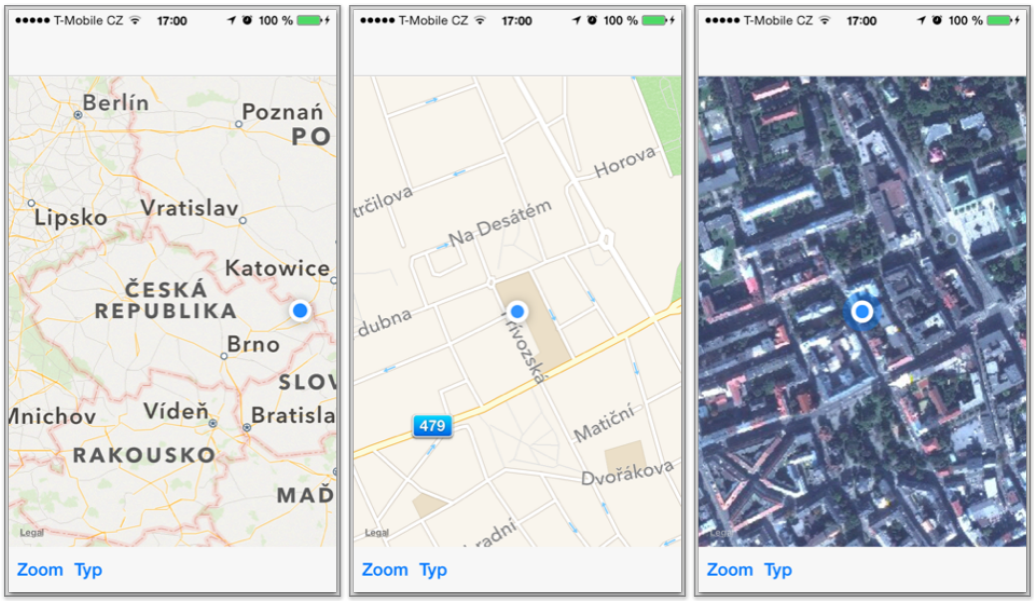
Soubor ViewController.h:

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

@interface ViewController : UIViewController
<MKMapViewDelegate>

@property (weak, nonatomic) IBOutlet MKMapView *mapView;

- (IBAction)zoomIn:(id)sender;
- (IBAction)changeMapType:(id)sender;

@end
```

Soubor ViewController.m:

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize mapView;

- (void)viewDidLoad
{
    [super viewDidLoad];
    mapView.showsUserLocation = YES;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}
```

```
-(void) mapView:(MKMapView *)mapView
didUpdateUserLocation:(MKUserLocation *)userLocation
{
    mapView.centerCoordinate =
userLocation.location.coordinate;
}
- (IBAction)zoomIn:(id)sender
{
    MKUserLocation *userLocation = mapView.userLocation;
    MKCoordinateRegion region =
MKCoordinateRegionMakeWithDistance(userLocation.location.c
oordinate, 500, 500);
    [mapView setRegion:region animated:NO];
}

- (IBAction)changeMapType:(id)sender
{
    if (mapView.mapType == MKMapTypeStandard)
        mapView.mapType = MKMapTypeSatellite;
    else
        mapView.mapType = MKMapTypeStandard;
}
@end
```



Obrázek 90 - spuštěná aplikace

Shrnutí kapitoly

Pro práci s určováním polohy využívá zařízení GPS, WiFi nebo GSM sítě. Pro vývoj můžeme používat frameworky CoreLocation a MapKit.

9. DALŠÍ OVLÁDACÍ PRVKY

V této kapitole se dozvíte:

Cílem této lekce seznámit se s dalšími ovládacími prvky, které nabízí Framework Cocoa Touch.

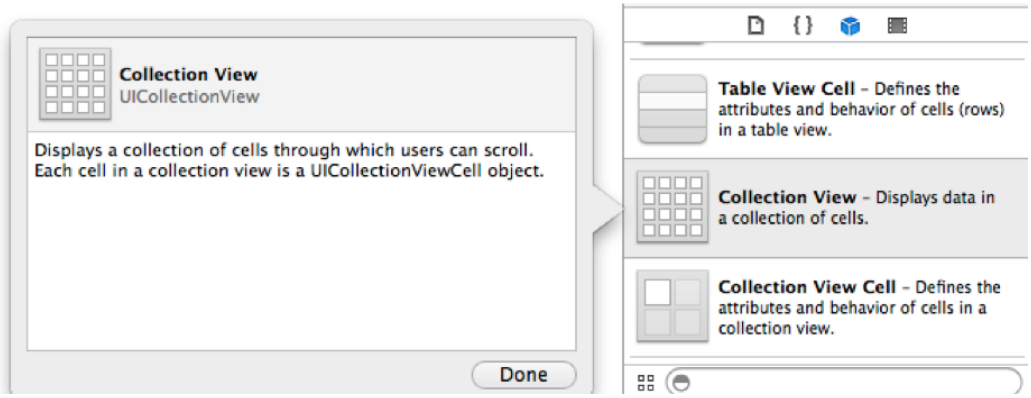
Po absolvování lekce budete:

- umět pracovat s Collection View
- umět vytvářet jednoduché aplikace

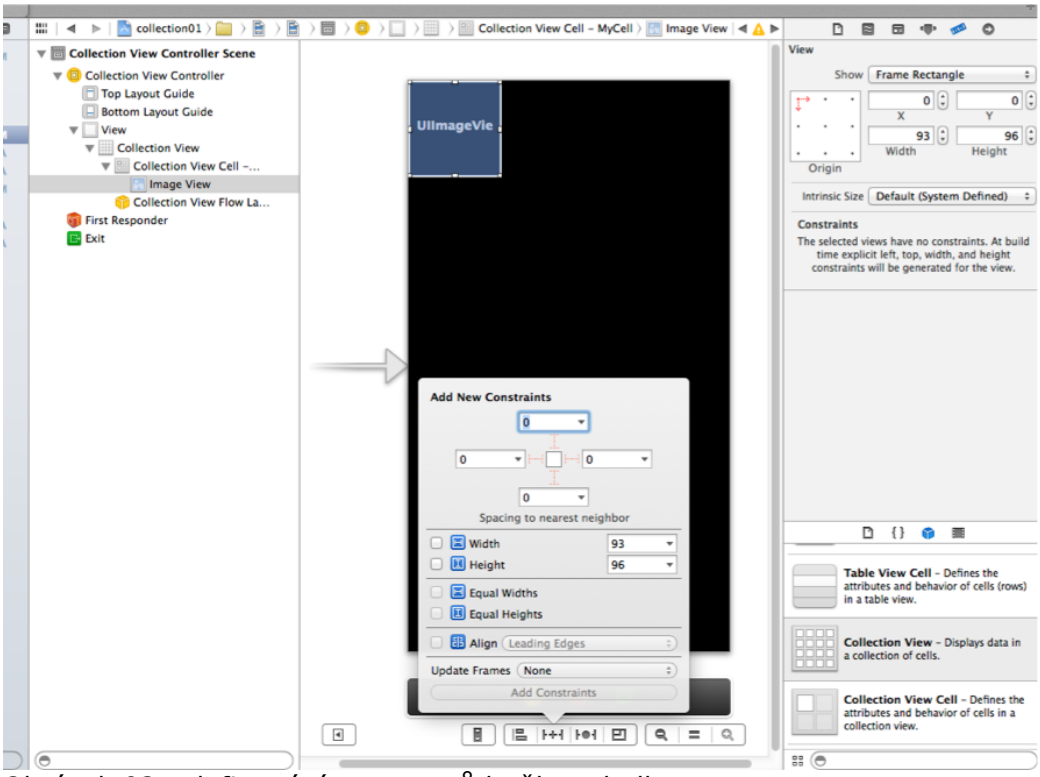
Klíčová slova této kapitoly:
Collection View

Čas ke studiu: 3 hodiny

Collection View je prvek, který ve formě tabulky umožňuje znázornit informace v aplikaci.

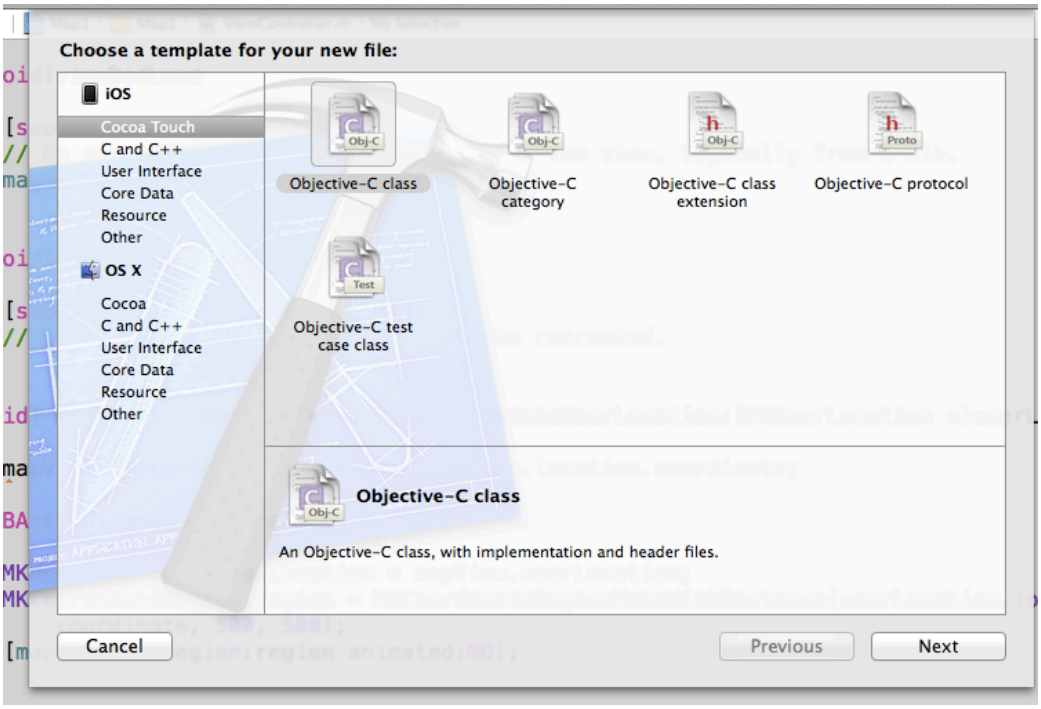


Obrázek 91 - Collection View

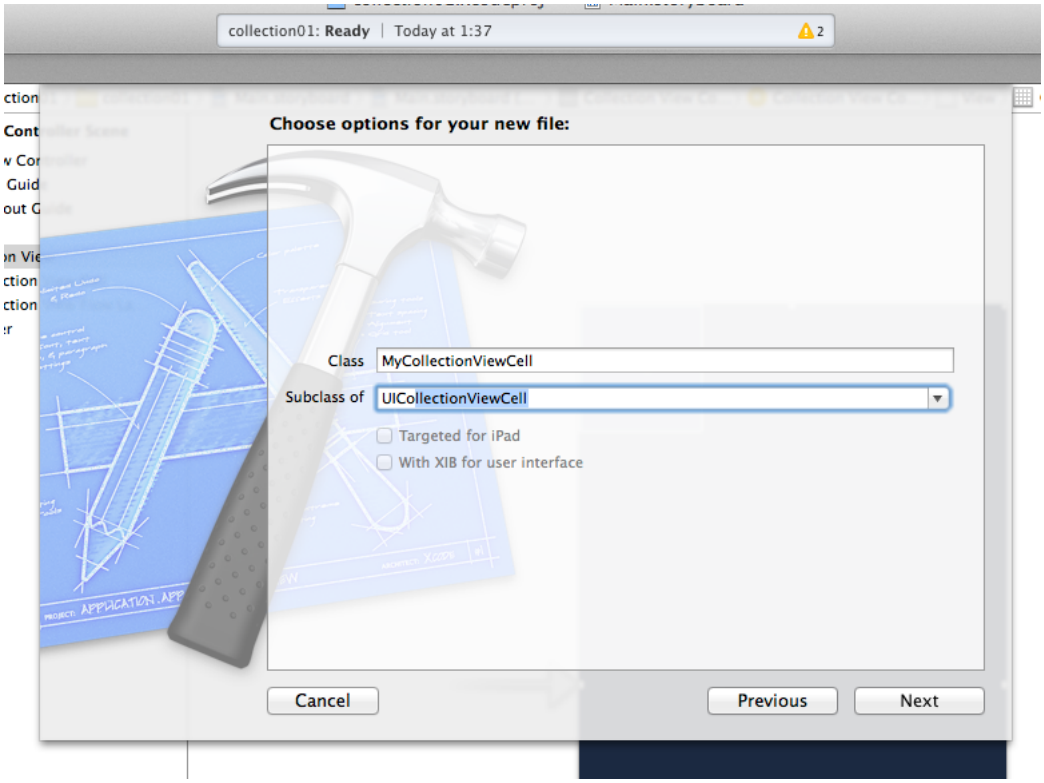


Obrázek 92 - definování parametrů buňky tabulky

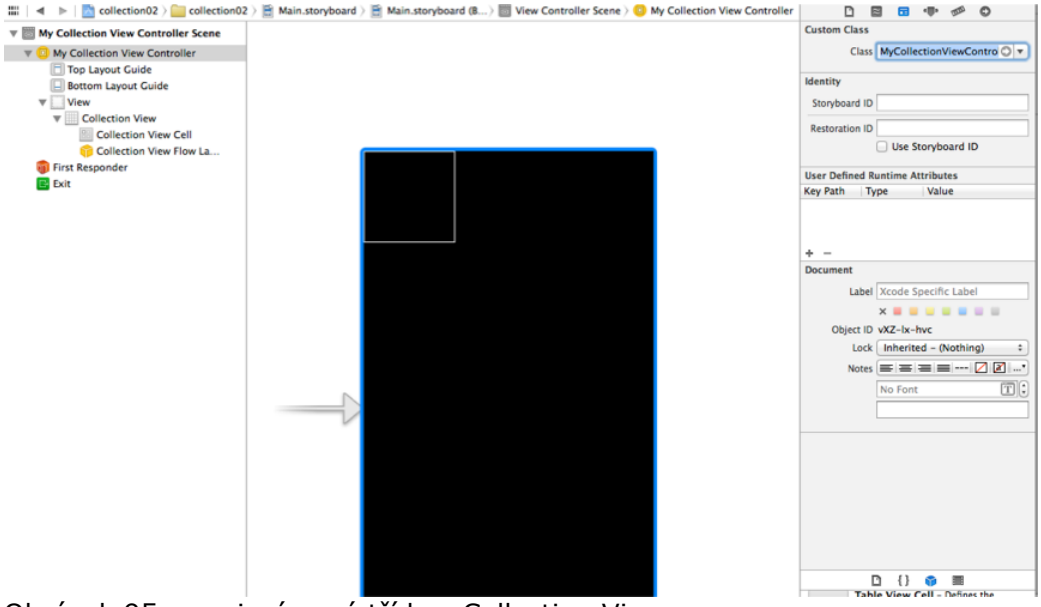
V menu *File – New – File* v části *Cocoa Touch* vyberte *Objective-C class*, pomocí které vytvoříme novou třídu pro Collection View.



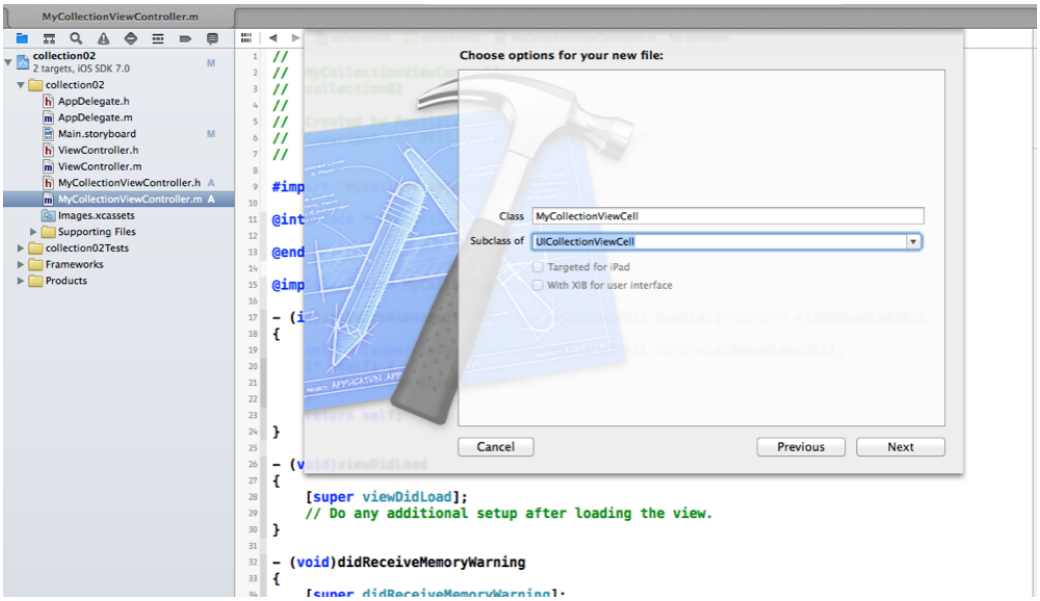
Obrázek 93 - vytvoření nové třídy



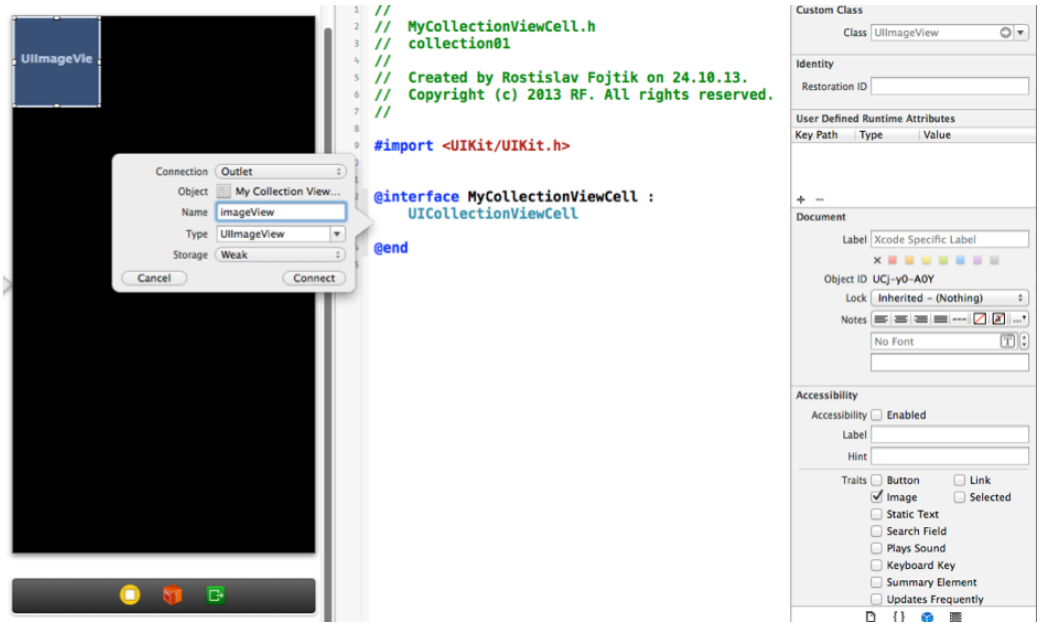
Obrázek 94 - určení jména a rodiče nové třídy



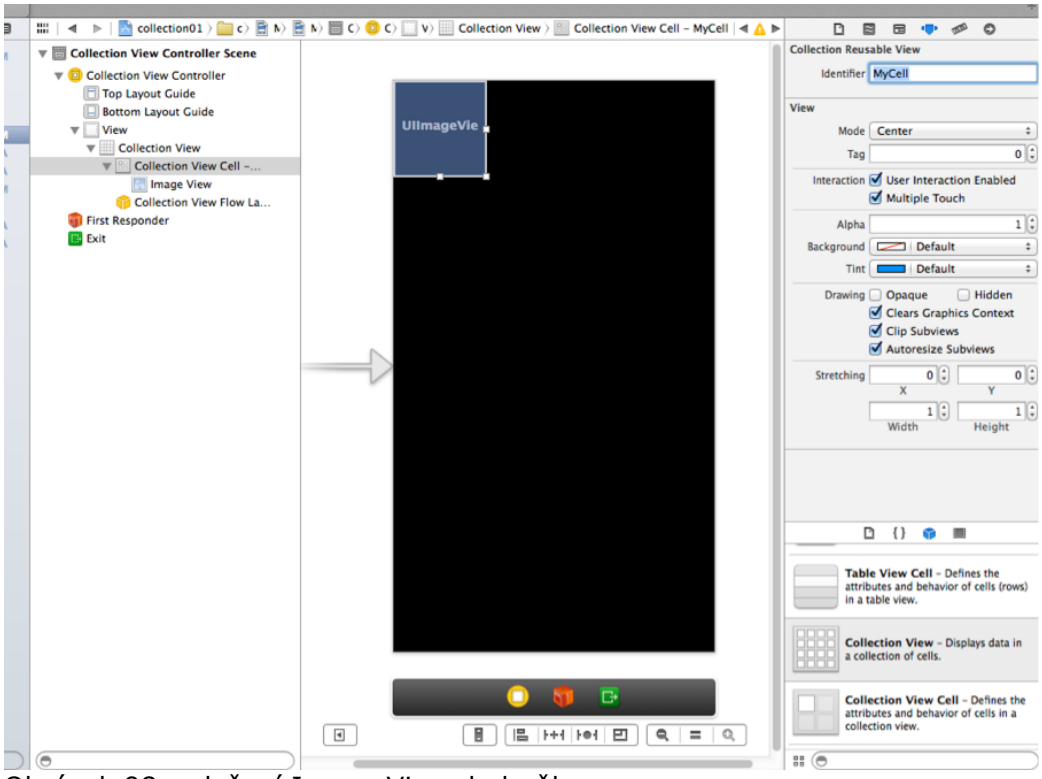
Obrázek 95 - spojení nové třídy s Collection View



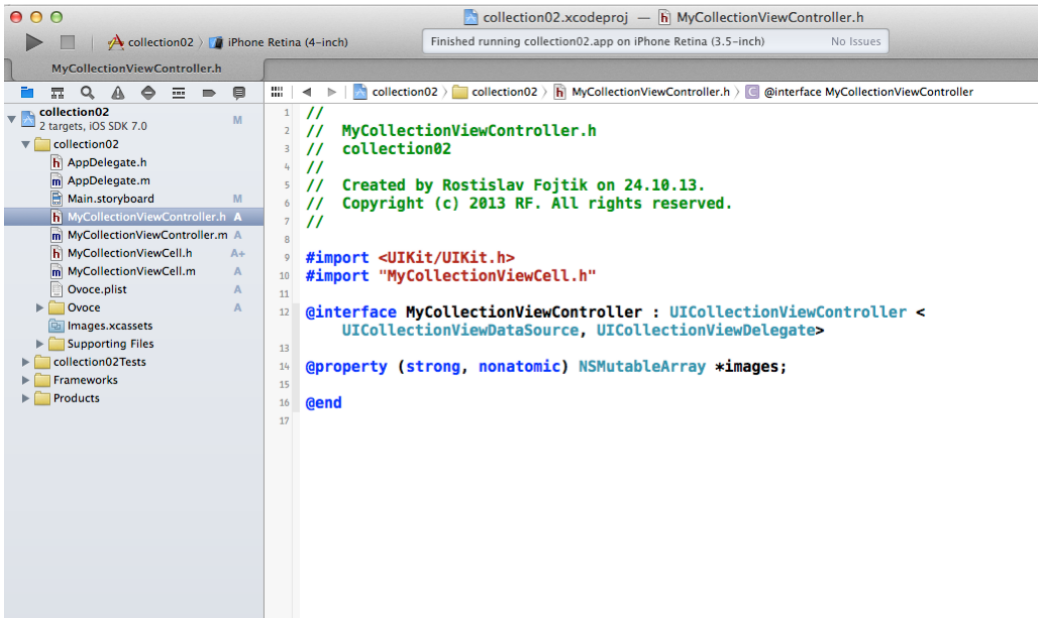
Obrázek 96 - vytvoření třídy pro buňky tabulky



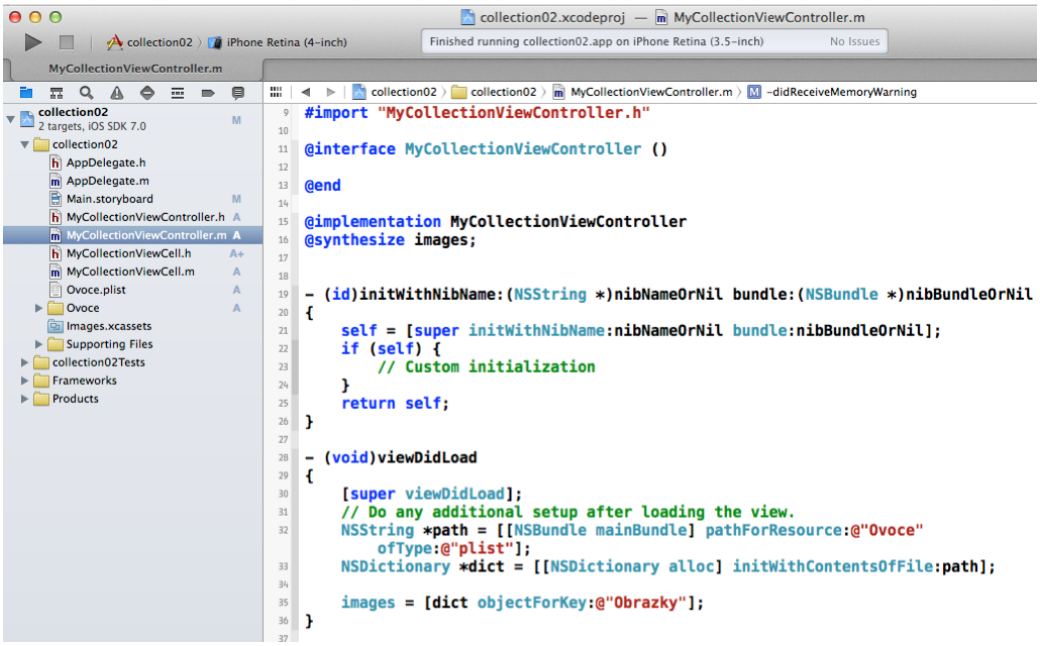
Obrázek 97 - vytvoření outlet



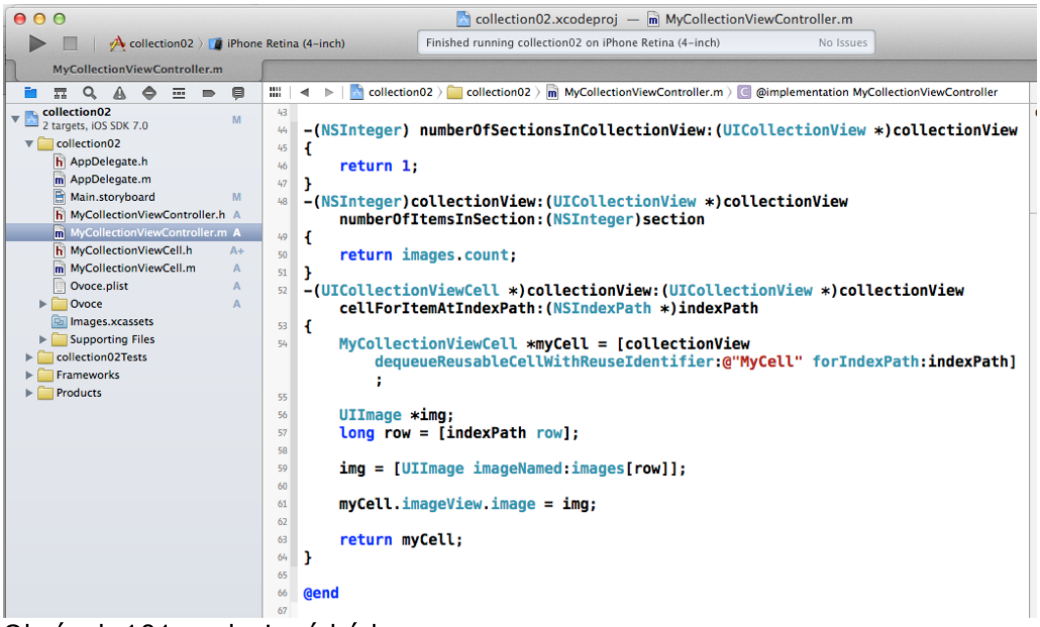
Obrázek 98 - vložení Image View do buňky



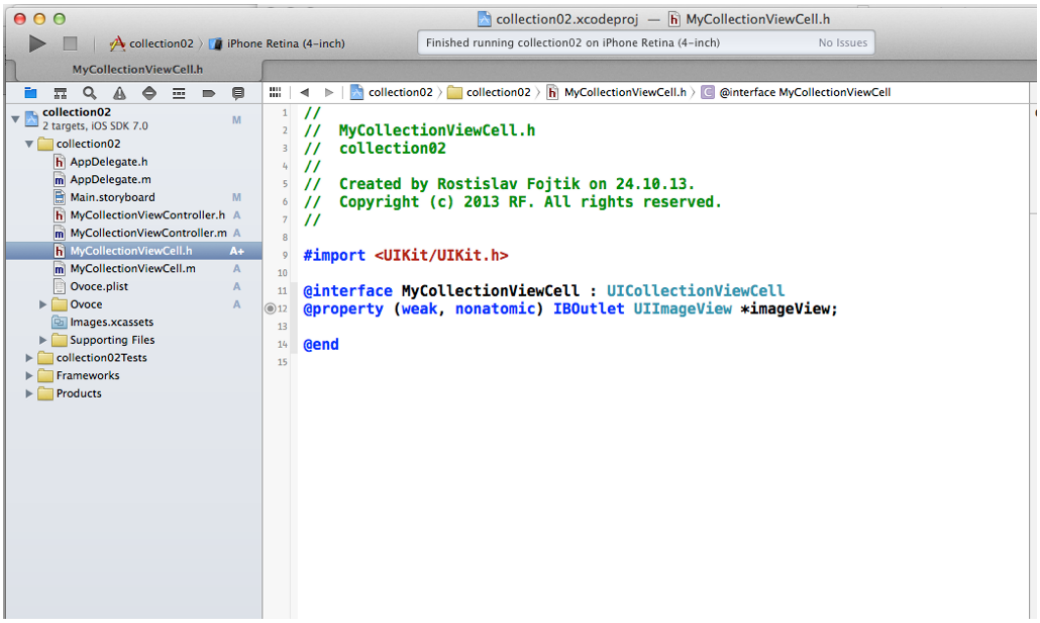
Obrázek 99 - zdrojové kódy



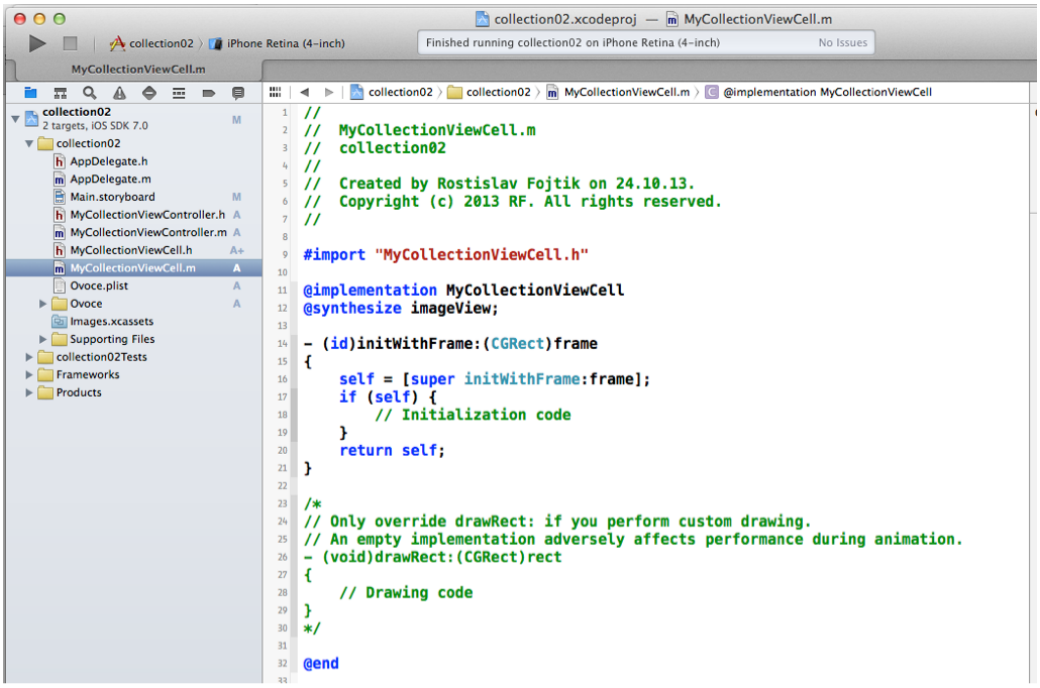
Obrázek 100 - zdrojové kódy



Obrázek 101 - zdrojové kódy



Obrázek 102 - zdrojové kódy



Obrázek 103 - zdrojové kódy

10. UCHOVÁNÍ DAT

V této kapitole se dozvíte:

Cílem této lekce seznámit se s možnostmi uchování dat v aplikacích.

Po absolvování lekce budete:

- umět ukládat data aplikací různými způsoby
- umět využívat SQLite

Klíčová slova této kapitoly:
seznam vlastností, archív objektů, SQLite, Core Data

Čas ke studiu: 3 hodiny

Data využívané mobilními aplikacemi lze ukládat následujícími způsoby:

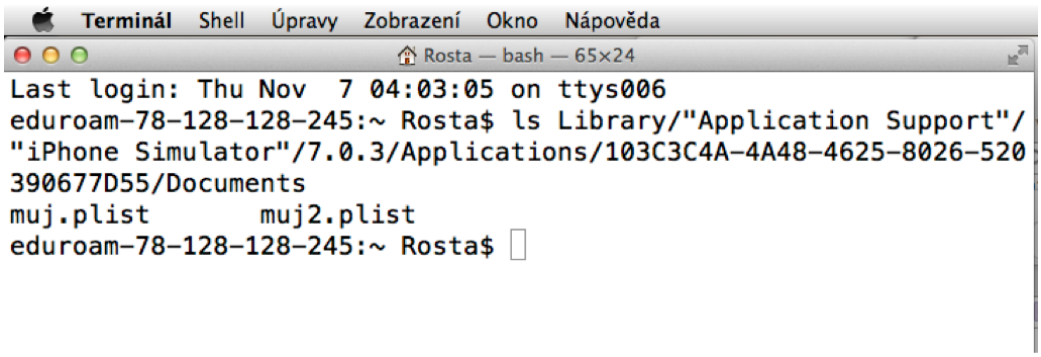
- seznam vlastností
- archív objektů
- realační databáze SQLite
- Core Data
- IO funkce

Seznam vlastností

Lze využít pouze pro serializované objekty tříd NSArray, NSMutableArray, NSDictionary, NSMutableDictionary, NSData, NSMutableData, NSString, NSMutableString, NSNumber, NSDate. Soubory s daty se ukládají do složky *Directory*. Její obsah si můžeme prohlédnout například přes terminál.

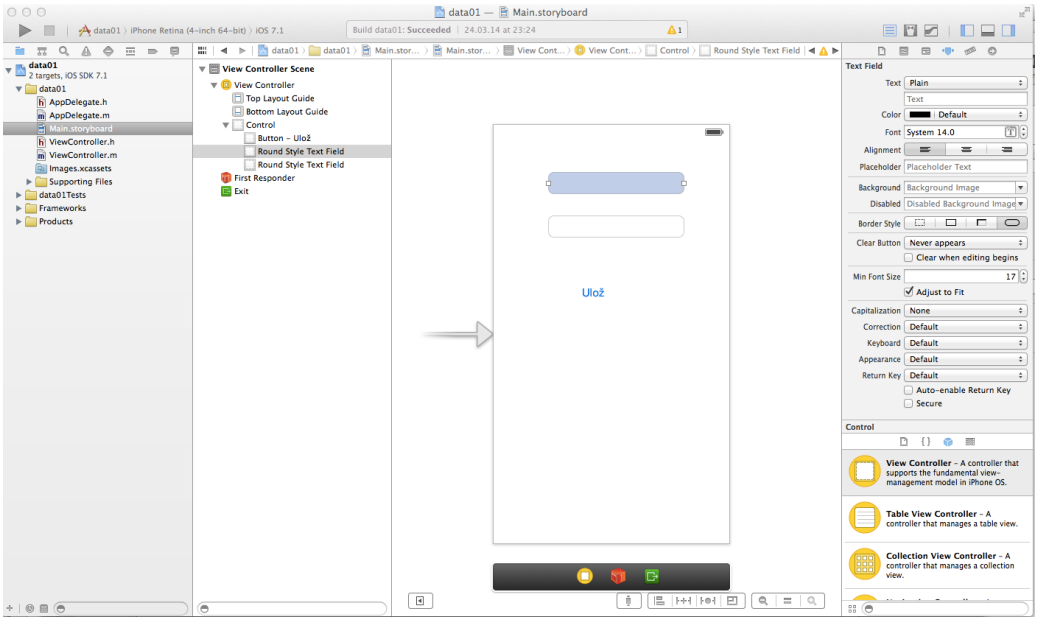
```
/Users/<user name>/Library/Application Support
/iPhone Simulator/<sdk version>/Applications/<app
id>/Documents
```

Obrázek 104 - cesta ke složce Directory



Obrázek 105 - terminál, přístup ke složce Directory

Příklad



Obrázek 106 - návrh aplikace

Kód aplikace:

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

@interface ViewController : UIViewController
@property (weak, nonatomic) IBOutlet UITextField
*editName;
@property (weak, nonatomic) IBOutlet UITextField
*editCity;

- (IBAction)Save:(id)sender;
-(IBAction)backgroundTap:(id)sender;

@end

//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize editCity,editName;

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
    NSString *path = [self dataFieldPath];
    if ([[NSFileManager defaultManager]
fileExistsAtPath:path])
```

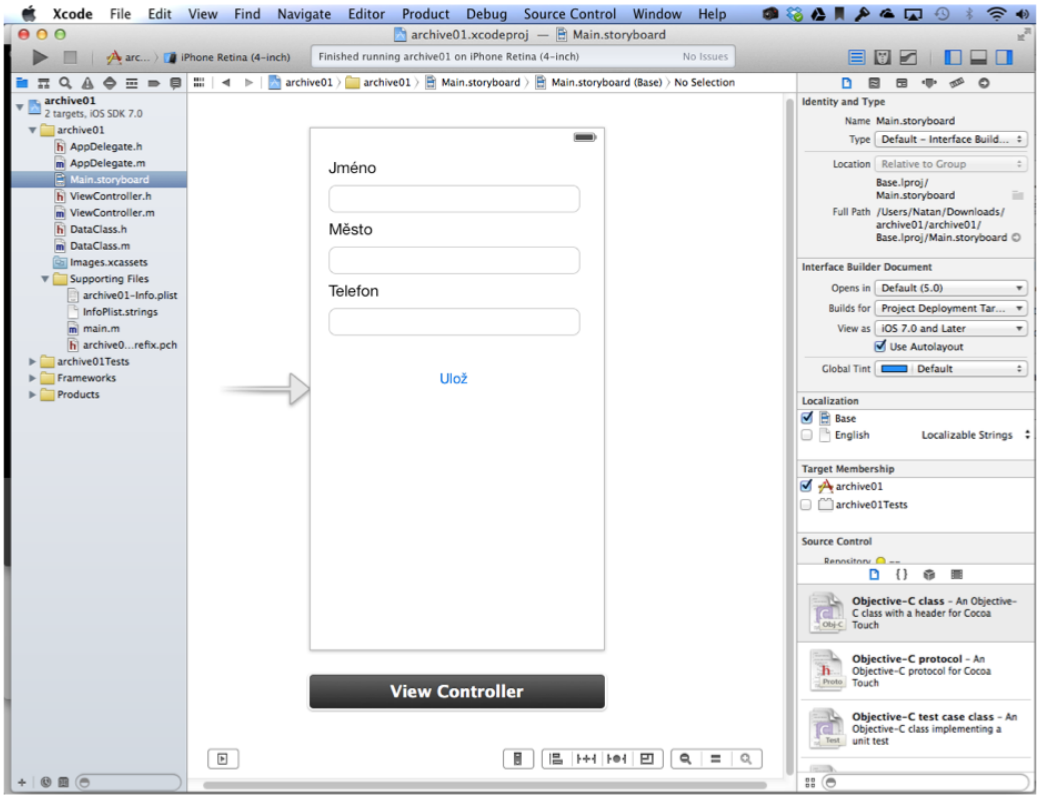
```
    {
        NSMutableArray *array = [[NSMutableArray alloc]
initWithContentsOfFile:path];
        editName.text = [array objectAtIndex:0];
        editCity.text = [array objectAtIndex:1];
    }
}


- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(NSString *) dataFieldPath
{
    NSArray *path =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *documentDir = path[0];//[path
objectAtIndex:0];

    NSString *fileName = [documentDir
stringByAppendingPathComponent:@"muj.plist"];

    return fileName;
}
- (IBAction)Save:(id)sender {

    NSMutableArray *array = [[NSMutableArray alloc] init];
    [array addObject:editName.text];
    [array addObject:editCity.text];
    [array writeToFile:[self dataFieldPath]
atomically:YES];
}
- (IBAction)backgroundTap:(id)sender
{
    [editName resignFirstResponder];
    [editCity resignFirstResponder];
}

@end
```

Archív objektů

Archív objektů slouží k ukládání dat libovolné třídy.

Příklad

Vytvořme aplikaci, která bude ukládat data pomocí seznamu vlastností.



Obrázek 107 - aplikace pro ukládání seznamu vlastností

Kód jednotlivých souborů aplikace:

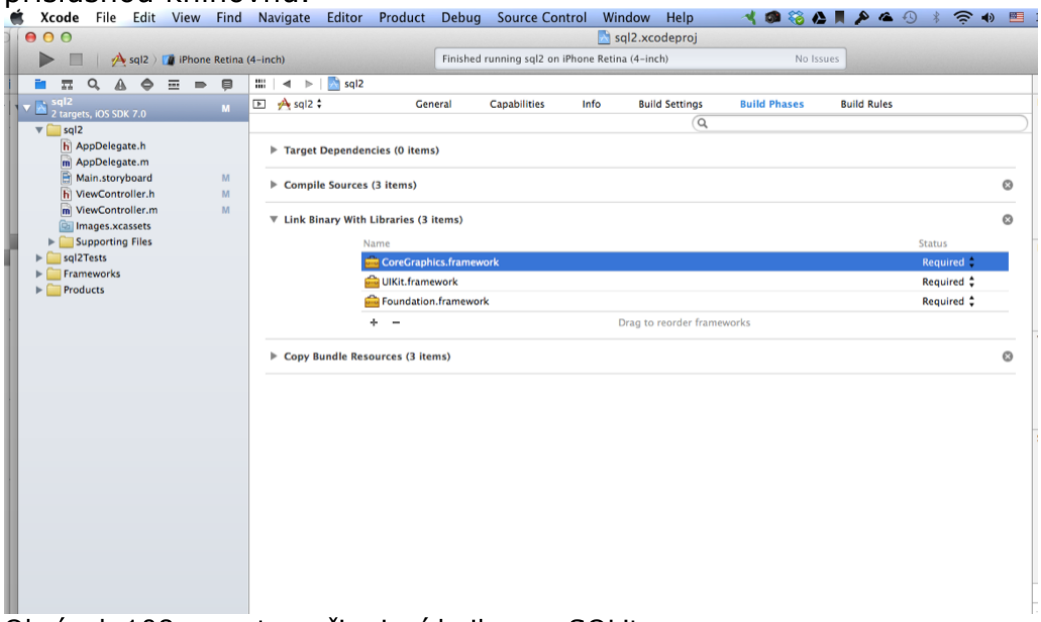
```
//  ViewController.h
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
@property (weak, nonatomic) IBOutlet UITextField
*editName;
@property (weak, nonatomic) IBOutlet UITextField
*editCity;
@property (weak, nonatomic) IBOutlet UITextField
*editPhone;
@property (strong, nonatomic) NSString *dataFilePath;

- (IBAction) saveInfo:(id)sender;
- (IBAction) backgroundTap:(id)sender;
@end

//  ViewController.m
#import "ViewController.h"
#import "DataClass.h"

@interface ViewController ()

@end

@implementation ViewController

@synthesize editCity, editName, editPhone;
@synthesize dataFilePath;
```

```
-(NSString *) dataFieldPath
{
    NSArray *path =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *documentDir = [path objectAtIndex:0];
    NSString *fileName = [documentDir
stringByAppendingPathComponent:@"archive.plist"];

    return fileName;

}

- (void)viewDidLoad
{
    [super viewDidLoad];
    NSFileManager *flm;
    NSArray *dirPaths;
    NSString *docsDir;

    flm = [NSFileManager defaultManager];
    dirPaths =
NSSearchPathForDirectoriesInDomains(NSDocumentationDirecto
ry, NSUserDomainMask, YES);
    docsDir =dirPaths[0];
    dataFilePath = [[NSString alloc]
initWithString:[docsDir
stringByAppendingString:@"data.archive"]];

    if ([flm fileExistsAtPath:dataFilePath])
    {
        NSMutableArray *dataArray;
        dataArray = [NSKeyedUnarchiver
unarchiveObjectWithFile:dataFilePath];
        editName.text = dataArray[0];
        editCity.text = dataArray[1];
        editPhone.text = dataArray[2];
    }
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)saveInfo:(id)sender
{

    NSMutableArray *myData = [[NSMutableArray alloc]
init];
    [myData addObject:editName.text];
    [myData addObject:editCity.text];
    [myData addObject:editPhone.text];
    [NSKeyedArchiver archiveRootObject:myData
toFile:dataFilePath];
```

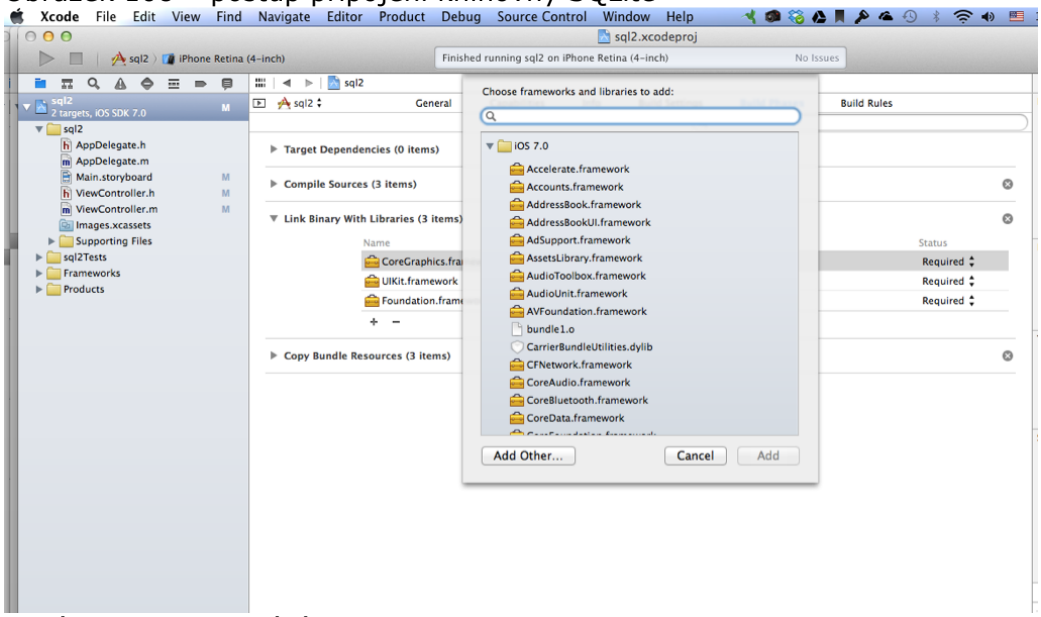
```
}

- (IBAction)backgroundTap:(id)sender
{
    [editName resignFirstResponder];
    [editCity resignFirstResponder];
    [editPhone resignFirstResponder];
}
@end

//  DataClass.h
#import <UIKit/UIKit.h>

@interface DataClass : NSObject <NSCoding, NSCopying>

@property (nonatomic,retain) NSString *Name;
@property (nonatomic,retain) NSString *City;
@property (nonatomic,retain) NSString *Phone;

@end

//  DataClass.m
#import "DataClass.h"

@implementation DataClass

@synthesize Name, City, Phone;

- (void) encodeWithCoder:(NSCoder *)aCoder
{
    [aCoder encodeObject:Name forKey:@"Name"];
    [aCoder encodeObject:Phone forKey:@"Phone"];
    [aCoder encodeObject:Name forKey:@"Name"];
}
- (id) initWithCoder:(NSCoder *)aDecoder
{
   if (self = [super init])
   {
       self.Name = [aDecoder decodeObjectForKey:@"Name"];
       self.City = [aDecoder decodeObjectForKey:@"City"];
       self.Phone = [aDecoder
decodeObjectForKey:@"Phone"];
   }
    return self;
}
- (id) copyWithZone:(NSZone *)zone
{
    DataClass *copy = [[[self class]
allocWithZone:zone]init];
    copy.Name = [self.Name copyWithZone:zone];
    copy.City = [self.City copyWithZone:zone];
    copy.Phone = [self.Phone copyWithZone:zone];

    return copy;
}
@end
```

SQLite

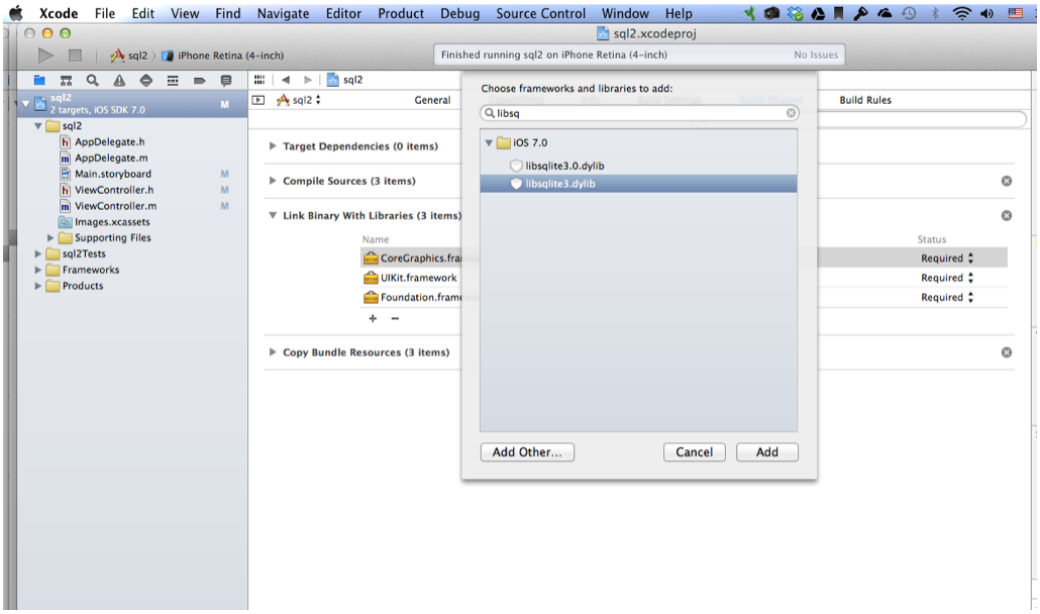
Pro ukládání a práci s daty mobilních aplikací v iOS je možné využívat rovněž relační databázi SQLite, která nabízí rychlejší komplexní agregaci. Pro převod do objektově orientovaného modelu je potřeba provést objektově-relační mapování. Je potřeba si uvědomit, že SQLite je napsána v jazyce C a ne v Objective-C. Pokud vytváříme aplikaci, která má využívat SQLite, je potřeba připojit příslušnou knihovnu.



Obrázek 108 – postup připojení knihovny SQLite



Obrázek 109 – hledání knihovny v seznamu



Obrázek 110 - výběr příslušné knihovny

Otevření databáze:

```
import <sqlite3.h>

sqlite3 *contactDB;
//. . .

const char *dbpath = [databasePath UTF8String];

if (sqlite3_open(dbpath, &contactDB) == SQLITE_OK)
{
   //. . .
}
```

Uzavření databáze:

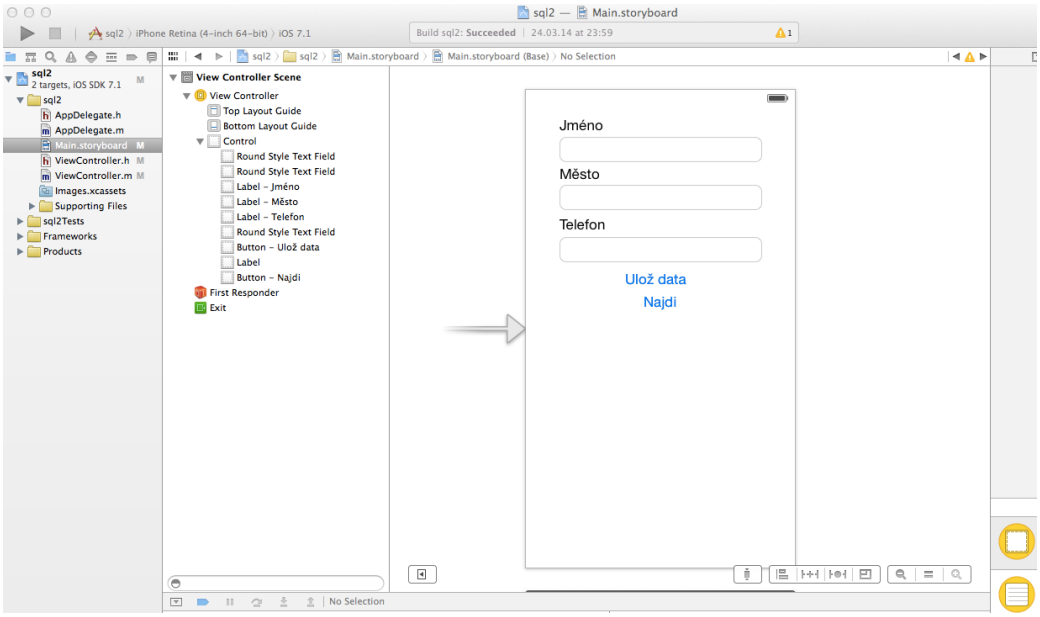
```
sqlite3_close(contactDB);
```

Vytvoření nové tabulky:

```
const char *sql_stmt = "CREATE TABLE IF NOT EXISTS
CONTACTS (ID INTEGER PRIMARY KEY AUTOINCREMENT, NAME TEXT,
CITY TEXT, PHONE TEXT)";
```

Načítání dat:

```
const char *sql_stmt = "CREATE TABLE IF NOT EXISTS
CONTACTS (ID INTEGER PRIMARY KEY AUTOINCREMENT, NAME TEXT,
CITY TEXT, PHONE TEXT)";
```



Obrázek 111 - návrh aplikace

Kódy aplikace:

```
//  ViewController.h
#import <UIKit/UIKit.h>
#import <sqlite3.h>

@interface ViewController : UIViewController

@property (strong, nonatomic) NSString *databasePath;
@property (nonatomic) sqlite3 *contactDB;

@property (weak, nonatomic) IBOutlet UITextField
*editName;
@property (weak, nonatomic) IBOutlet UITextField
*editPhone;
@property (weak, nonatomic) IBOutlet UITextField
*editCity;
@property (weak, nonatomic) IBOutlet UILabel *status;

- (IBAction)backgroundTap:(id)sender;

- (IBAction)findData:(id)sender;
- (IBAction)saveData:(id)sender;

@end

//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize databasePath, contactDB, status;
@synthesize editName, editCity, editPhone;
```

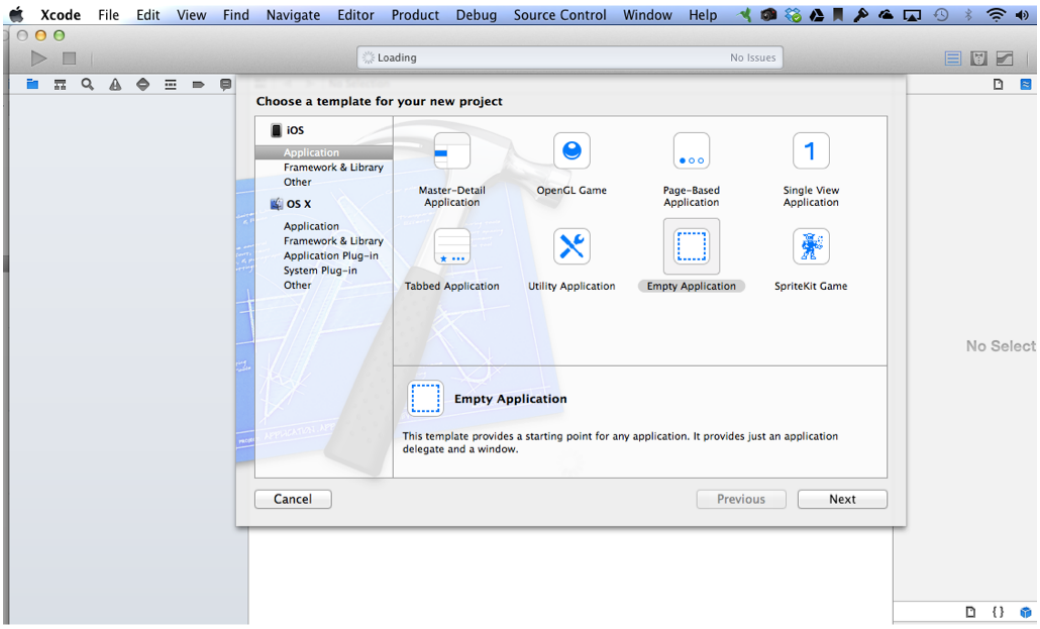
```
- (void)viewDidLoad
{
    [super viewDidLoad];

    NSString *docDir;
    NSArray *dirPaths;

    dirPaths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    docDir = dirPaths[0];

    databasePath = [[NSString alloc]
initWithString:[docDir stringByAppendingPathComponent:
@"contacts.db" ]];
    NSFileManager *fileManager = [NSFileManager
defaultManager];
    if ([fileManager fileExistsAtPath:databasePath] == NO)
    {
        const char *dbpath = [databasePath UTF8String];
        if (sqlite3_open(dbpath, &contactDB) == SQLITE_OK)
        {
            char *error;
            const char *sql_stmt = "CREATE TABLE IF NOT
EXISTS CONTACTS (ID INTEGER PRIMARY KEY AUTOINCREMENT,
NAME TEXT, CITY TEXT, PHONE TEXT)";
            if (sqlite3_exec(contactDB, sql_stmt, NULL,
NULL, &error) != SQLITE_OK)
            {
                status.text = @"Chyba vytvoření
databáze!";
            }
            //sqlite3_close(contactDB);
            else
            {
                status.text = @"Chyba vytvoření
databáze!";
            }
        }
    }

}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)backgroundTap:(id)sender
{
    [editPhone resignFirstResponder];
    [editName resignFirstResponder];
    [editCity resignFirstResponder];
}
```

```
- (IBAction)findData:(id)sender
{
    const char *dbpath = [databasePath UTF8String];
    sqlite3_stmt *statement;

    if (sqlite3_open(dbpath, &contactDB) == SQLITE_OK)
    {
        NSString *querySQL = [NSString
stringWithFormat:@"SELECT city, phone FROM contacts WHERE
name=\"%@\"",editName.text];
        const char *query_stmt = [querySQL UTF8String];
        if (sqlite3_prepare_v2(contactDB, query_stmt, -1,
&statement, NULL) == SQLITE_OK)
        {
            if (sqlite3_step(statement) == SQLITE_ROW)
            {
                NSString *cityField = [[NSString alloc]
initWithUTF8String:(const char*)
sqlite3_column_text(statement, 0)];
                editCity.text = cityField;
                NSString *phoneField = [[NSString alloc]
initWithUTF8String:(const char
*)sqlite3_column_text(statement, 1)];
                editPhone.text = phoneField;
                status.text = @"Kontakt nalezen";
            }
            else
            {
                status.text = @"Kontakt nenalezen!";
                editCity.text = @"";
                editPhone.text = @"";
            }
            sqlite3_finalize(statement);
        }
        sqlite3_close(contactDB);
    }
}

- (IBAction)saveData:(id)sender
{
    sqlite3_stmt *statement;
    const char *dbpath = [databasePath UTF8String];

    if (sqlite3_open(dbpath, &contactDB) == SQLITE_OK)
    {
        NSString *insertSQL = [NSString
stringWithFormat:@"INSERT INTO CONTACTS (name, city,
phone) VALUES(\"%@\",\"%@\",\"%@\")",editName.text,
editCity.text, editPhone.text];
        const char *insert_stmt = [insertSQL UTF8String];
        sqlite3_prepare_v2(contactDB, insert_stmt, -1,
&statement, NULL);
        if (sqlite3_step(statement) == SQLITE_DONE)
        {
            status.text = @"Kontakt přidán";
```

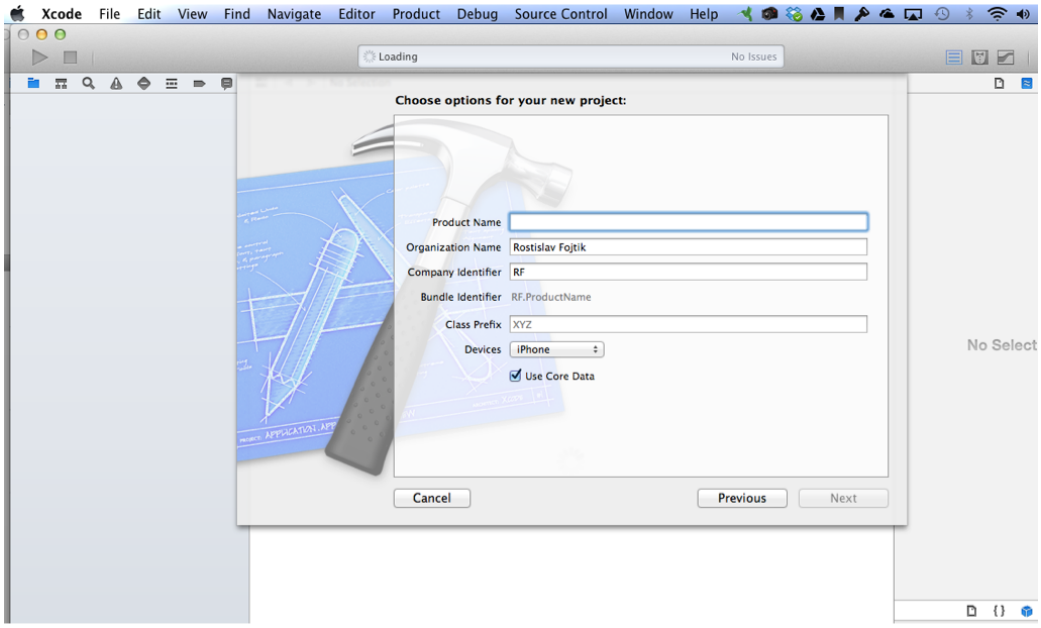
```
            editName.text = @"";
            editCity.text = @"";
            editPhone.text = @"";
        }
        else
        {
            status.text = @"Chyba při přidávání
kontaktu!";
        }
        sqlite3_finalize(statement);
        sqlite3_close(contactDB);
    }
}
@end
```

Core Data

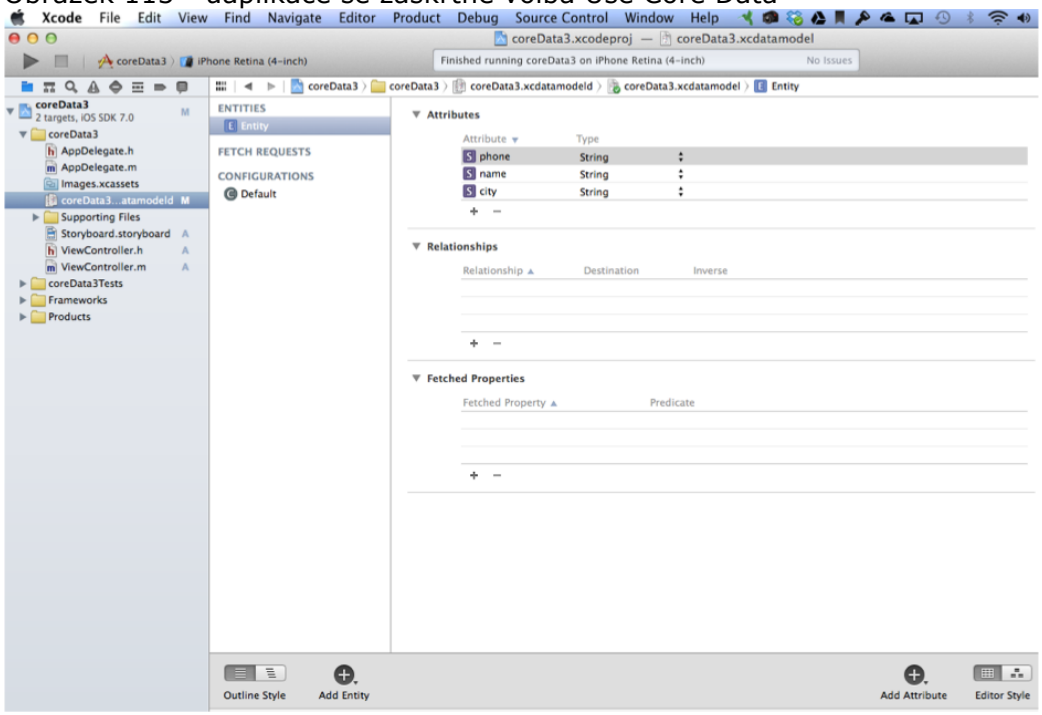
Core Data jsou nástroj k vizuálnímu vytváření datového modelu. Následující obrázky ukazují postupu vytvoření vizuálního modelu.



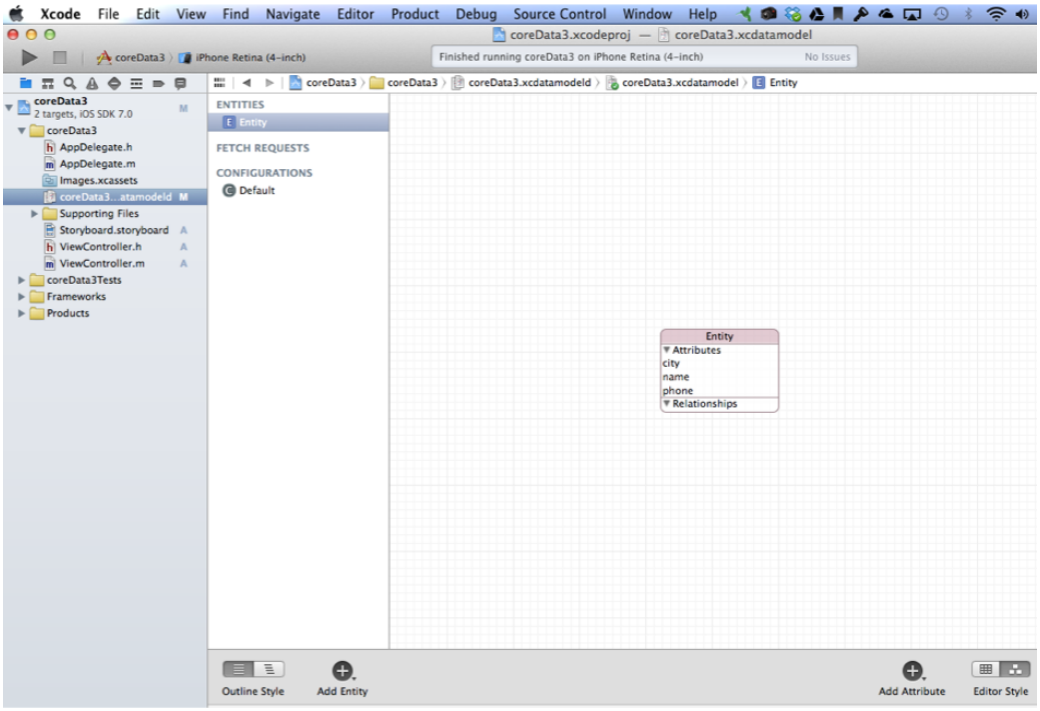
Obrázek 112 - vytvoření nové prázdné aplikace



Obrázek 113 - uaplikace se zaškrtne volba Use Core Data



Obrázek 114 - vytvoření entity s atributy



Obrázek 115 - pohled na entitu

Shrnutí kapitoly

Data v mobilních aplikací lze ukládat následujícími pomocí:

- seznam vlastností
- archív objektů
- realační databáze SQLite
- Core Data
- IO funkce

11. APLIKACE VYUŽÍVAJÍCÍ ICLOUD

V této kapitole se dozvíte:

Cílem této lekce seznámit se s možnostmi datového úložiště iCloud.

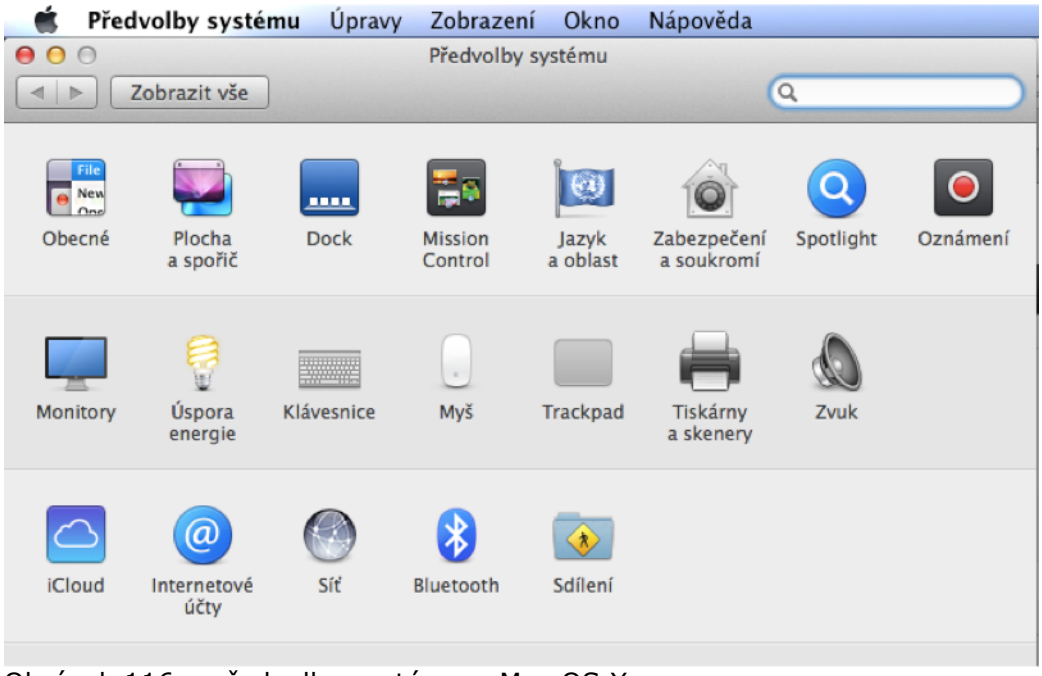
Po absolvování lekce budete:

- umět využívat uložiště iCloud pro své aplikace

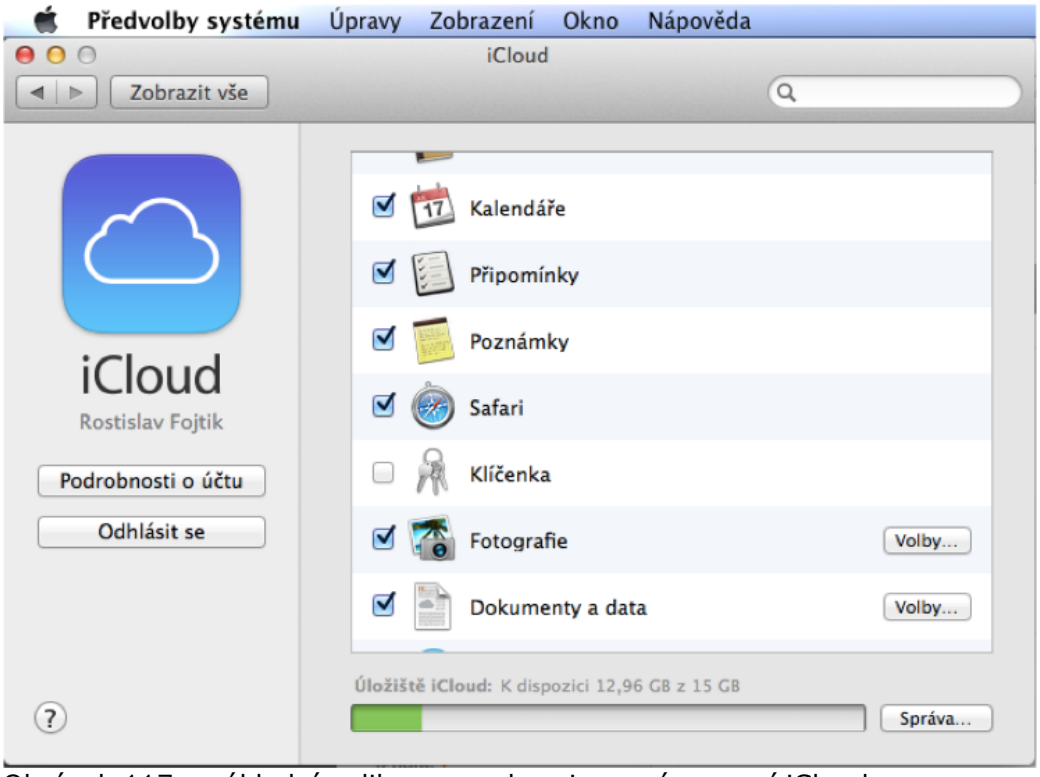
Klíčová slova této kapitoly:
iCloud

Čas ke studiu: 3 hodiny

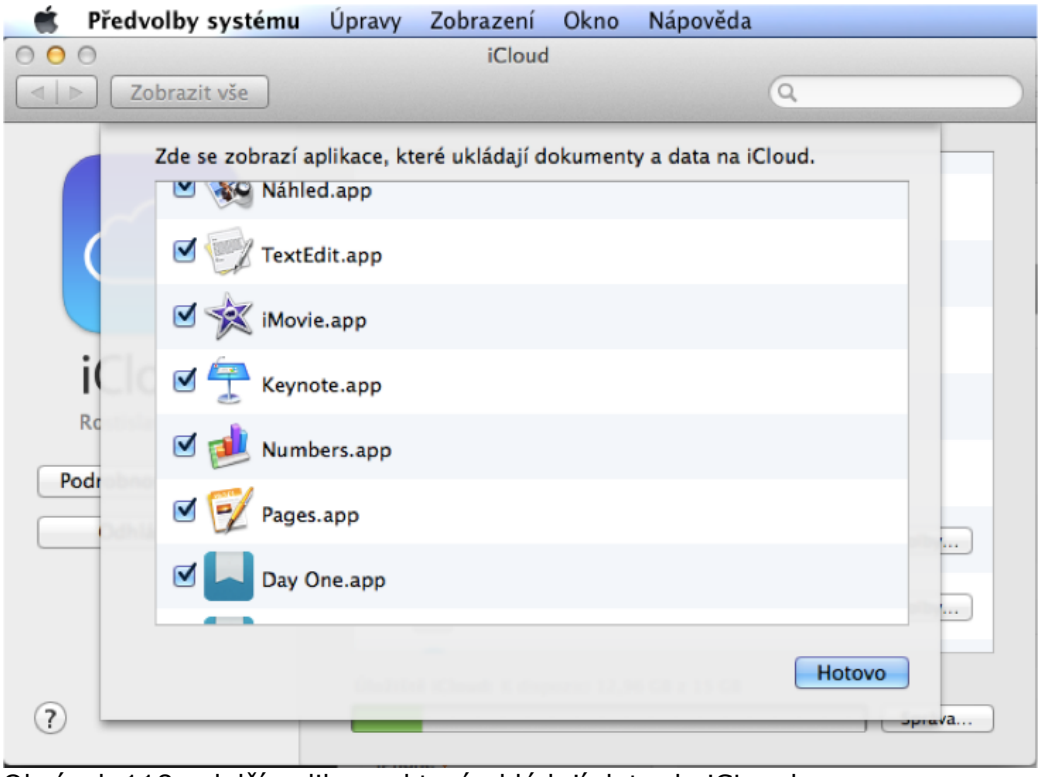
Datové uložiště firmy Apple poskytuje v základu 5 GB. Za poplatek lze kapacitu uložiště zvýšit. Uložiště iCloud slouží primárně k ukládání a synchronizaci dat některých aplikací. Data mohou být synchronizována mezi mobilními i desktopovými zařízeními.



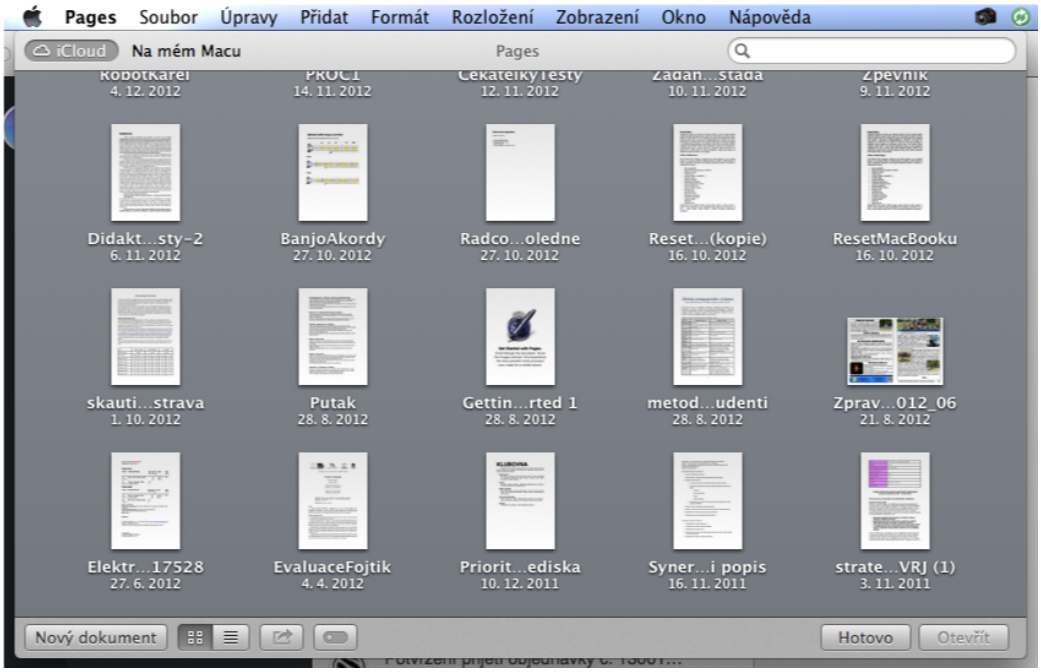
Obrázek 116 – předvolby systému v Mac OS X



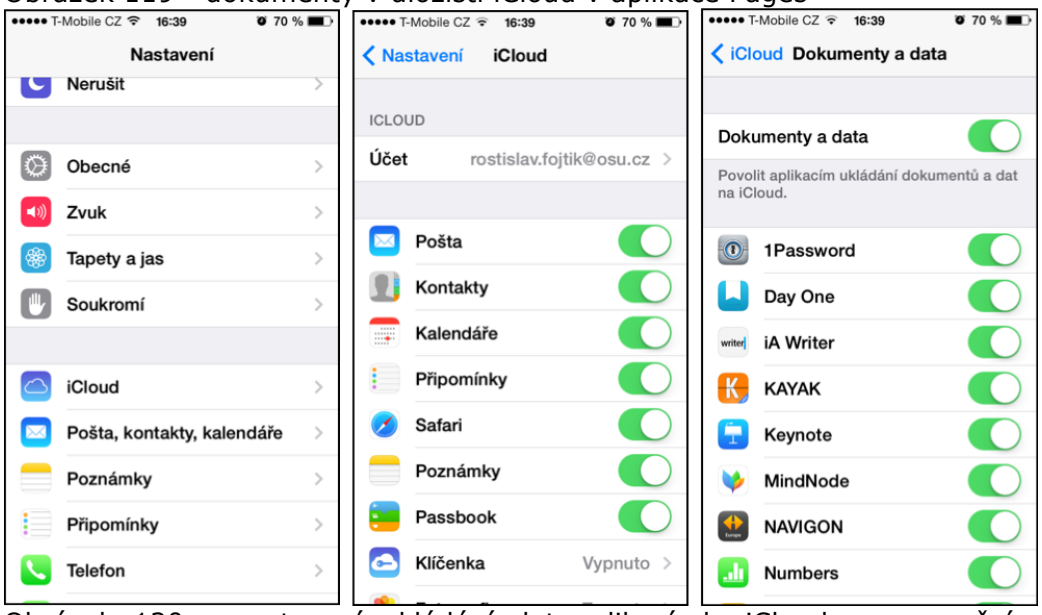
Obrázek 117 – základní aplikace synchronizované pomocí iCloud



Obrázek 118 - další aplikace, které ukládají data do iCLoud



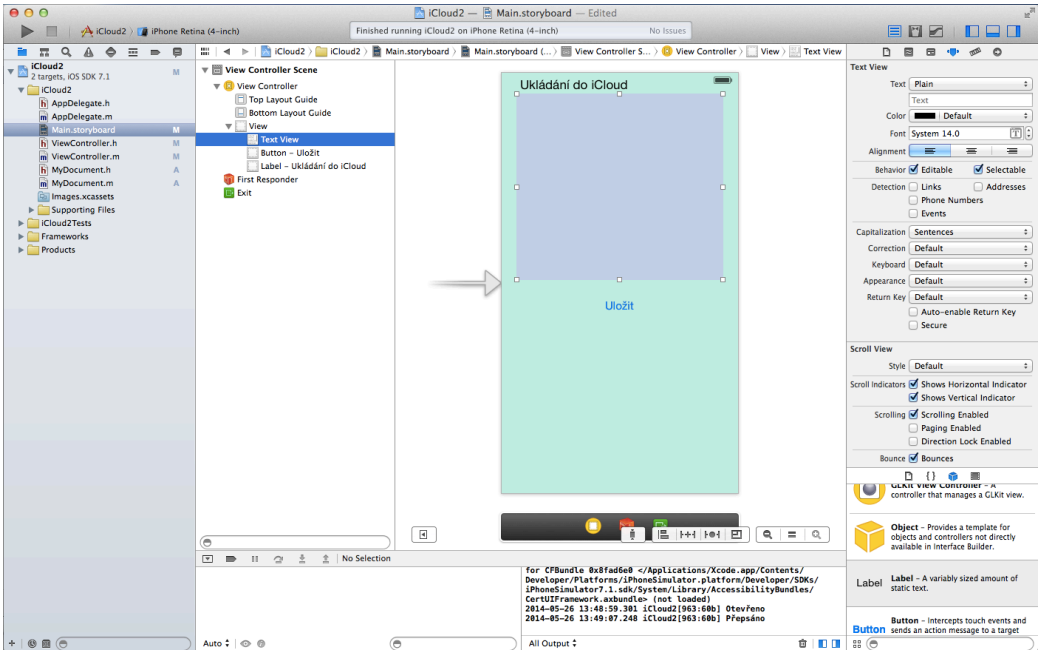
Obrázek 119 - dokumenty v uložišti iCloud v aplikace Pages



Obrázek 120 - nastavení ukládání dat aplikací do iCloudu v operačním systému iOS

Příklad

Následující ukázkový příklad využívá k ukládání dat iCloud. Program bude obsahovat textové pole, do kterého si uživatel může zapsat poznámku a pomocí tlačítka poznámku uloží do uložiště iCloud. Při opětovném spuštění aplikace se text poznámky objeví v aplikaci.



Obrázek 121 - návrh uživatelského rozhraní

Nejprve vytvořme novou třídu, která bude potomkem třídy UIDocument.

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

@interface MyDocument : UIDocument

@property (strong, nonatomic) NSString *userText;
@end

//  MyDocument.m
#import "MyDocument.h"

@implementation MyDocument

@synthesize userText;

-(id)contentsForType:(NSString *)typeName error:(NSError
*__autoreleasing *)outError
{
    return [NSData dataWithBytes:[userText UTF8String]
length:[userText length]];
}
-(BOOL)loadFromContents:(id)contents ofType:(NSString
*)typeName error:(NSError *__autoreleasing *)outError
{
    if ([contents length] > 0)
    {
        userText = [[NSString alloc]
initWithBytes:[contents bytes] length:[contents length]
encoding:NSUTF8StringEncoding];
    }
    else
    {
```

```
        userText = @"";
    }
    return YES;
}
@end
```

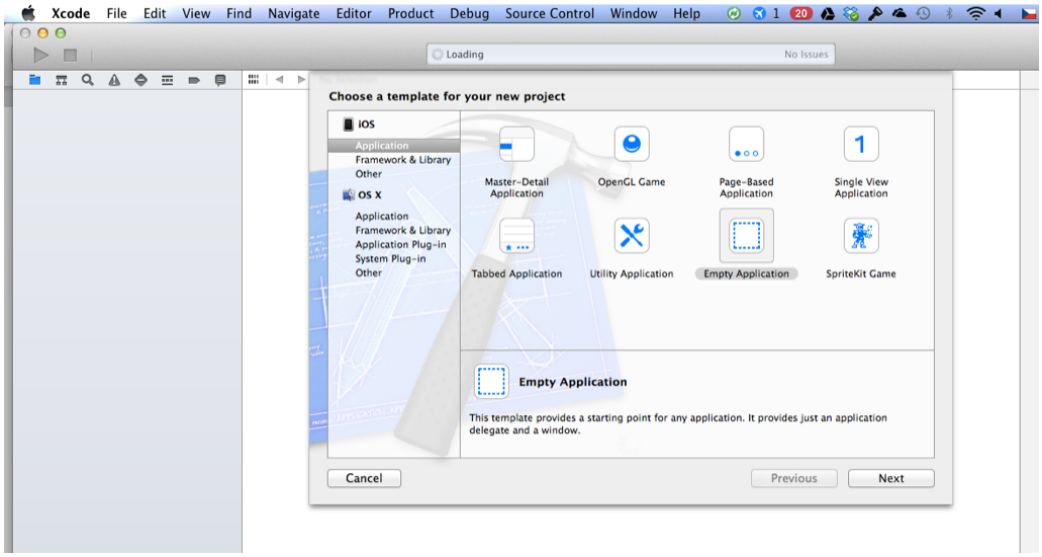
Třídu použijeme ve třídě ViewController.

```
//  ViewController.h
#import <UIKit/UIKit.h>
#import "MyDocument.h"

@interface ViewController : UIViewController

@property (strong, nonatomic) MyDocument *document;
@property (strong, nonatomic) NSURL *documentURL;

@property (strong, nonatomic) IBOutlet UITextView *myText;
- (IBAction)saveText:(id)sender;

@end

//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize document, documentURL, myText;

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
    NSArray *dirPaths =
NSSearchPathForDirectoriesInDomains(NSDocumentationDirecto
ry, NSUserDomainMask, YES);
    NSString *docsDir = dirPaths[0];
    NSString *dataFile = [docsDir
stringByAppendingString:@"document.txt"];
    documentURL = [NSURL fileURLWithPath:dataFile];
    document = [[MyDocument alloc]
initWithFileURL:documentURL];
    document.userText = @"";
    NSFileManager *fileManager = [NSFileManager
defaultManager];
    if ([fileManager fileExistsAtPath:dataFile])
    {
        [document openWithCompletionHandler:^(BOOL succes)
         {
             if (succes)
             {
                 NSLog(@"Otevřeno");
                 myText.text = document.userText;
             }
```

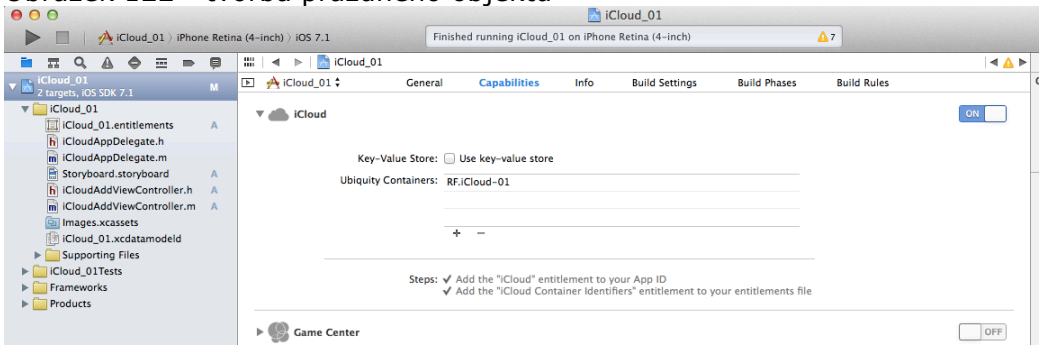
```
            else
            {
                NSLog(@"Neotevřeno");
            }
        }
        ];
    }
    else
    {
        [document saveToURL:documentURL
forSaveOperation:UIDocumentSaveForCreating
completionHandler:^(BOOL succes)
            {
                if (succes)
                {
                    NSLog(@"Vytvořeno");
                }
                else
                {
                    NSLog(@"Nevytvořeno");
                }
            }
        ];
    }
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)saveText:(id)sender
{
    [document saveToURL:documentURL
forSaveOperation:UIDocumentSaveForOverwriting
completionHandler:^(BOOL succes)
     {
         if (succes)
         {
             NSLog(@"Přepsáno");
         }
         else
         {
             NSLog(@"Nepřepsáno");
         }
     }
];
}
@end
```

Příklad

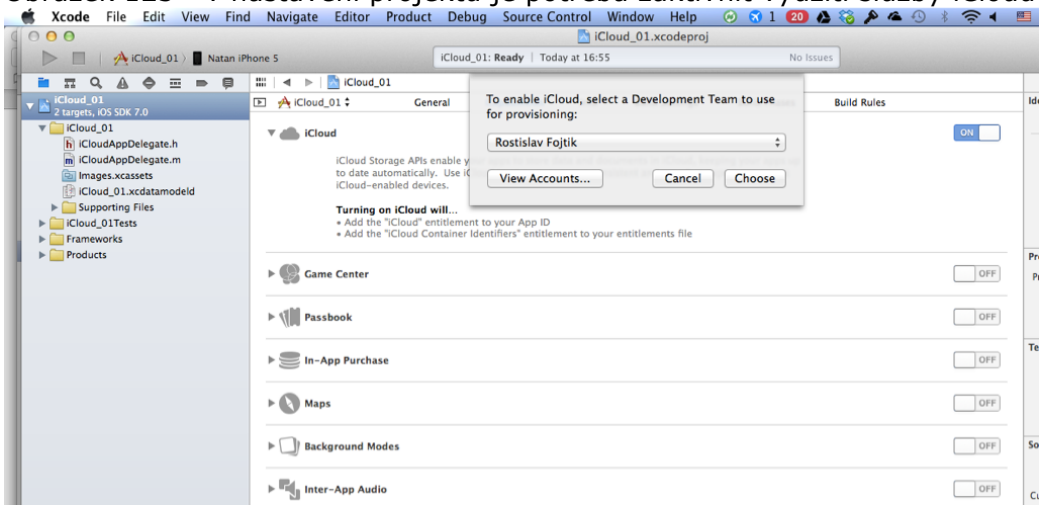
Vytvořme jednoduchou aplikaci pro ukládání poznámek do iCloudu. Nejprve vytvořme prázdný projekt, do kterého umístíme jednotlivé scény aplikace.



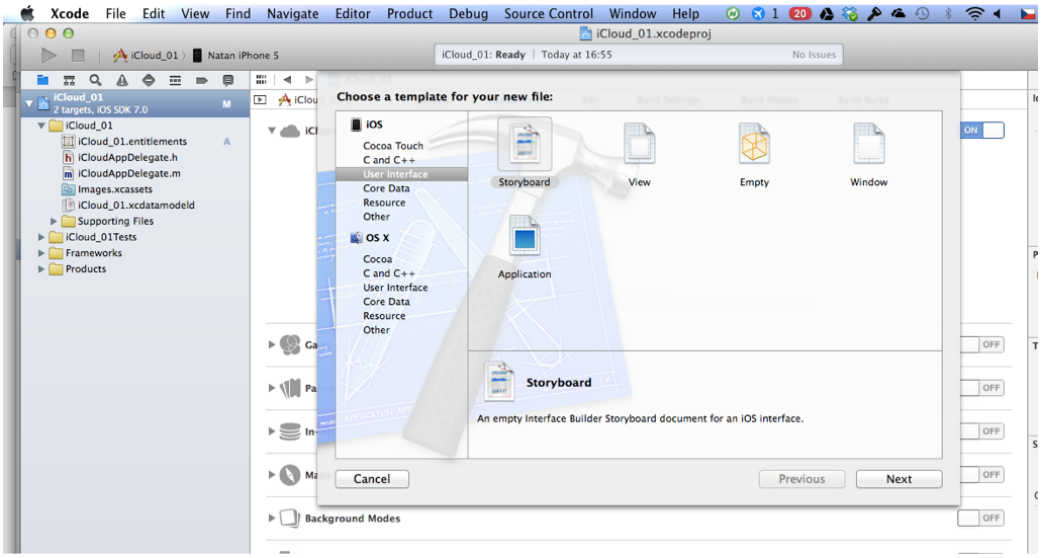
Obrázek 122 - tvorba prázdného objektu



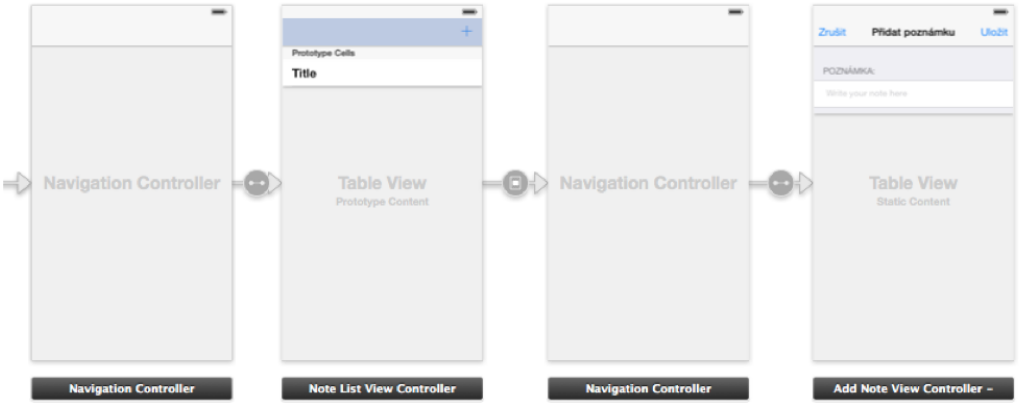
Obrázek 123 - v nastavení projektu je potřeba zaktivnit využití služby iCloud



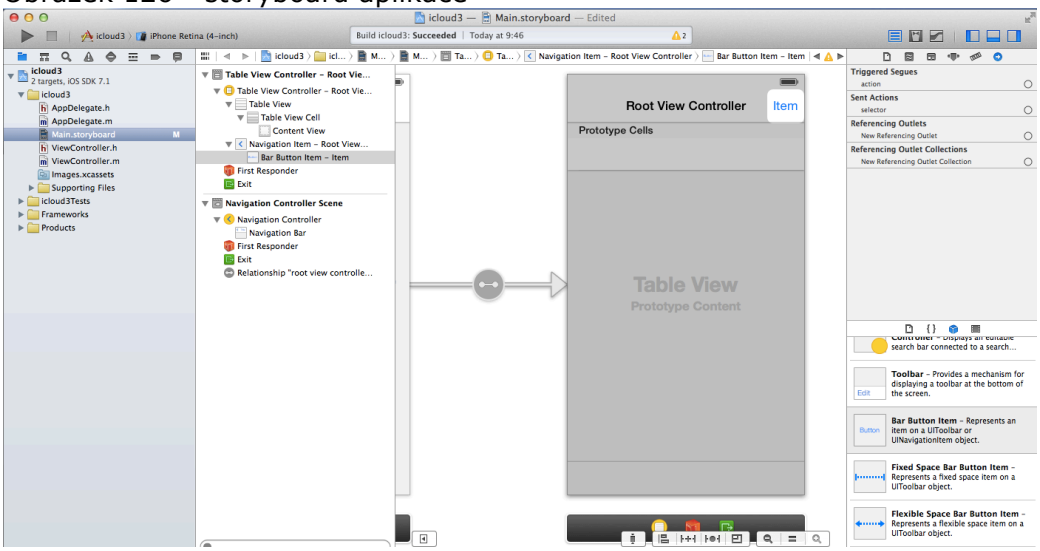
Obrázek 124 - výběr vyvojáře



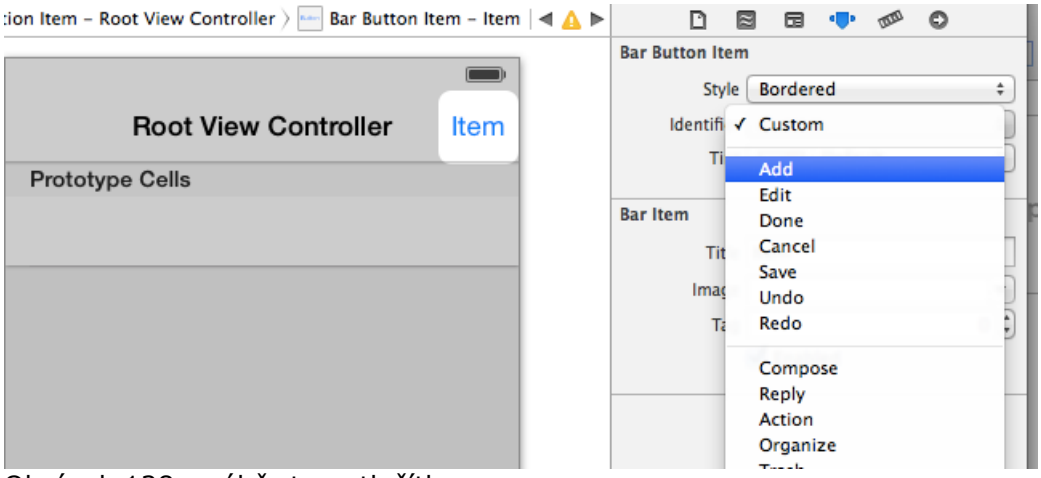
Obrázek 125 - vytvoření storyboardu



Obrázek 126 - storyboard aplikace



Obrázek 127 - vložení Bar Button Item

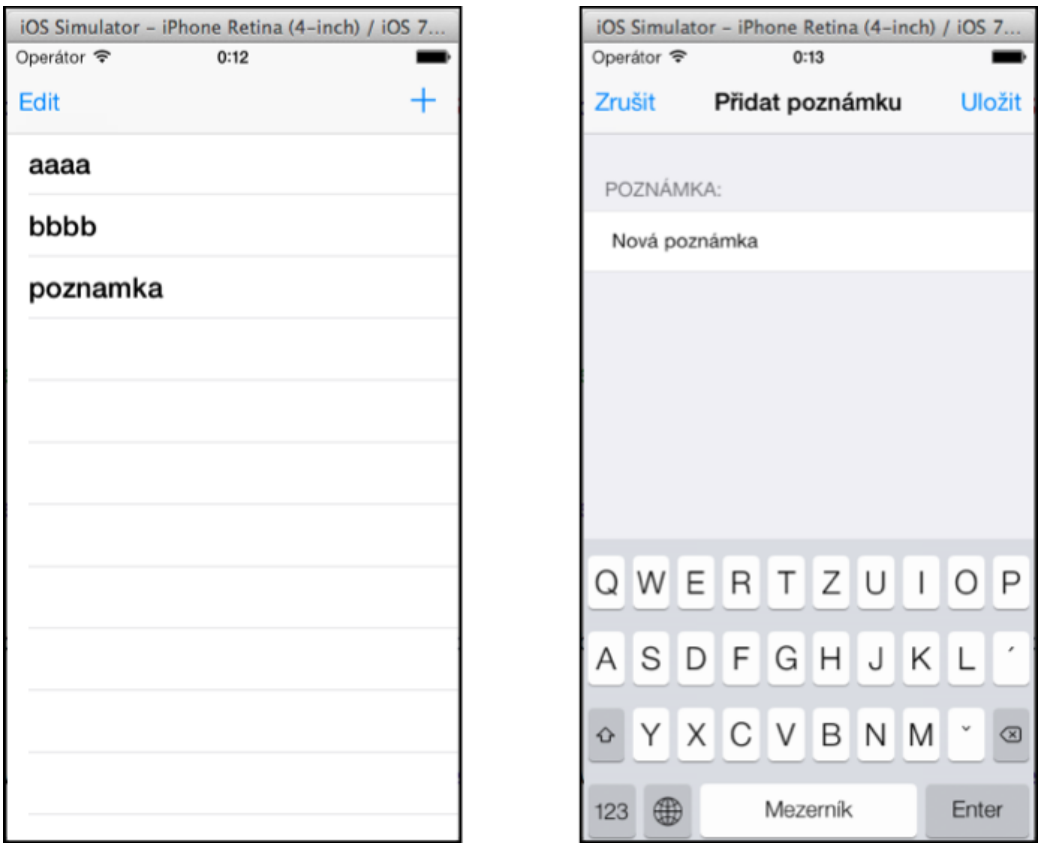


Obrázek 128 - výběr typu tlačítka

Obrázek 129 - změna typu tlačítka

```
- (IBAction)cancel:(id)sender {
    [self dismissViewControllerAnimated:YES
completion:nil];
}

- (IBAction)save:(id)sender {
    // Notify the previouse view to save the changes
locally
    [[NSNotificationCenter defaultCenter]
postNotificationName:@"New Note" object:self
userInfo:[NSDictionary
dictionaryWithObject:self.noteTextField.text
forKey:@"Note"]];

    [self dismissViewControllerAnimated:YES
completion:nil];
}


- (void)viewDidLoad
{
    [super viewDidLoad];

    self.navigationItem.leftBarButtonItem =
self.editButtonItem;

    //  Observer to catch changes from iCloud
    NSUbiquitousKeyValueStore *store =
[NSUbiquitousKeyValueStore defaultStore];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(storeDidChange:)
```

```
name:NSUbiquitousKeyValueStoreDidChangeExternallyNotificat
ion
object:store];

    [[NSUbiquitousKeyValueStore defaultStore]
synchronize];

    // Observer to catch the local changes
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(didAddNewNote:) name:@"New Note"
object:nil];
}
```



Obrázek 130 - výsledná podoba aplikace

Shrnutí kapitoly

Datové uložiště iCloud firmy Apple slouží k ukládání dat aplikací nebo synchronizaci dat aplikací mezi více zařízeními. Typicky mezi mobilním a desktopovým zařízením. Pro uživatel s Apple ID je zdarma k dispozici uložiště o kapacitě 5 GB, která lze za poplatek navýšit.

12. VYUŽITÍ GEST A DOTYKŮ

V této kapitole se dozvíte:

Cílem této lekce seznámit se s možnostmi využívat v aplikaci dotyků a gest.

Po absolvování lekce budete:

- umět pracovat s dotyky, klepnutí a gesty
- umět vytvářet jednoduchá gesta

Klíčová slova této kapitoly:
dotyky, gesta, klepnutí, multidotykový displej

Čas ke studiu: 3 hodiny

Zařízení využívající operační systém iOS (iPhone, iPad) využívají kapacitní dotykový displej, který podporuje multidotykové ovládání. Uživatel může využít více prstů najednou. Při ovládání aplikací pomocí dotykového displeje můžeme využít:

- gesta (gesture)
- dotyk (touch)
- klepnutí (tap)

Pro více dotyková gesta můžeme využít následující zprávy (metody):

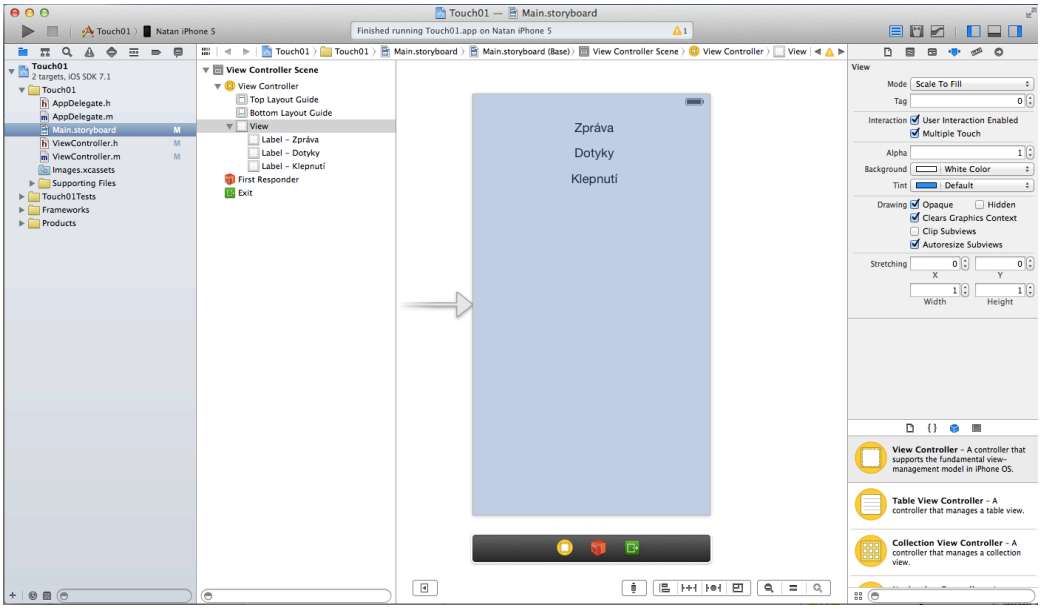
```
- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent
*)event
- (void) touchesCancelled:(NSSet *)touches
withEvent:(UIEvent *) event
- (void) touchesMoved:(NSSet *)touches withEvent:(UIEvent
*)event
- (void) touchesEnded:(NSSet *)touches withEvent:(UIEvent
*)event
```

Programátor si může vytvořit i vlastní gesta. Při jejich definici je potřeba vytvářet jednoduché pohyby. U složitých pohybu objektů by bylo velmi složité zajistit přesné snímání.

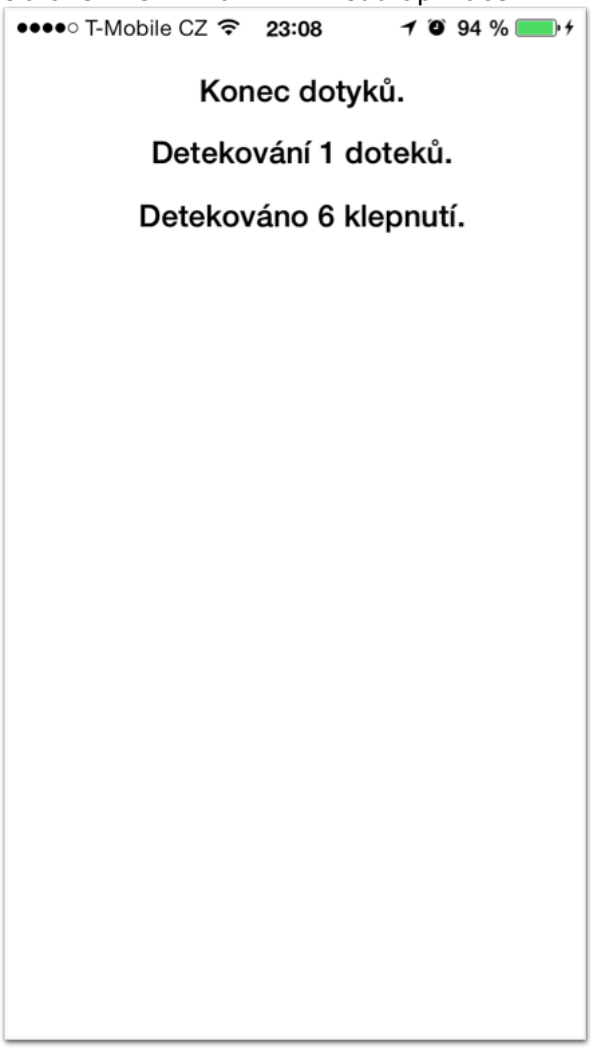
Příklad

Následující ukázková aplikace demonstruje způsoby ovládání aplikace dotyky, gesty a klepnutími. Podle toho, jakým způsobem pracuje s displejem, se vypíše typ a případný počet dotknutí.

Vytvořme nový projekt na základě šablony Single View Application a umístíme do View tři komponenty typu Label. V jednom se bude vypisovat o jaký typ gesta nebo dotyku se jedná. Druhé dva budou vypisovat počet doteků nebo klepnutí. Tuto aplikaci je vhodnější testovat na konkrétním reálném zařízení.



Obrázek 131 - návrh vzhledu aplikace



Obrázek 132 - aplikace na reálném zařízení

Prohledněte si zdrojové kódy aplikace:

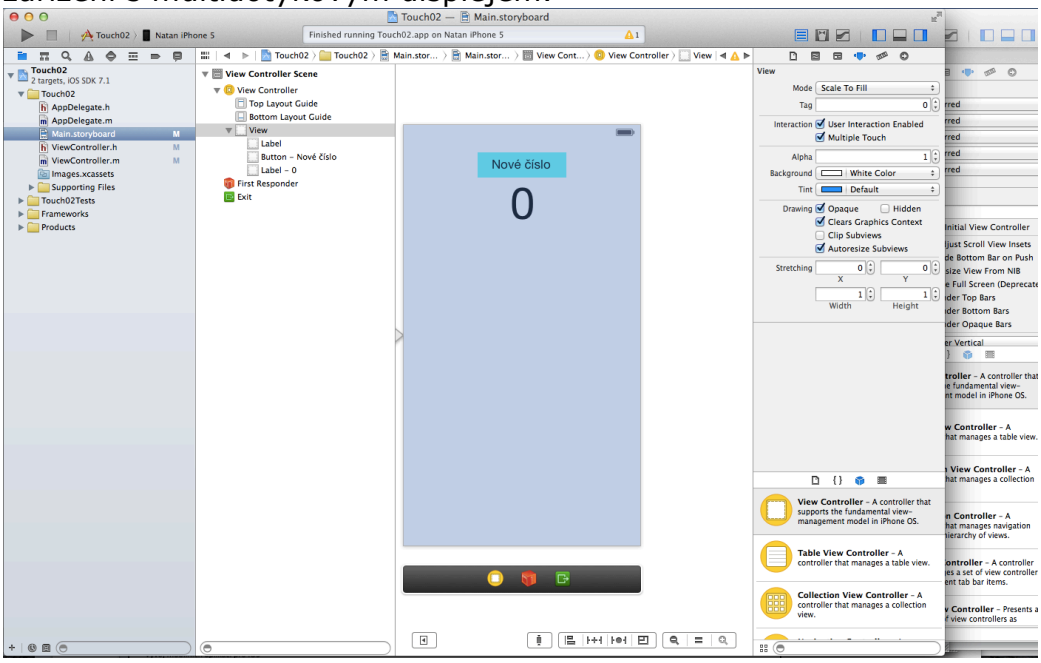
```
//  ViewController.h
#import <UIKit/UIKit.h>
```

```
@interface ViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel
*messageLabel;
@property (weak, nonatomic) IBOutlet UILabel
*touchesLabel;
@property (weak, nonatomic) IBOutlet UILabel *tapsLabel;

@end

//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize messageLabel, tapsLabel, touchesLabel;

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (void) updateLabelIsFromTouches:(NSSet *)touches
{
    NSUInteger numberTap = [[touches anyObject] tapCount];
    NSString * tapMessage = [[NSString alloc]
initWithFormat:@"Detekováno %d klepnutí.", numberTap];
    tapsLabel.text = tapMessage;

    NSUInteger numberTouche = [touches count];
    NSString *touchMessage = [[NSString alloc]
initWithFormat:@"Detekování %d doteků.", numberTouche];
    touchesLabel.text = touchMessage;
}

- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent
*)event
{
    messageLabel.text = @"Začátek dotyků.";
    [self updateLabelIsFromTouches:touches];
}
- (void) touchesCancelled:(NSSet *)touches
withEvent:(UIEvent *)event
{
    messageLabel.text = @"Dotyky zrušeny.";
```

```
    [self updateLabelIsFromTouches:touches];
}
- (void) touchesEnded:(NSSet *)touches withEvent:(UIEvent
*)event
{
    messageLabel.text = @"Konec dotyků.";
    [self updateLabelIsFromTouches:touches];
}
- (void) touchesMoved:(NSSet *)touches withEvent:(UIEvent
*)event
{
    messageLabel.text = @"Detekováno potažení.";
    [self updateLabelIsFromTouches:touches];
}

@end
```

Příklad

Druhý ukázkový příklad je vlastně taková jednoduchá hra. Na obrazovce se budou ukazovat náhodná čísla od 1 do 4 a úkolem „hráče" je udělat příslušný počet doteků. Tlačítkem vyvoláme nové číslo. Ve spodní části obrazovky se budou zobrazovat počty skutečně provedených doteků. Tato aplikace lze zkoušet jen na reálném zařízení s multidotykovým displejem.



Obrázek 133 - návrh vzhledu aplikace

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

@interface ViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *numberLabel;

@property (weak, nonatomic) IBOutlet UILabel
*feedbackLabel;
- (IBAction)newNumber:(id)sender;
```

```
@end

//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize  numberLabel, feedbackLabel;

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (void) updateLabelIsFromTouches:(NSSet *)touches
{
    NSInteger number = [numberLabel.text integerValue];

    NSUInteger numberTouche = [touches count];
    NSString *touchMessage = [[NSString alloc]
initWithFormat:@" - %d doteků.", numberTouche];

    NSString * str;
    if (number == numberTouche)
        str = [[NSString alloc] initWithFormat:@"Správně
%d %@", number, touchMessage];
    else
        str = [[NSString alloc] initWithFormat:@"Chyba %d
%@", number, touchMessage];

    feedbackLabel.text = str;
}

- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent
*)event
{
    [self updateLabelIsFromTouches:touches];
}
- (IBAction)newNumber:(id)sender
{
    static BOOL seeded = NO;
    if (!seeded) {
        seeded = YES;
        srandom(time(NULL));
    }
```

```
    NSInteger number =  (random() % 4) + 1;
    NSString *fedbackStr = [[NSString alloc]
initWithFormat:@"%d", number];
    numberLabel.text = fedbackStr;
}
@end
```

Shrnutí kapitoly

Hardwarová zařízení s iOS využívají multidotykové displeje. K ovládání aplikací pomocí dotykového displeje můžeme využít:

- gesta (gesture)
- dotyk (touch)
- klepnutí (tap)

13. PRÁCE S GRAFIKOU

V této kapitole se dozvíte:

Cílem této lekce seznámit se s možnostmi využití grafiky při tvorbě aplikací v iOS.

Po absolvování lekce budete:

- umět pracovat s knihovnami Quartz a OpenGL ES

Klíčová slova této kapitoly:
Core Graphics, OpenGL ES, Quartz

Čas ke studiu: 3 hodiny

Většina aplikací si vystačí pro vykreslení grafických komponent s frameworkem UIKit. Pro některé aplikace potřebují vykreslit vlastní grafické části. K tomu lze využít dvě knihovny:

- Quartz 2D
- OpenGL ES

OpenGL ES

OpenGL ES využívá zcela jiný přístup než Quartz 2D. Bod o souřadnicích (0,0) se nachází v levém dolním rohu. Knihovna pracuje jako stavový automat a funguje jako okno do třídimenzionálního světa. Knihovna OpenGL ES je výkonnější a komplexnější než knihovna Quartz.

Knihovna Quartz

Při kreslení se využívá virtuální plátno, na které se nanáší grafické 2D objekty. Při práci s knihovnou Quartz 2D se stejně jako při práci s frameworkem Core Graphics odehrává kreslení v *grafickém kontextu* (graphics context). Souřadnicový systém má bod (0,0) v levém horním rohu.

```
CGContextRef context = UIGraphicsGetCurrentContext();
```

- grafický kontext

```
CGContextMoveToPoint(context, x, y);
```

- přesun pera

```
CGContextAddLineToPoint(context, x, y);
```

- kreslení čáry

```
CGContextStrokePath(context);
```

- vykreslení

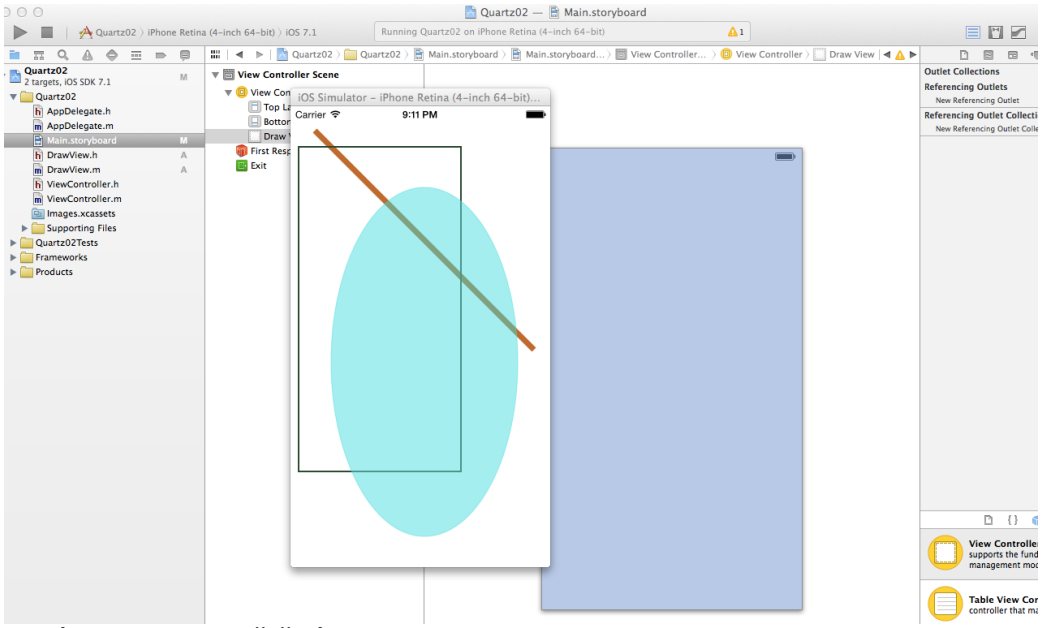
Příklad

Vytvořme aplikaci, která vykreslí úsečku, obdélník, elipsu.
Vytvoříme si třídu DrawView.

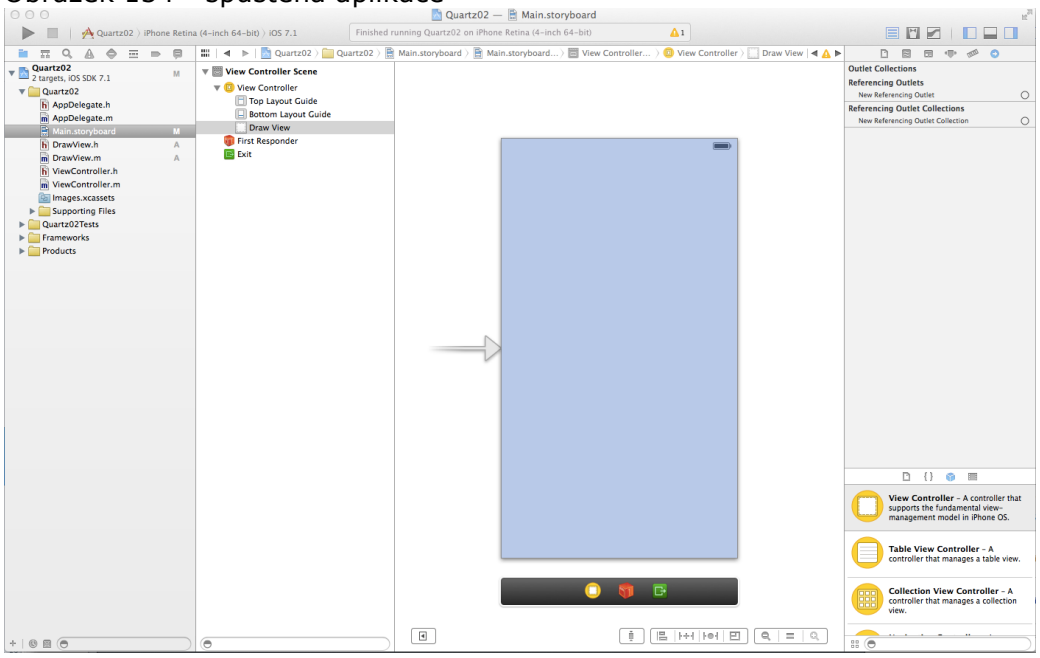
```
//  DrawView.h
#import <UIKit/UIKit.h>

@interface DrawView : UIView
```

```
@end

//  DrawView.m
#import "DrawView.h"

@implementation DrawView

- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code
    }
    return self;
}


// Only override drawRect: if you perform custom drawing.
// An empty implementation adversely affects performance
during animation.
- (void)drawRect:(CGRect)rect
{
    // Drawing code
    CGContextRef context = UIGraphicsGetCurrentContext();
    //line
    CGContextSetLineWidth(context, 7.0);
    CGColorSpaceRef colorSpace =
CGColorSpaceCreateDeviceRGB();
    CGColorSpaceRelease(colorSpace);
    CGFloat components[] = {0.8,0.4,0.1,1.0};
    CGColorRef color = CGColorCreate(colorSpace,
components);

CGContextSetStrokeColorWithColor(context,color);//[UIColor
redColor].CGColor);

    CGContextMoveToPoint(context, 30, 30);
    CGContextAddLineToPoint(context,300, 300);
    CGContextStrokePath(context);

    //rectangle
    CGContextSetLineWidth(context, 2.0);
    CGFloat components2[] = {0.1,0.3,0.2,1.0};
    color = CGColorCreate(colorSpace, components2);
    CGContextSetStrokeColorWithColor(context,color);
    CGRect rectangle = CGRectMake(10,50, 200,400);
    CGContextAddRect(context, rectangle);
    CGContextStrokePath(context);

    //ellipse
    CGContextSetLineWidth(context, 1.0);
    CGFloat components3[] = {0.1,0.9,0.9,0.5};
    color = CGColorCreate(colorSpace, components3);
    CGContextSetStrokeColorWithColor(context,color);
    rectangle = CGRectMake(50,100, 230,430);
    CGContextAddEllipseInRect(context, rectangle);
```

```
    CGContextStrokePath(context);

    //fill ellipse
    CGContextSetFillColorWithColor(context, color);
    CGContextFillEllipseInRect(context, rectangle);
    CGContextSetLineWidth(context, 7.0);
    CGColorSpaceRelease(colorSpace);
    CGColorRelease(color);

    //

}


@end
```



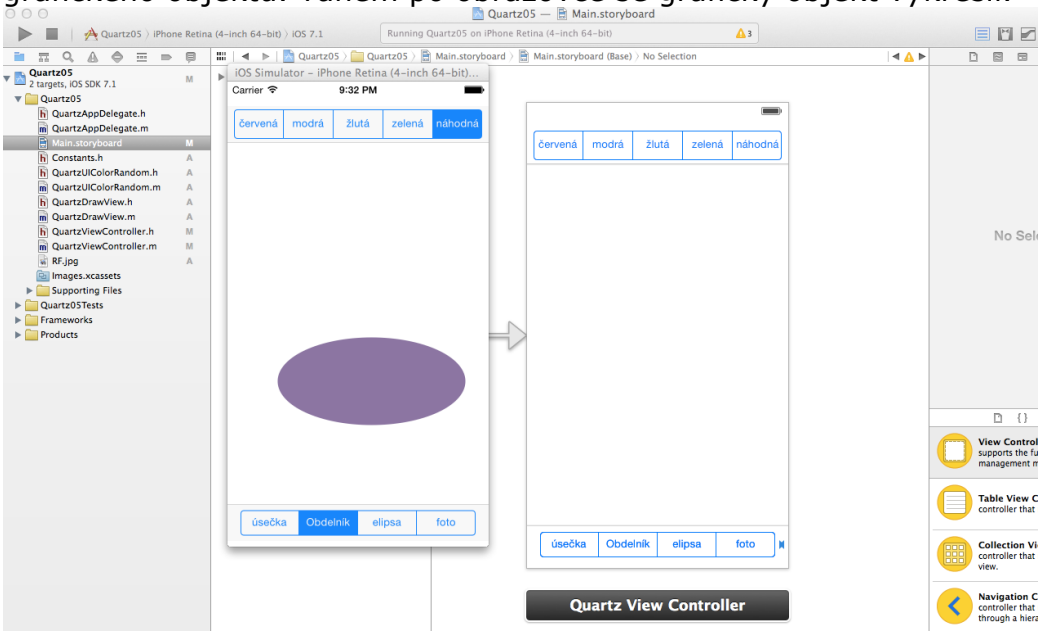
Obrázek 134 - spuštěná aplikace



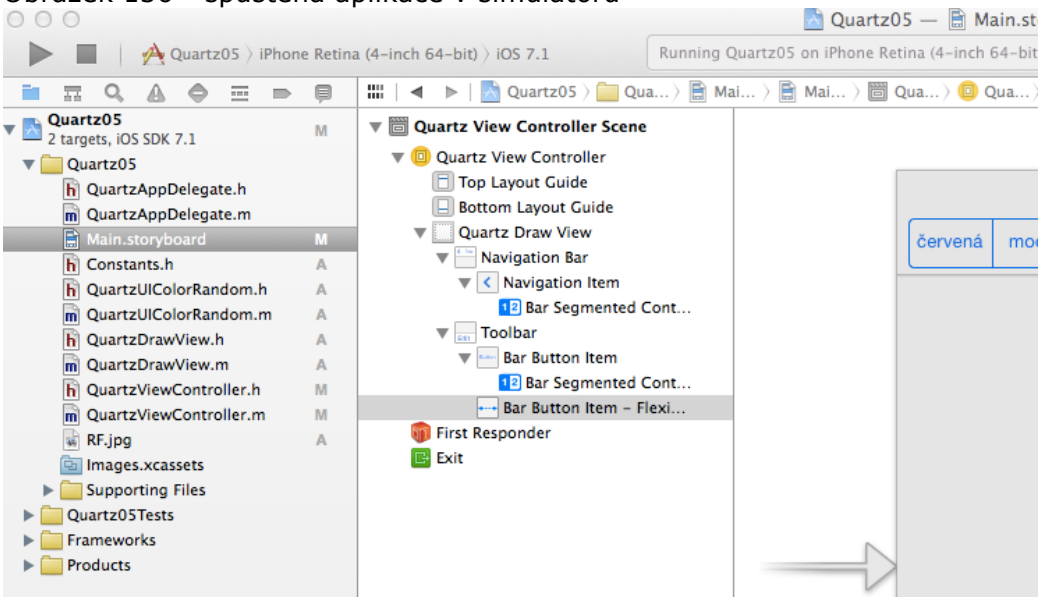
Obrázek 135 - prostředí Xcode

Příklad

Vytvořme aplikaci, která umožní vybrat barvu a druh grafického objektu. Tahem po obrazovce se grafický objekt vykreslí.

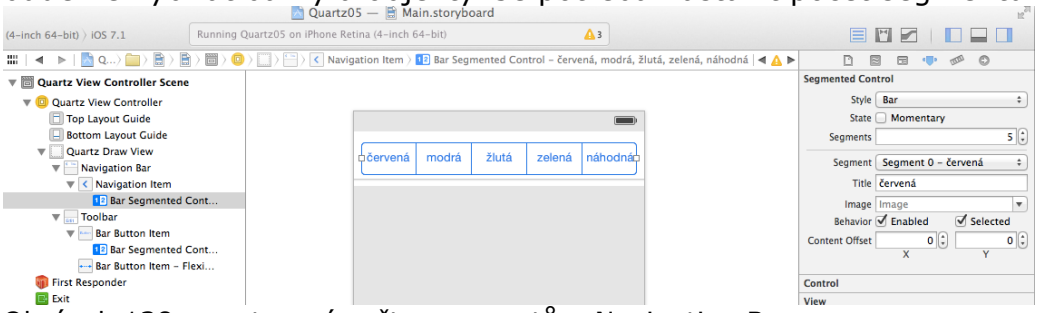


Obrázek 136 - spuštěná aplikace v simulátoru

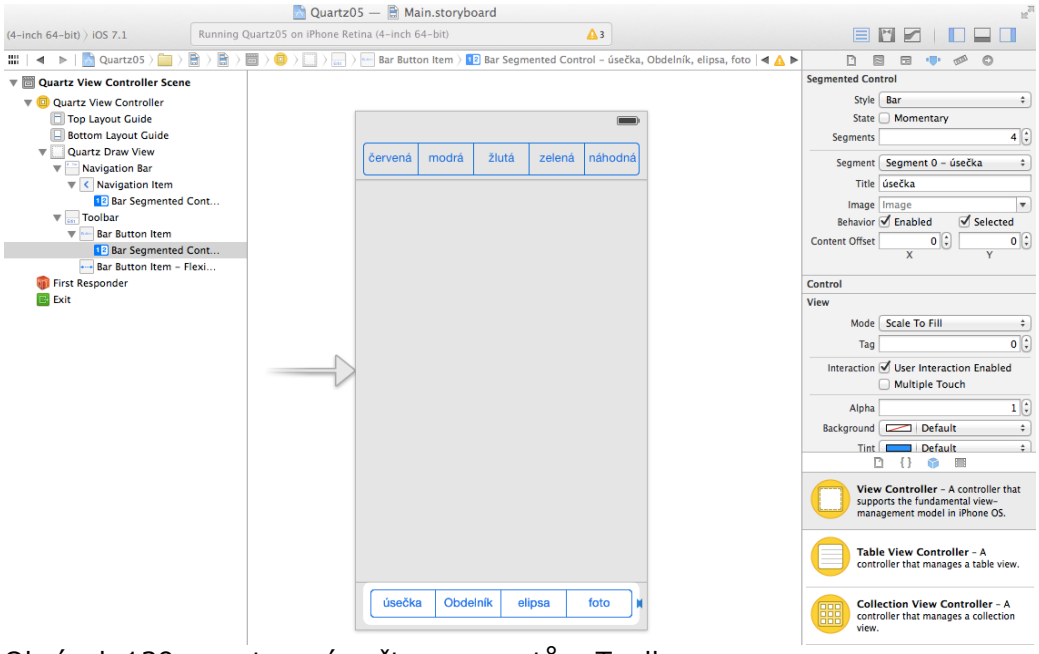


Obrázek 137 - části aplikace

Do aplikace přidáme Navigator Bar a Toolbar, ve kterých budeme vybírat barvy a objekty. Je potřeba nastavit počet segmentů.



Obrázek 138 - nastavení počtu segmentů u Navigation Bar



Obrázek 139 - nastavení počtu segmentů u Toolbaru

Zdrojové kódy aplikace:

```
//  Constants.h
typedef enum {
    kLineShape  = 0,
    kRectShape,
    kEllipseShape,
    kImageShape
} ShapeType;

typedef enum {
    kRedColorTab = 0,
    kBlueColorTab,
    kYellowColorTab,
    kGreenColorTab,
    kRandomColorTab
} ColorTabIndex;
#define degreesToRadian(x) (M_PI * (x) / 180.0)

//  QuartzUIColorRandom.h
#import <UIKit/UIKit.h>

@interface UIColor(Random)
+(UIColor *)randomColor;
@end

//  QuartzUIColorRandom.m
#import "QuartzUIColorRandom.h"

@implementation UIColor(Random)
+(UIColor *)randomColor
{
    static BOOL seeded = NO;
    if (!seeded) {
        seeded = YES;
```

```
        srandom(time(NULL));
    }
    CGFloat red =  (CGFloat)random()/(CGFloat)RAND_MAX;
    CGFloat blue = (CGFloat)random()/(CGFloat)RAND_MAX;
    CGFloat green = (CGFloat)random()/(CGFloat)RAND_MAX;
    return [UIColor colorWithRed:red green:green blue:blue
alpha:1.0f];
}
@end

//  QuartzDrawView.h
#import <UIKit/UIKit.h>
#import "Constants.h"

@interface QuartzDrawView : UIView
{
    CGPoint         firstTouch;
    CGPoint         lastTouch;
    UIColor         *currentColor;
    ShapeType       shapeType;
    UIImage         *drawImage;
    BOOL            useRandomColor;
    CGRect          redrawRect;
}
@property CGPoint firstTouch;
@property CGPoint lastTouch;
@property (nonatomic, retain) UIColor *currentColor;
@property ShapeType shapeType;
@property (nonatomic, retain) UIImage *drawImage;
@property BOOL useRandomColor;
@property CGRect redrawRect;
@end

//  QuartzDrawView.m
#import "QuartzDrawView.h"
#import "QuartzUIColorRandom.h"

@implementation QuartzDrawView
@synthesize firstTouch;
@synthesize lastTouch;
@synthesize currentColor;
@synthesize shapeType;
@synthesize drawImage;
@synthesize useRandomColor;
@synthesize redrawRect;


- (CGRect)currentRect {
    return CGRectMake ((firstTouch.x > lastTouch.x) ?
lastTouch.x : firstTouch.x,
                       (firstTouch.y > lastTouch.y) ?
lastTouch.y : firstTouch.y,
                       fabsf(firstTouch.x - lastTouch.x),
                       fabsf(firstTouch.y - lastTouch.y));
}
```

```
- (id)initWithCoder:(NSCoder*)coder
{
    if ( ( self = [super initWithCoder:coder] ) ) {
        self.currentColor = [UIColor redColor];
        self.useRandomColor = NO;
        if (drawImage == nil)
            self.drawImage = [UIImage
imageNamed:@"RF.jpg"];
    }
    return self;
}

- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code
    }
    return self;
}
- (void)drawRect:(CGRect)rect {

    CGContextRef context = UIGraphicsGetCurrentContext();

    CGContextSetLineWidth(context, 4.0);
    CGContextSetStrokeColorWithColor(context,
currentColor.CGColor);
    CGContextSetFillColorWithColor(context,
currentColor.CGColor);
    CGRect currentRect = CGRectMake ((firstTouch.x >
lastTouch.x) ? lastTouch.x : firstTouch.x, (firstTouch.y >
lastTouch.y) ? lastTouch.y :
firstTouch.y,fabsf(firstTouch.x - lastTouch.x),
fabsf(firstTouch.y - lastTouch.y));

    switch (shapeType) {
        case kLineShape:
            CGContextMoveToPoint(context, firstTouch.x,
firstTouch.y);
            CGContextAddLineToPoint(context, lastTouch.x,
lastTouch.y);
            CGContextStrokePath(context);
            break;
        case kRectShape:
            CGContextAddRect(context, currentRect);
            CGContextDrawPath(context, kCGPathFillStroke);
            break;
        case kEllipseShape:
            CGContextAddEllipseInRect(context,
currentRect);
            CGContextDrawPath(context, kCGPathFillStroke);
            break;
        case kImageShape:
        {
            CGFloat horizontalOffset =
drawImage.size.width / 2;
```

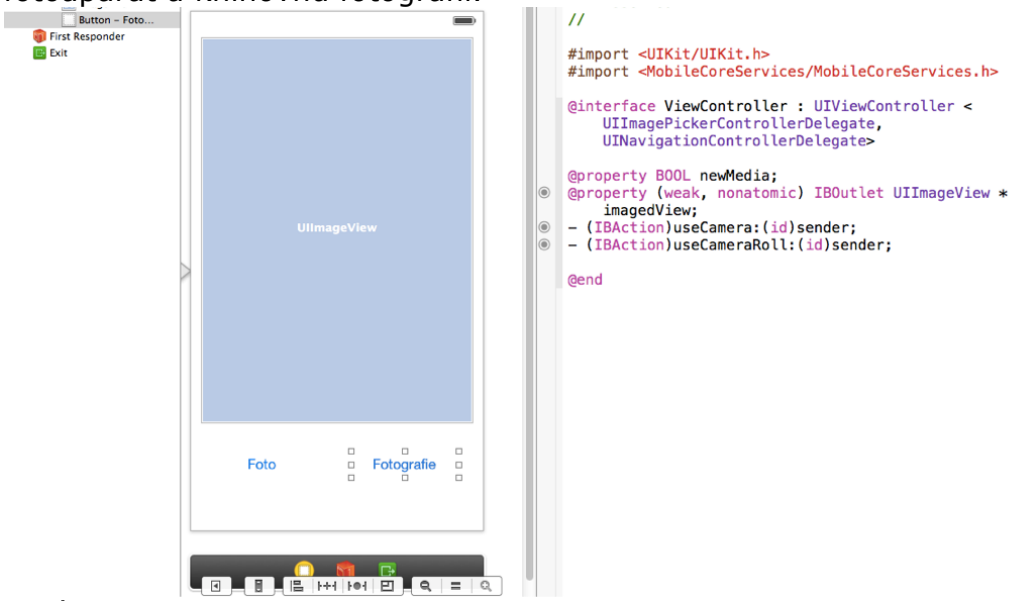
```
            CGFloat verticalOffset = drawImage.size.height
/ 2;
            CGPoint drawPoint = CGPointMake(lastTouch.x -
horizontalOffset, lastTouch.y - verticalOffset);
            [drawImage drawAtPoint:drawPoint];
            break;
        }
        default:
            break;
    }
}
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent
*)event {
    if (useRandomColor)
        self.currentColor = [UIColor randomColor];
    UITouch *touch = [touches anyObject];
    firstTouch = [touch locationInView:self];
    lastTouch = [touch locationInView:self];
    [self setNeedsDisplay];
}
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent
*)event {
    UITouch *touch = [touches anyObject];
    lastTouch = [touch locationInView:self];

    if (shapeType == kImageShape) {
        CGFloat horizontalOffset =
drawImage.size.width/2.0;
        CGFloat verticalOffset =
drawImage.size.height/2.0;
        redrawRect = CGRectUnion(redrawRect,
CGRectMake(lastTouch.x - horizontalOffset, lastTouch.y -
verticalOffset, drawImage.size.width,
drawImage.size.height));
    }
    else
        redrawRect = CGRectUnion(redrawRect,
self.currentRect);

    redrawRect = CGRectInset(redrawRect, -2.0, -2.0);
    [self setNeedsDisplayInRect:redrawRect];

}
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent
*)event {
    UITouch *touch = [touches anyObject];
    lastTouch = [touch locationInView:self];

    if (shapeType == kImageShape) {
        CGFloat horizontalOffset =
drawImage.size.width/2.0;
        CGFloat verticalOffset =
drawImage.size.height/2.0;
        redrawRect = CGRectUnion(redrawRect,
CGRectMake(lastTouch.x - horizontalOffset, lastTouch.y -
```

```
verticalOffset, drawImage.size.width,
drawImage.size.height));
    }
    else
        redrawRect = CGRectUnion(redrawRect,
self.currentRect);
    redrawRect = CGRectInset(redrawRect, -2.0, -2.0);
    [self setNeedsDisplayInRect:redrawRect];
}

@end

//  QuartzViewController.h
#import <UIKit/UIKit.h>

@interface QuartzViewController : UIViewController
{
    UISegmentedControl *colorControl;
}
@property (nonatomic, retain) IBOutlet UISegmentedControl
*colorControl;
- (IBAction)changeColor:(id)sender;
- (IBAction)changeShape:(id)sender;

@end

//  QuartzViewController.m
#import "QuartzViewController.h"
#import "QuartzDrawView.h"
#import "QuartzUIColorRandom.h"
#import "Constants.h"


@interface QuartzViewController ()

@end

@implementation QuartzViewController
@synthesize colorControl;
/*
- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
*/
- (IBAction)changeColor:(id)sender {
    UISegmentedControl *control = sender;
    NSInteger index = [control selectedSegmentIndex];
```

```
    QuartzDrawView *quartzView = (QuartzDrawView
*)self.view;

    switch (index) {
        case kRedColorTab:
            quartzView.currentColor = [UIColor redColor];
            quartzView.useRandomColor = NO;
            break;
        case kBlueColorTab:
            quartzView.currentColor = [UIColor blueColor];
            quartzView.useRandomColor = NO;
            break;
        case kYellowColorTab:
            quartzView.currentColor = [UIColor
yellowColor];
            quartzView.useRandomColor = NO;
            break;
        case kGreenColorTab:
            quartzView.currentColor = [UIColor
greenColor];
            quartzView.useRandomColor = NO;
            break;
        case kRandomColorTab:
            quartzView.useRandomColor = YES;
            break;
        default:
            break;
    }
}
- (IBAction)changeShape:(id)sender {
    UISegmentedControl *control = sender;
    [(QuartzDrawView *)self.view setShapeType:[control

selectedSegmentIndex]];

    if ([control selectedSegmentIndex] == kImageShape)
        colorControl.hidden = YES;
    else
        colorControl.hidden = NO;
}

- (void)didReceiveMemoryWarning {
      // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

      // Release any cached data, images, etc that aren't
in use.
}

- (void)viewDidUnload {
      // Release any retained subviews of the main view.
      // e.g. self.myOutlet = nil;
    self.colorControl = nil;
}
@end
```

Práce s fotografiemi

Následující příklad ukazuje jednoduchou aplikací využívající fotoaparát a knihovnu fotografií.



Obrázek 140 – aplikace

Kód využití fotoaparátu:

```
//  ViewController.h
#import <UIKit/UIKit.h>
#import <MobileCoreServices/MobileCoreServices.h>

@interface ViewController : UIViewController
<UIImagePickerControllerDelegate,
UINavigationControllerDelegate>

@property BOOL newMedia;
@property (weak, nonatomic) IBOutlet UIImageView
*imagedView;
- (IBAction)useCamera:(id)sender;
- (IBAction)useCameraRoll:(id)sender;

@end

//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize newMedia,imagedView;

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
}
```

```
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)useCamera:(id)sender
{
    if ([UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerSourceTypeCam
era])
    {
        UIImagePickerController *imagePicker =
                        [[UIImagePickerController alloc]
init];
        imagePicker.sourceType =

     UIImagePickerControllerSourceTypeCamera;
        imagePicker.mediaTypes = @[(NSString *)
kUTTypeImage];
        imagePicker.allowsEditing = NO;
        [self presentViewController:imagePicker
animated:YES
                        completion:nil];
        newMedia = YES;
    }
}
- (IBAction)useCameraRoll:(id)sender
{
    if ([UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerSourceTypeSav
edPhotosAlbum])
    {
        UIImagePickerController *imagePicker =
[[UIImagePickerController
                                alloc] init];
        imagePicker.sourceType =

     UIImagePickerControllerSourceTypePhotoLibrary;
        imagePicker.mediaTypes = @[(NSString *)
kUTTypeImage];
        imagePicker.allowsEditing = NO;
        [self presentViewController:imagePicker
animated:YES
                                completion:nil];
        newMedia = NO;
    }
}
-(void)imagePickerController:(UIImagePickerController
*)picker didFinishPickingMediaWithInfo:(NSDictionary
*)info
{
    NSString *mediaType =
info[UIImagePickerControllerMediaType];
```

```
    [self dismissViewControllerAnimated:YES
completion:nil];
    if ([mediaType isEqualToString:(NSString*)
kUTTypeImage])
    {
        UIImage *image =
info[UIImagePickerControllerOriginalImage];
        imagedView.image = image;
        if (newMedia)
UIImageWriteToSavedPhotosAlbum(image, self,

     @selector(image:finishedSavingWithError:
contextInfo:), nil);
    }
    else
        if ([mediaType isEqualToString:(NSString
*)kUTTypeMovie])
        {
            //...
        }
}

-(void) imagePickerControllerDidCancel :
(UIImagePickerController *)picker
{
    [self dismissViewControllerAnimated:YES
completion:nil];
}
```

Shrnutí kapitoly

Většina aplikací si vystačí pro vykreslení grafických komponent s frameworkem UIKit. Pro některé aplikace potřebují vykreslit vlastní grafické části. K tomu lze využít dvě knihovny:

- Quartz 2D
- OpenGL ES

14. SPRITKIT FRAMEWORK

V této kapitole se dozvíte:

Cílem této lekce ukázat možnosti frameworku SpritKit pro tvorbu her.

Po absolvování lekce budete:

- umět pracovat s frameworkem SpritKit

Klíčová slova této kapitoly:
SpritKit, 2D grafika, hry

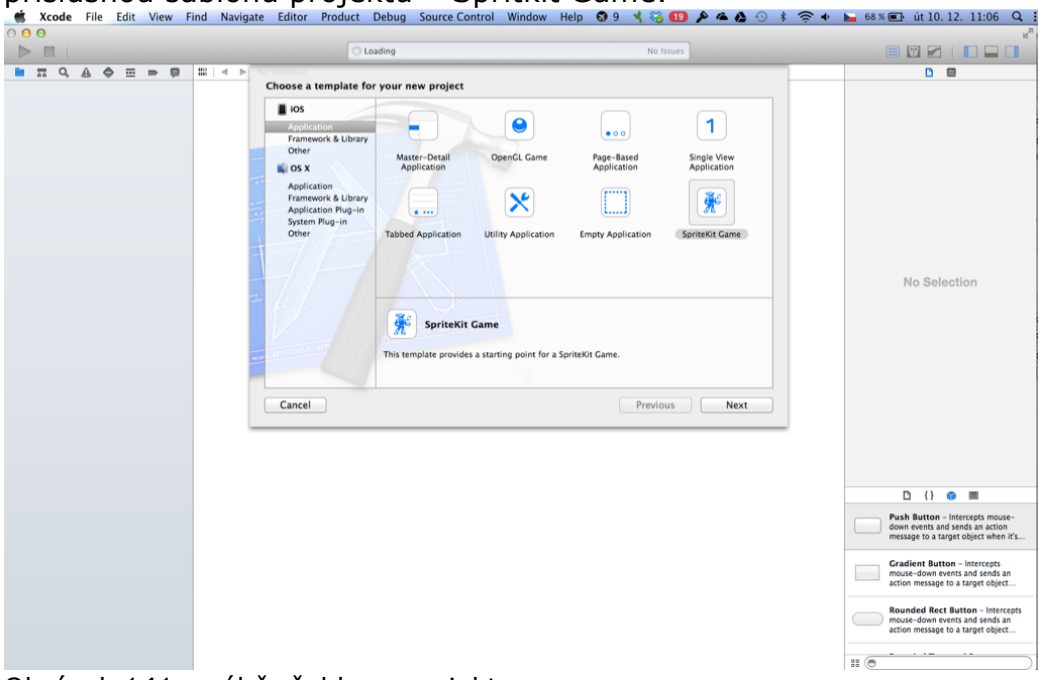
Čas ke studiu: 2 hodiny

Pro tvorbu 2D her v iOS nebo Mac OS X je možné využít framework SpritKit, který obsahuje následující elementy:

- Sprit Kit View
- Scenes
- Nodes – s třídami SKSpriteNode, SKLabelNode, SKShapeNode, SKEmitterNode, SKVideoNode, SKEffectNode, SKCropNode
- Physics Bodies
- Physics World
- Actions
- Transitions
- Texture Atlas

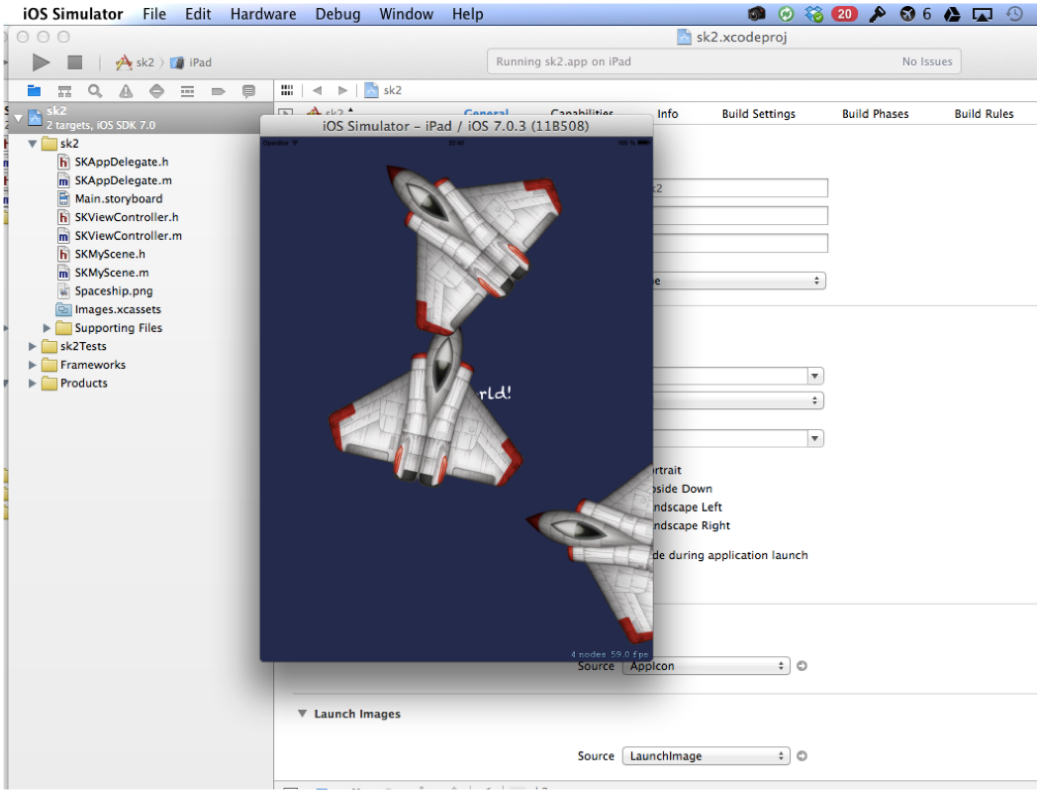
Příklad

V následujícím příkladu je jednoduchá ukázka využití SpritKit frameworku. Při zakládání nového projektu je potřeba vybrat příslušnou šablonu projektu – Spritkit Game.

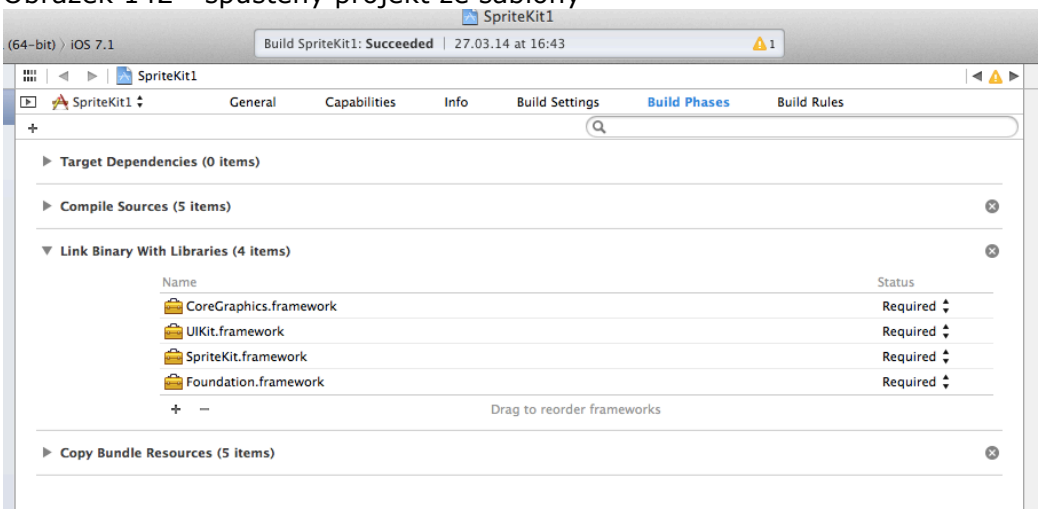


Obrázek 141 - výběr šablony projektu

Šablona již obsahuje ukázkový program, ve kterém můžeme částečně ovládat otáčení letadel.

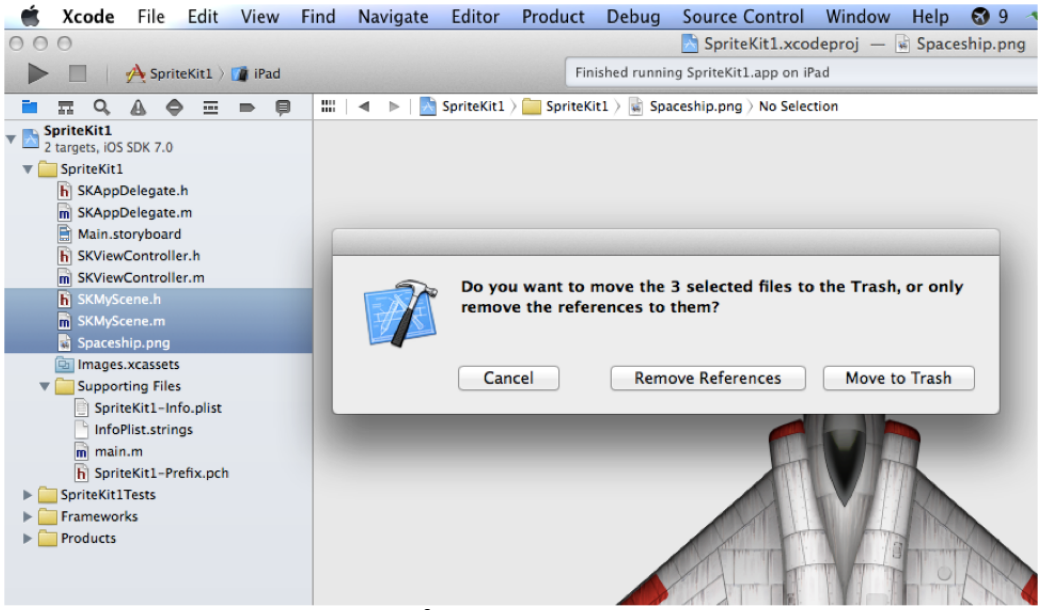


Obrázek 142 - spuštěný projekt ze šablony



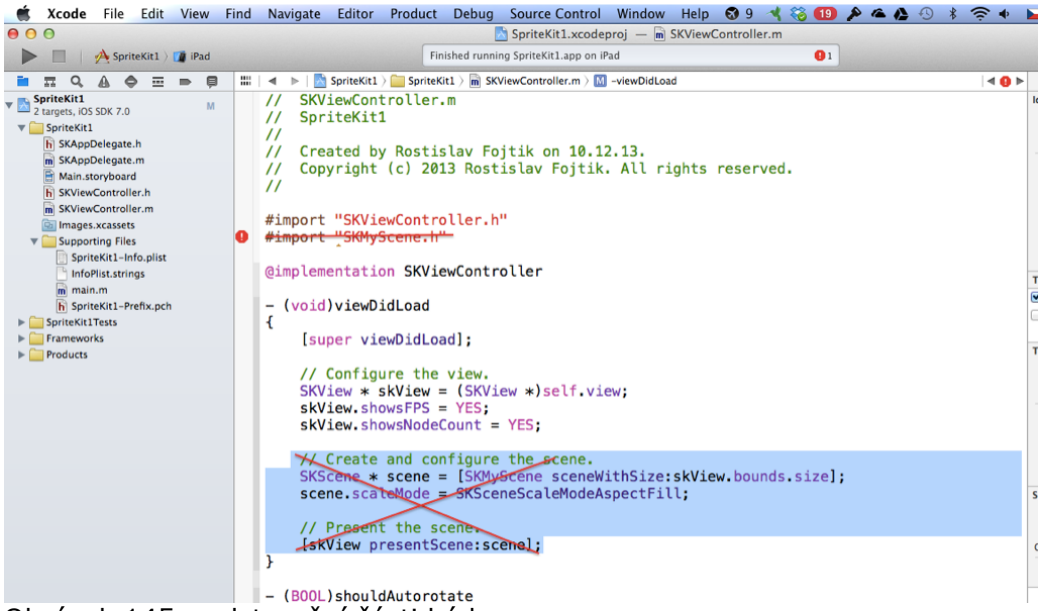
Obrázek 143 - součásti projektu je framework SpritKit

Vytváříme-li vlastní hru, je potřeba nejprve vyřadit z projektu ukázkové kódy. Je potřeba odebrat soubory: SKMyScene.h, SKMyScene.m, Spaceship.png.



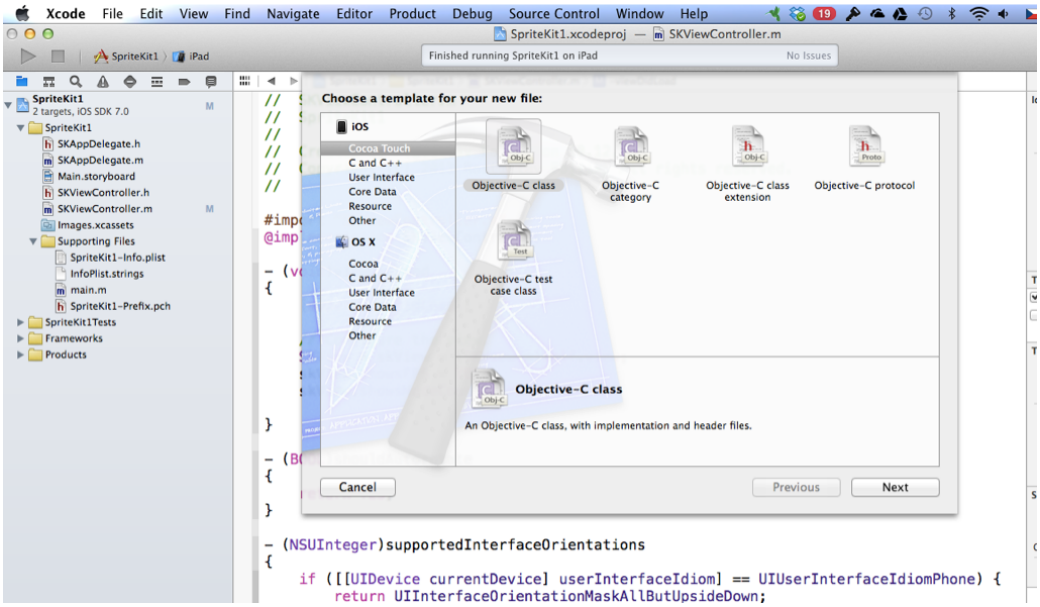
Obrázek 144 - vyřazení souborů z projektu

Dále je potřeba odstranit některé části kódu v souboru SKViewController.m.

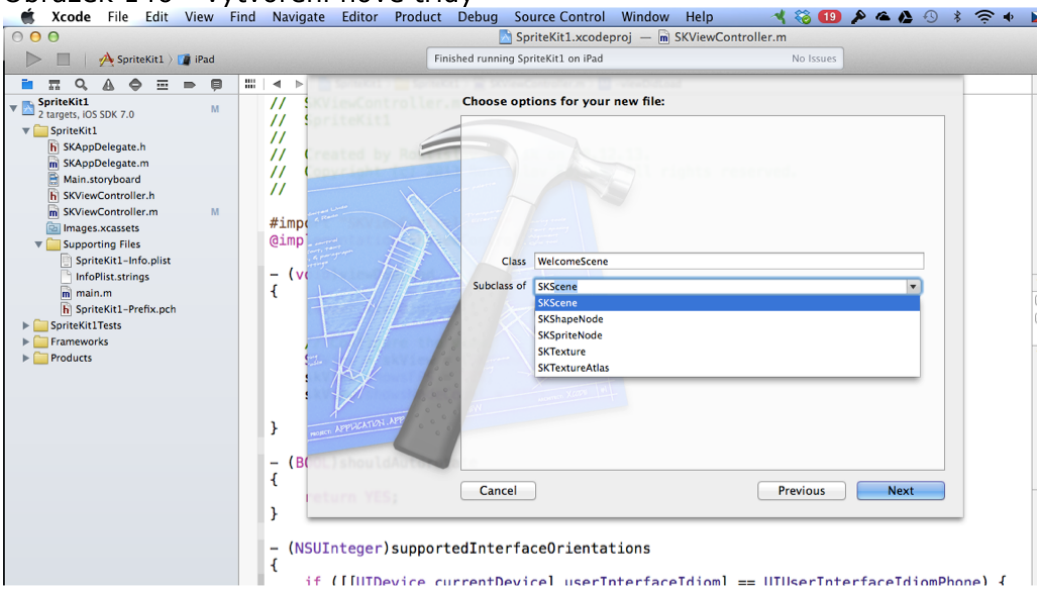


Obrázek 145 - odstranění části kódu

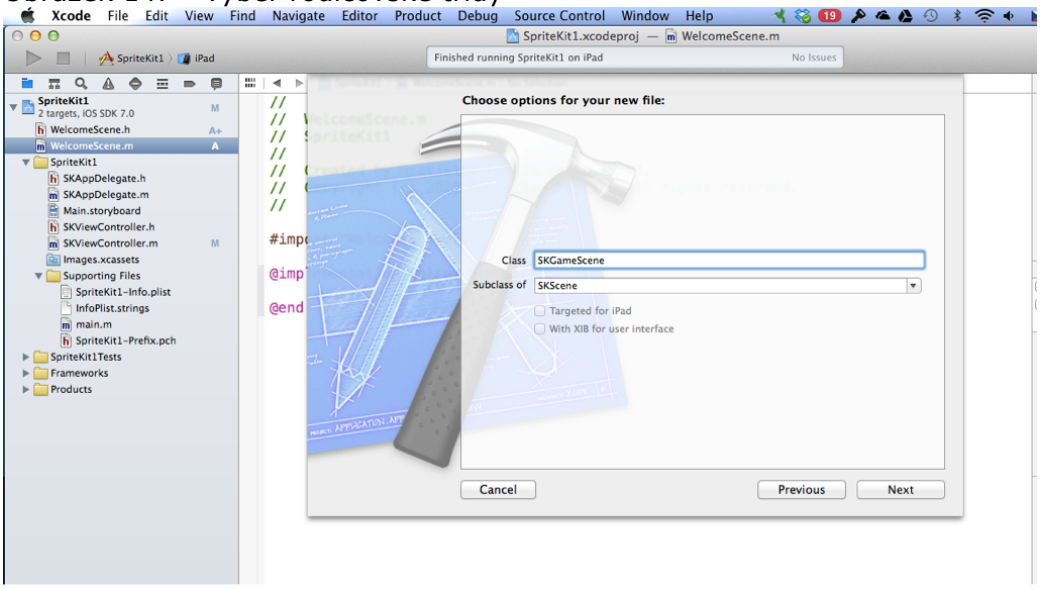
Následně vytvoříme novou třídu pro práci se scénou hry. Třída bude potomkem třídy SKScene.



Obrázek 146 - vytvoření nové třídy



Obrázek 147 - výběr rodičovské třídy

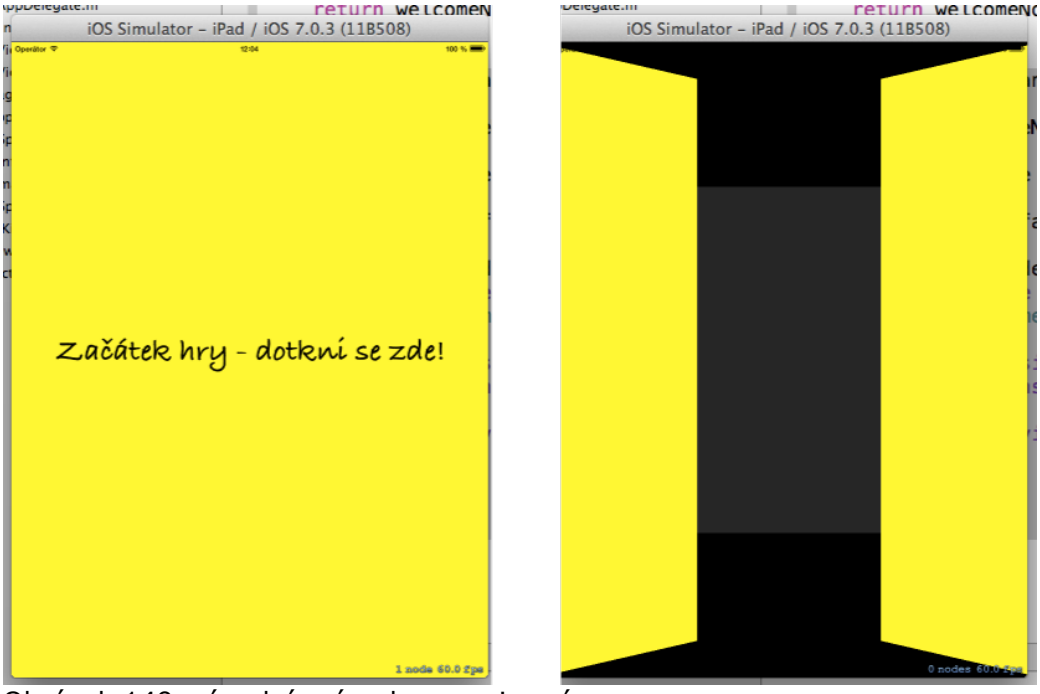


Obrázek 148 - pojmenování nové třídy

Následují kódy jsou pro úvodní scénu:

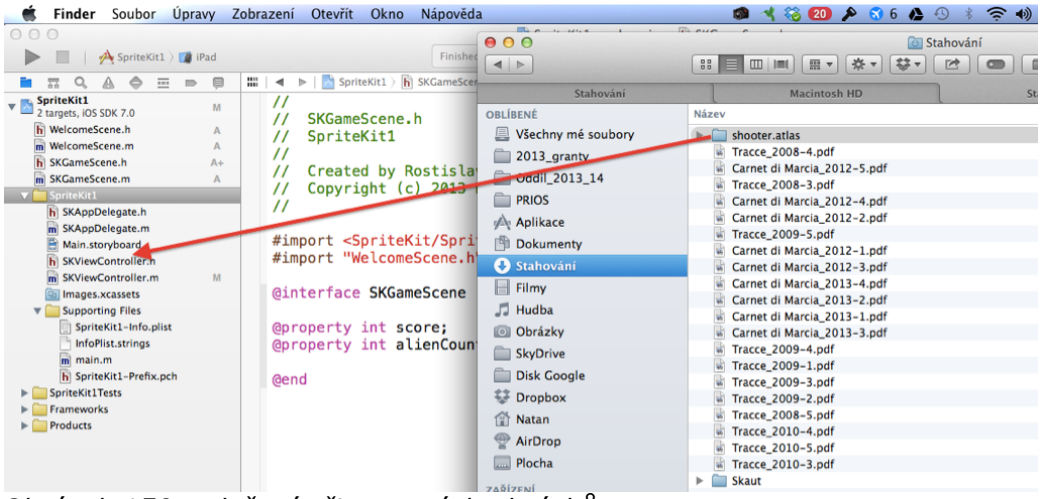
```
//  WelcomeScene.h
#import <SpriteKit/SpriteKit.h>

@interface WelcomeScene : SKScene

@end

//  WelcomeScene.m
#import "WelcomeScene.h"
#import "SKGameScene.h"

@interface WelcomeScene ()
@property BOOL sceneCreated;
@end

@implementation WelcomeScene

- (void) didMoveToView:(SKView *)view
{
    if (!self.sceneCreated)
    {
        self.backgroundColor = [SKColor yellowColor];
        self.scaleMode = SKSceneScaleModeAspectFill;
        [self addChild: [self createWelcomeNode]];
        self.sceneCreated = YES;
    }
}

- (SKLabelNode *) createWelcomeNode
{
    SKLabelNode *welcomeNode =
    [SKLabelNode labelNodeWithFontNamed:@"Bradley Hand"];

    welcomeNode.name = @"welcomeNode";
    welcomeNode.text = @"Začátek hry";
    welcomeNode.fontSize = 50;
    welcomeNode.fontColor = [SKColor blackColor];

    welcomeNode.position =
    CGPointMake(CGRectGetMidX(self.frame),
CGRectGetMidY(self.frame));

    return welcomeNode;
}

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent
*)event
{
    SKNode *welcomeNode = [self
childNodeWithName:@"welcomeNode"];

    if (welcomeNode != nil)
    {
```

```
        SKAction *fadeAway = [SKAction
fadeOutWithDuration:1.5];

        [welcomeNode runAction:fadeAway completion:^{
            SKScene *gameScene =
            [[SKGameScene alloc]initWithSize:self.size];

            SKTransition *doors =
            [SKTransition doorwayWithDuration:1.0];

            [self.view presentScene:gameScene
transition:doors];
        }
         ];
    }
}
@end
```

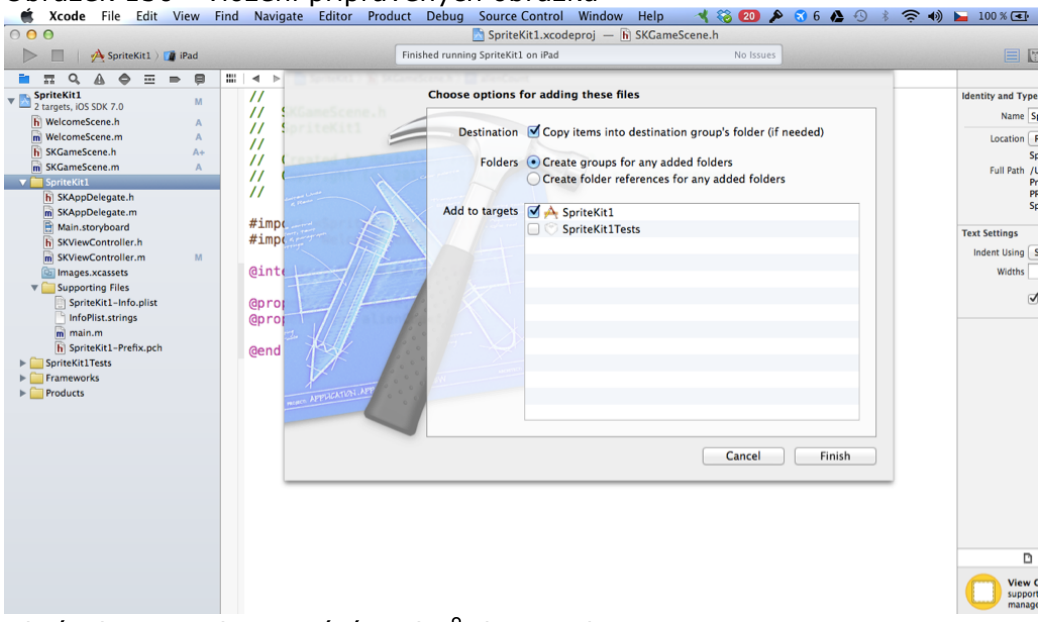


Obrázek 149 - úvodní scéna hry s animací

Do projektu si vložíme připravené obrázky pro animace ve hře. V našem ukázkovém příkladu se bude jednat o postavičku, která hází kamenem, a obrázek kamenu. Vše máme připravené v adresáři.



Obrázek 150 - vložení připravených obrázků



Obrázek 151 - zkopirování souboů do projektu

Kódy pro scénu hry, kterou jsme si vytvořili:

```
//  SKGameScene.h
#import <SpriteKit/SpriteKit.h>
#import "WelcomeScene.h"

@interface SKGameScene : SKScene
<SKPhysicsContactDelegate>

@end


//  SKGameScene.m
#import "SKGameScene.h"
#import "WelcomeScene.h"

@interface SKGameScene()

@property int score;
@property int alienCount;
@property NSArray *shooterAnimation;
```

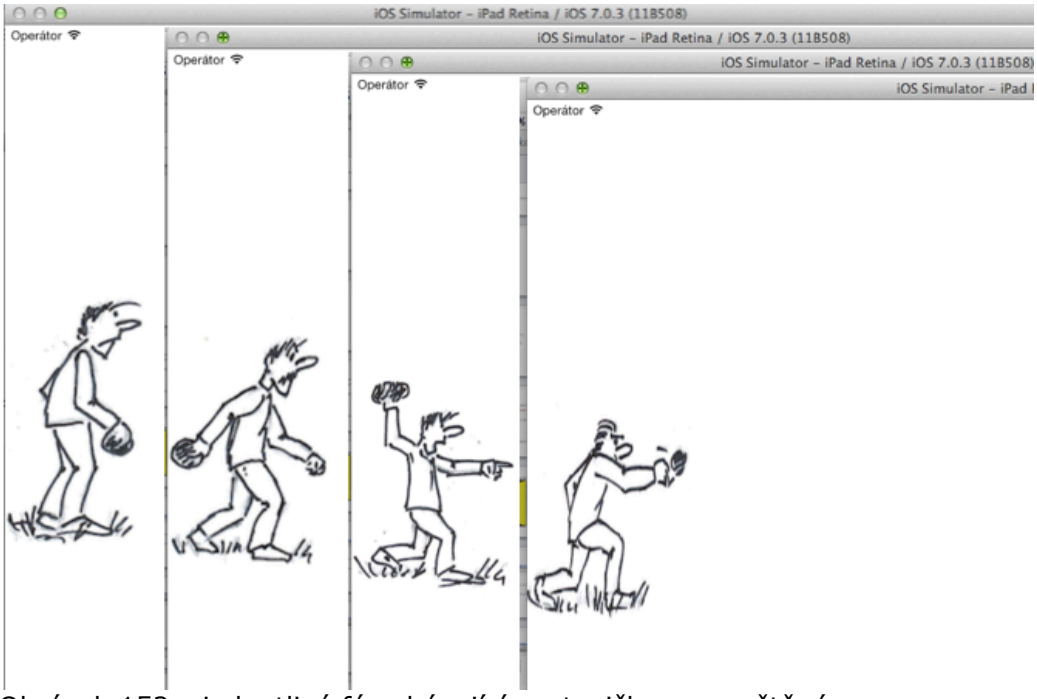
```
@property BOOL sceneCreated;
@property int ballCount;

@end

@implementation SKGameScene
static const uint32_t rockCategory = 0x1 << 0;
//static const uint32_t ballCategory = 0x1 << 1;
- (void)didMoveToView:(SKView *)view
{
    if (!self.sceneCreated)
    {
        self.score = 0;
  self.sceneCreated = YES;
    }
}

- (void) initSKGameScene
{
    self.backgroundColor = [SKColor whiteColor];
    self.scaleMode = SKSceneScaleModeAspectFill;

    SKSpriteNode *shooterNode = [self createShooterNode];

    shooterNode.position =
    CGPointMake(CGRectGetMinX(self.frame)+55,
                CGRectGetMidY(self.frame));

    [self addChild:shooterNode];

    NSMutableArray *gameFrames = [NSMutableArray array];

    SKTextureAtlas *shooterAtlas =
      [SKTextureAtlas atlasNamed:@"shooter"];

    for (int i = 1; i <= shooterAtlas.textureNames.count;
++i)
    {
        NSString *texture =
        [NSString stringWithFormat:@"shooter%d", i];

        [gameFrames addObject:[shooterAtlas
textureNamed:texture]];
    }

    self.shooterAnimation = gameFrames;

}

- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent
*)event
{
    SKNode *shooterNode = [self
childNodeWithName:@"shooterNode"];

    if (shooterNode != nil)
```

```
    {
        SKAction *animate = [SKAction

animateWithTextures:self.shooterAnimation
                                timePerFrame: 0.15];

        SKAction *shootRock = [SKAction runBlock:^{

            SKNode *rockNode = [self createRockNode];

            [self addChild:rockNode];
            [rockNode.physicsBody
applyImpulse:CGVectorMake(30.0, 0)];
        }];

        SKAction *sequence =
        [SKAction sequence:@[animate, shootRock]];



        [shooterNode runAction:sequence];
    }
}


- (SKSpriteNode *)createShooterNode
{
    SKSpriteNode *shooterNode =
    [[SKSpriteNode alloc]
initWithImageNamed:@"shooter1.png"];

    shooterNode.name = @"shooterNode";
    return shooterNode;
}

- (SKSpriteNode *) createRockNode
{
    SKSpriteNode *rock =
    [[SKSpriteNode alloc] initWithImageNamed:@"rock.png"];

    rock.position =
CGPointMake(CGRectGetMinX(self.frame)+60,

CGRectGetMidY(self.frame)+30);

    rock.name = @"rockNode";

    rock.physicsBody =
    [SKPhysicsBody
bodyWithRectangleOfSize:rock.frame.size];

    rock.physicsBody.usesPreciseCollisionDetection = YES;
    rock.physicsBody.categoryBitMask = rockCategory;

    return rock;
}
```

```
- (void) createBallNode
{
    SKSpriteNode *ball = [[SKSpriteNode alloc]

initWithImageNamed:@"BallTexture.png"];

    ball.position = CGPointMake(randomBetween(200,
self.size.width),
                                self.size.height-50);

    ball.name = @"ballNode";
    ball.physicsBody =
    [SKPhysicsBody
bodyWithCircleOfRadius:(ball.size.width/2)-7];

    ball.physicsBody.usesPreciseCollisionDetection = YES;
   // ball.physicsBody.categoryBitMask = ballCategory;

    [self addChild:ball];
}

- (void) didBeginContact:(SKPhysicsContact *)contact
{
    SKSpriteNode *firstNode, *secondNode;

    firstNode = (SKSpriteNode *)contact.bodyA.node;
    secondNode = (SKSpriteNode *) contact.bodyB.node;

    if (contact.bodyA.categoryBitMask == rockCategory)
     {
        CGPoint contactPoint = contact.contactPoint;

        float contact_y = contactPoint.y;
        float target_x = secondNode.position.x;

        float target_y = secondNode.position.y;
        float margin = secondNode.frame.size.height/2 -
25;

        if ((contact_y > (target_y - margin)) &&
            (contact_y < (target_y + margin)))
        {
            NSString *burstPath =
            [[NSBundle mainBundle]
             pathForResource:@"BurstParticle"
ofType:@"sks"];

            SKEmitterNode *burstNode =
            [NSKeyedUnarchiver
unarchiveObjectWithFile:burstPath];

            burstNode.position = CGPointMake(target_x,
target_y);

            [secondNode removeFromParent];
```

```
            [self addChild:burstNode];
            self.score++;
        }
    }
}

- (SKLabelNode *) createScoreNode
{
    SKLabelNode *scoreNode =
    [SKLabelNode labelNodeWithFontNamed:@"Bradley Hand"];

    scoreNode.name = @"scoreNode";

    NSString *newScore =
    [NSString stringWithFormat:@"Score: %i", self.score];

    scoreNode.text = newScore;
    scoreNode.fontSize = 60;
    scoreNode.fontColor = [SKColor redColor];

    scoreNode.position =
    CGPointMake(CGRectGetMidX(self.frame),
CGRectGetMidY(self.frame));

    return scoreNode;
}

- (void) gameOver
{
    SKLabelNode *scoreNode = [self createScoreNode];

    [self addChild:scoreNode];

    SKAction *fadeOut = [SKAction sequence:@[[SKAction

waitForDuration:3.0],
                                             [SKAction
fadeOutWithDuration:3.0]]];

    SKAction *welcomeReturn = [SKAction runBlock:^{

        SKTransition *transition =
        [SKTransition
revealWithDirection:SKTransitionDirectionDown
                                  duration:1.0];

        WelcomeScene *welcomeScene =
        [[WelcomeScene alloc] initWithSize:self.size];

        [self.scene.view presentScene: welcomeScene
                           transition:transition];
    }];

    SKAction *sequence = [SKAction sequence:@[fadeOut,
welcomeReturn]];
```

```
    [self runAction:sequence];
}

static inline CGFloat randomFloat()
{
    return rand() / (CGFloat) RAND_MAX;
}

static inline CGFloat randomBetween(CGFloat low, CGFloat
high)
{
    return randomFloat() * (high - low) + low;
}

@end
```



Obrázek 152 - jednotlivé fáze házející postavičky ve spuštěném programu

Shrnutí kapitoly

Pro tvorbu 2D her lze využít framework SpritKit, který usnadňuje vývoj aplikace.

15. PRÁCE S MÉDII A SOCIÁLNÍMI SÍTĚMI

V této kapitole se dozvíte:

Cílem této lekce ukázat možnosti využití médií a sociálních sítí v mobilních aplikacích.

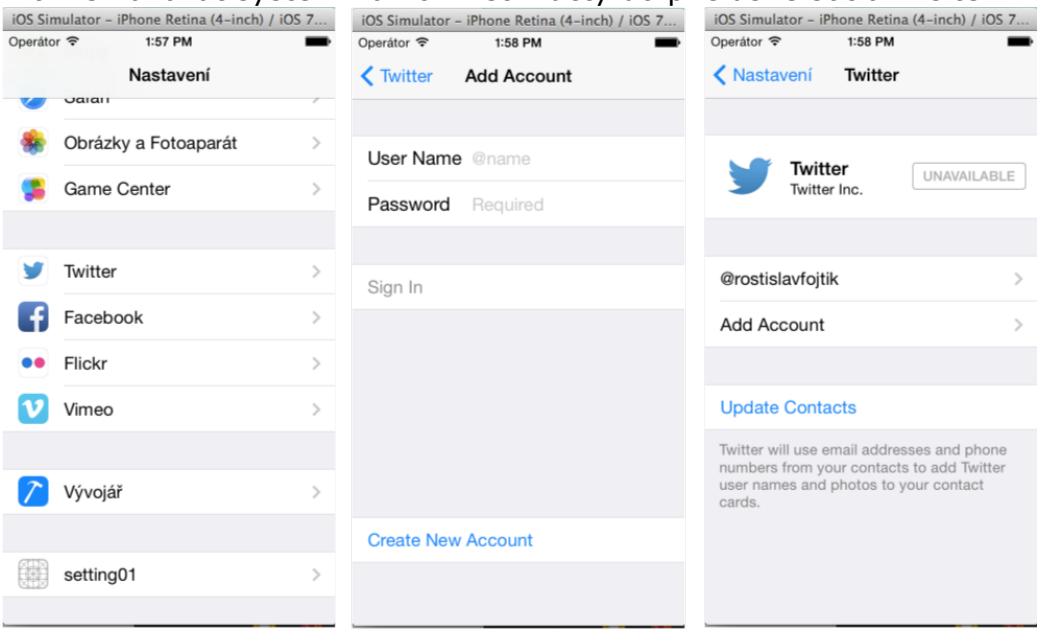
Po absolvování lekce budete:

- umět pracovat s fotoaparátem, audiem, knihovnami obrázků a zvuků
- vytvářet v aplikacích podporu pro práci se sociálními sítěmi

Klíčová slova této kapitoly:
audio, fotoaparát, sociální sítě, facebook, twitter

Čas ke studiu: 3 hodiny

Operační systém iOS nabízí přímou podporu pro práci se sociálními sítěmi typu Facebook a Twitter. V nastavení systému je možné navázat systém na konkrétní účty do příslušné sociální sítě.

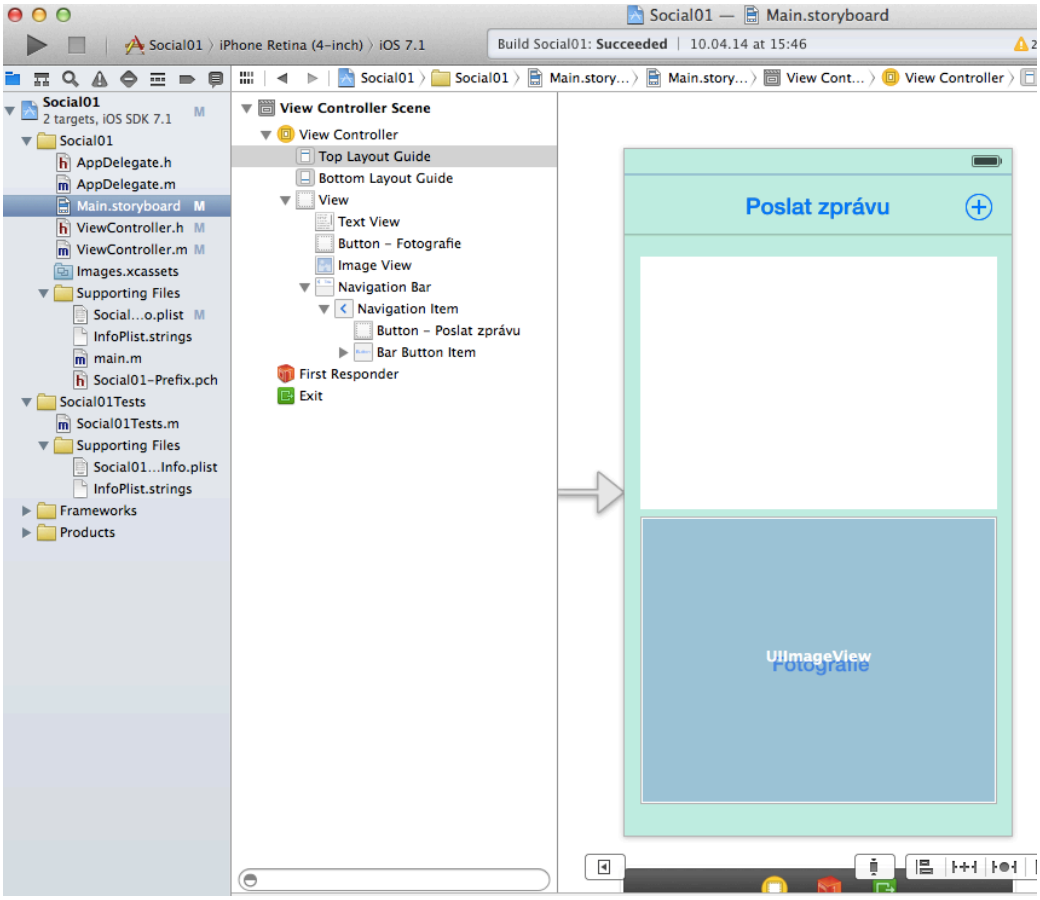


Obrázek 153 - podpora sociálních sítí v operačním systému iOS

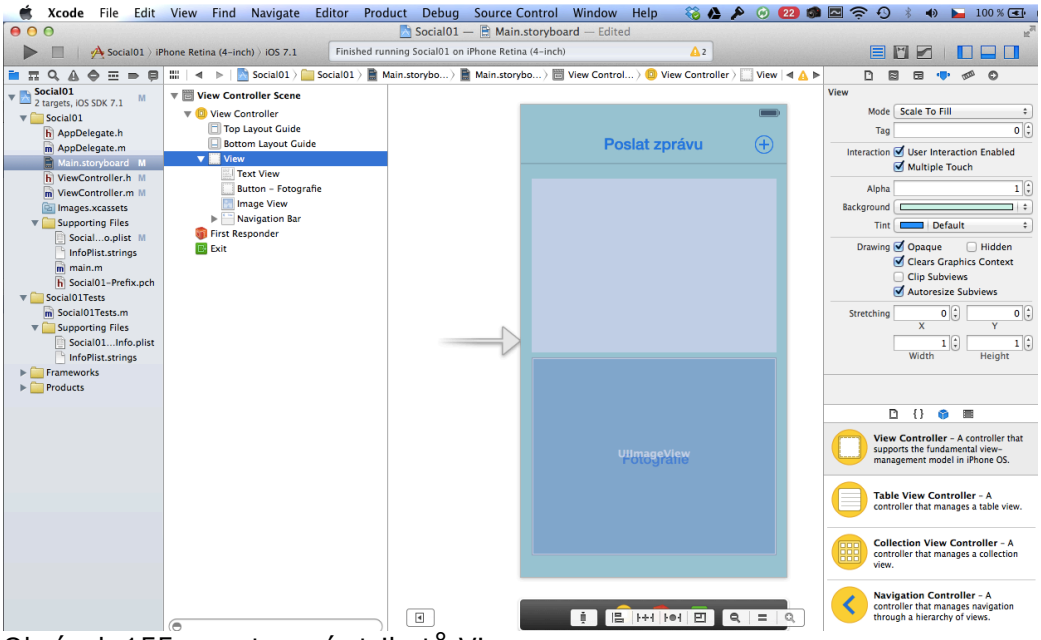
Příklad

Následující příklad ukazuje jednoduchou aplikaci, která umožní napsat zprávu, připojit k ní z knihovny obrázků fotografii a poslat do příslušné sociální sítě. Ve storyboardu si vytvořme vzhled aplikace, do které umístíme komponenty:

- TextView – pro zápis zprávy
- Button – pro vyvolání výběru fotografie
- Image View – pro zobrazení vybrané fotografie
- Navigation Bar, který bude obsahovat Button pro vytvoření nové zprávy a druhý Button pro odeslání vytvoření zprávy



Obrázek 154 - návrh vzhledu aplikace



Obrázek 155 - nastavení atributů View

Abychom mohli využívat sociální sítě a další posílání zprav, je potřeba připojit příslušné hlavičkové soubory:

```
#import <Social/Social.h>
#import <MobileCoreServices/MobileCoreServices.h>
```

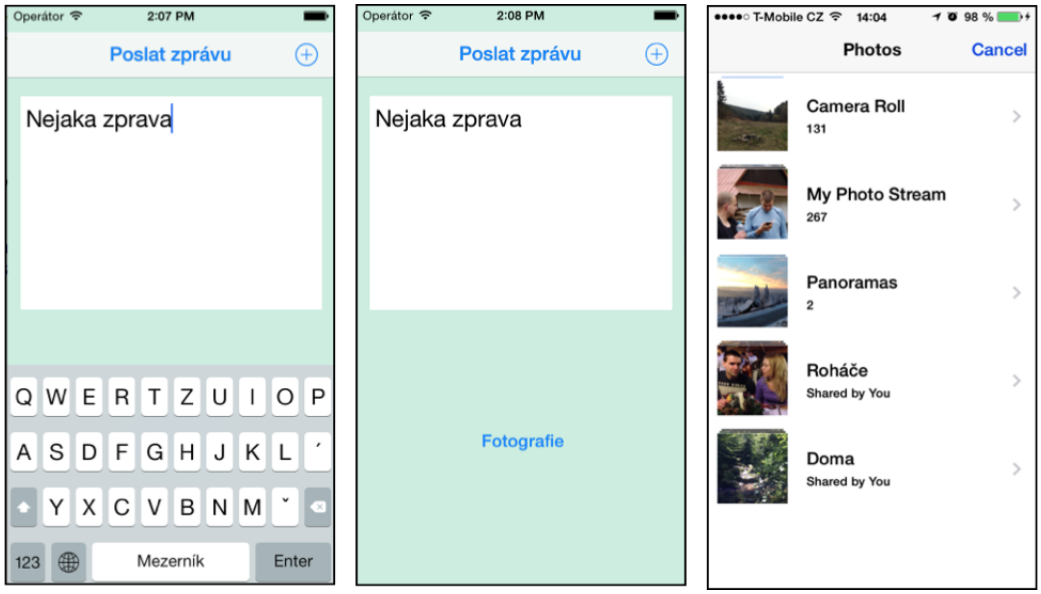
Kódy aplikace, které je nutné vytvořit:

```
//  ViewController.h
#import <UIKit/UIKit.h>
#import <Social/Social.h>
#import <MobileCoreServices/MobileCoreServices.h>

@interface ViewController : UIViewController
<UIImagePickerControllerDelegate,
UINavigationControllerDelegate>

@property (weak, nonatomic) IBOutlet UITextView *postText;
@property (strong, nonatomic) IBOutlet UIImageView
*postImage;

- (IBAction)sendPost:(id)sender;
- (IBAction)cancelAll:(id)sender;

@end

//  ViewController.m
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize postImage, postText;

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)selectedImage:(id)sender
{
    if ([UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerSourceTypeSav
edPhotosAlbum])
    {
        UIImagePickerController *imagePicker =
[[UIImagePickerController alloc] init];
        imagePicker.delegate = self;
        imagePicker.sourceType =
UIImagePickerControllerSourceTypePhotoLibrary;
        imagePicker.mediaTypes = @[(NSString*)
kUTTypeImage];
        imagePicker.allowsEditing = NO;
```

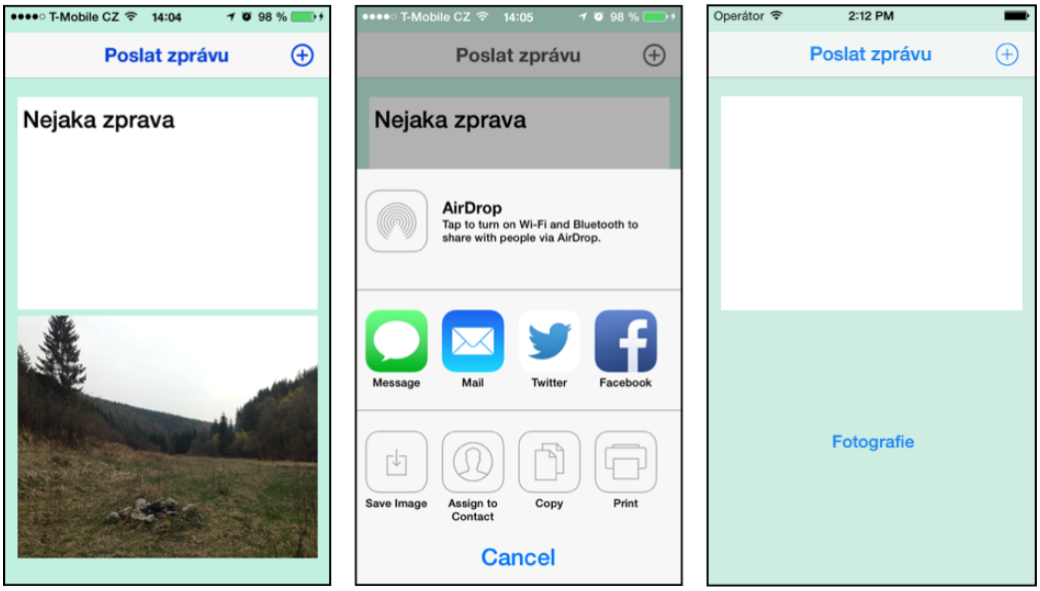
```
        [self presentViewController:imagePicker
animated:YES completion:nil];

    }
}

- (IBAction)sendPost:(id)sender
{
    NSArray *activityItems;

    if (postImage != nil)
    {
        activityItems = @[postText.text, postImage.image];
    }
    else
    {
     //   activityItems = @[postText.text];
     // UIAlertView *alert = [[UIAlertView
alloc]initWithTitle:@"Chyba" message:@"Chyba" delegate:nil
cancelButtonTitle:@"OK" otherButtonTitles: nil];
    }
    UIActivityViewController *activityController =
[[UIActivityViewController alloc]
initWithActivityItems:activityItems
applicationActivities:nil];
    [self presentViewController:activityController
animated:YES completion:nil];
}

- (IBAction)cancelAll:(id)sender
{
    postImage.image = nil;
    postText.text = @"";
}

-(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent
*)event
{
    UITouch *touch = [[event allTouches] anyObject];
    if ([postText isFirstResponder] && [touch view] !=
postText)
    {
        [postText resignFirstResponder];
    }
    [super touchesBegan:touches withEvent:event];
}

#pragma mark -
#pragma UIImagePickerControllerDelegate
-(void) imagePickerController:(UIImagePickerController
*)picker didFinishPickingMediaWithInfo:(NSDictionary
*)info
{
    NSString *mediaType =
info[UIImagePickerControllerMediaType];
```

```
    [self dismissViewControllerAnimated:YES
completion:nil];
    if ([mediaType isEqualToString:(NSString*)
kUTTypeImage])
    {
        UIImage *image =
info[UIImagePickerControllerOriginalImage];
        postImage.image = image;
    }
}
-(void)
imagePickerControllerDidCancel:(UIImagePickerController
*)picker
{
    [self dismissViewControllerAnimated:YES
completion:nil];
}
@end
```

Díky podpoře iOS nemusíme programově řešit fázi posílání zprávy. Po kliknutí na tlačítko *Poslat zprávu* se objeví standardní dialog pro výběr kam a pomocí kterých nástrojů se zpráva odešle.



Obrázek 156 - jednotlivé fáze spuštěné aplikace

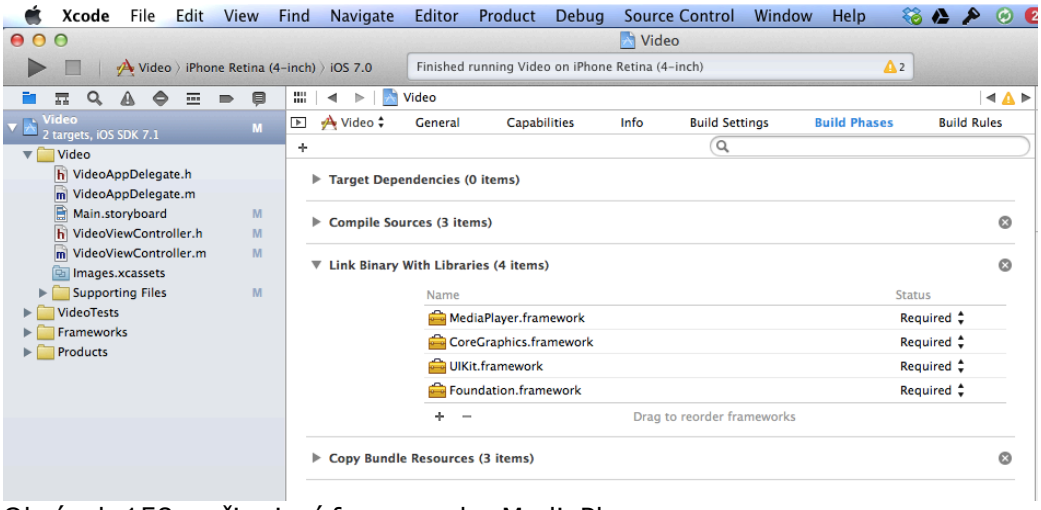


Obrázek 157 - jednotlivé fáze spuštěné aplikace

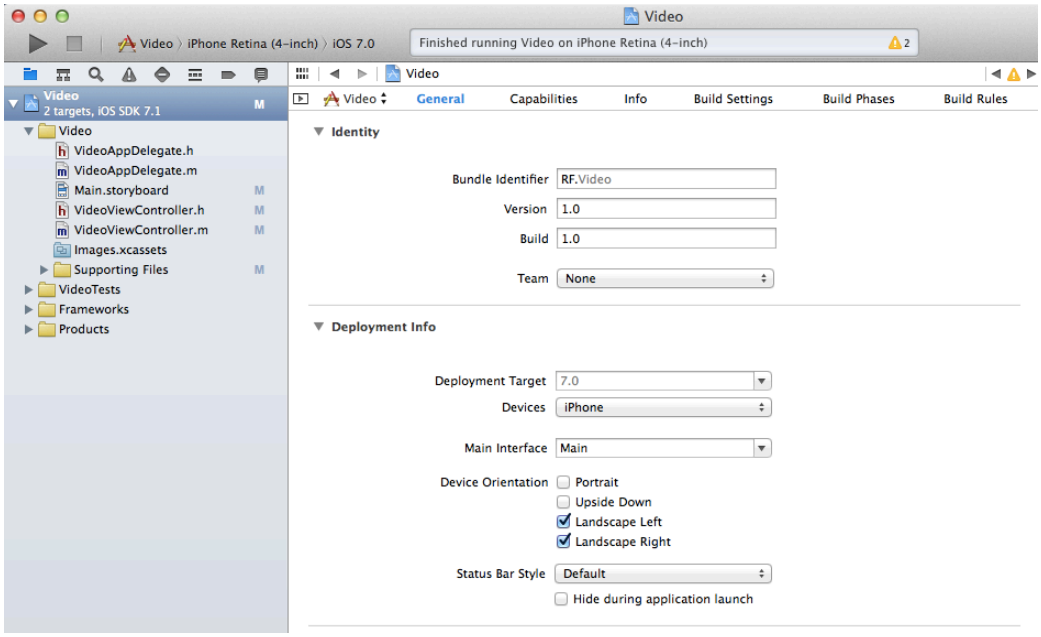
Příklad

Následující příklad ukazuje využití video přehrávače v mobilní aplikaci. Operační systém iOS a mobilní zařízení s ním pracující podporují následující typy videa s příponou .mov, .m4v, mpv a .3gpv s pomocí kompresе H.264 Baseline Profile Level 3.0 nebo MPEG-4 Part 2 video.

Po vytvoření nového projektu je potřeba nejprve připojit framework MediaPlayer do souboru VideoViewController přidáme hlavičkový soubor: #import <MediaPlayer/MediaPlayer.h>.



Obrázek 158 - připojení frameworku MediaPlayer



Obrázek 159 - u aplikace nastavíme pouze Landscape režim

```
//  VideoViewController.h
#import <UIKit/UIKit.h>
#import <MediaPlayer/MediaPlayer.h>

@interface VideoViewController : UIViewController

@property (strong, nonatomic) MPMoviePlayerController
*moviePlayer;
- (IBAction)playMovie:(id)sender;

@end

//  VideoViewController.m
#import "VideoViewController.h"

@interface VideoViewController ()

@end

@implementation VideoViewController
@synthesize moviePlayer;

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view,
typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

```
- (IBAction)playMovie:(id)sender
{
    NSURL *url = [NSURL
URLWithString:@"http://www.nejaka_adresa.cz/pokus.m4v"];
    moviePlayer = [[MPMoviePlayerController alloc]
initWithContentURL:url];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(moviePlayBackDidFinish)
name:MPMoviePlayerPlaybackDidFinishNotification
object:moviePlayer];
    moviePlayer.controlStyle = MPMovieControlStyleDefault;
    moviePlayer.shouldAutoplay = YES;

    [self.view addSubview:moviePlayer.view];
    [moviePlayer setFullscreen:YES animated:YES];
}

-(void)
moviePlayBackDidFinish:(NSNotification*)notification
{
    MPMoviePlayerController *player = [notification
object];
    [[NSNotificationCenter defaultCenter]
removeObserver:self
name:MPMoviePlayerPlaybackDidFinishNotification
object:player];
    if ([player
respondsToSelector:@selector(setFullscreen:animated:)])
    {
        [player.view removeFromSuperview];
    }
}
@end
```

Obrázek 160 - spuštěné video

KORESPONDENČNÍ ÚKOL

Vytvořte jednoduchou aplikaci pro iPhone, která bude sloužit jako převodník základních jednotek. Vyberte si alespoň 3 druhy jednotek SI. Vytvořený program umožní převod metrických jednotek například na angloamerický měrný systém.

Můžete například převádět délkové jednotky z metrů na míle, stopy a palce. Teploty ve stupních Celsia na Farada a Kelvína, hmotnost z kg na libry a podobně.

Využijte v aplikaci například ovládací prvek Text Field pro zadávání hodnot, Label pro výpis převedených hodnot, Picker View pro výběr konkrétní jednotky.

Aplikaci není potřeba optimalizovat pro různá rozlišení obrazovky a jiné verze iOS.

LITERATURA

[1] Mark, D., LaMarche, J.: *iPhone SDK, Průvodce vývojem aplikací pro iPhone a iPod touch*, Computer Press, Brno 2010, ISBN 978-80-251-2820-6

[2] Kochan, S. G.: *Objective-C 2.0, Výukový kurz programování pro Mac OS X a iPhone*, Computer Press, Brno 2010, ISBN 978-80-251-2654-7

[3] Nutting, J., Mark D., LaMarche J.: *Cocoa, Průvodce programováním pro Mac*, Computer Press, Brno 2011, ISBN 978-80-251-2883-1

[4] Alessi, P.: *Vývoj her pro iPhone a iPad*, Zoner Press, Brno 2012, ISBN 978-80-7413-199-8

[5] Čada, O.: *Cocoa, úvod do programování počítačů Apple*, Grada, Praha 2009, ISBN 978-80-247-2778-3

[6] *Object-Oriented Programming with Objective-C,*Apple Inc., iBooks. https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewBook?id=409922309, 2010

[7] *The Objective-C Programming Language*, Apple Inc., iBooks. https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewBook?id=409922308, 2010

[8] *iOS Application Programming Guide*, Apple Inc., iBooks. https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewBook?id=409921492, 2010

[9] *Cocoa Fundamentals Guide*, Apple Inc., iBooks, 2010

[10] *iOS Human Interface Guidelines*, Apple Inc., iBooks, 2010

[11] *iOS Technology Overview*, Apple Inc., iBooks, 2010

[12] Neil, S.: *iPad iOS 6, Development Essentials,* iBooks, 2013, ISBN 978-1480191297

[13] Druska, P.: *Návrh iOS aplikácie od úplného začiátku*, iBooks, 2012

[14] *Beginning Objective C Programming, A concise guide to learning Objective C Programming*, iBooks, 2011, ISBN 978-1-105-375646-7

SEZNAM OBRÁZKŮ