ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

FAKULTA
STROJNÍ

BAKALÁŘSKÁ
PRÁCE

MOBILNÍ APLIKACE PRO VYHLEDÁVÁNÍ UČEBEN

2019

ONDŘEJ
BIMKA

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bimka** Jméno: **Ondřej** Osobní číslo: **465361**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídicí techniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Mobilní aplikace pro vyhledávání učeben

Název bakalářské práce anglicky:

Mobile application for a classroom look-up

Pokyny pro vypracování:

1. Zinstalujte a popište instalaci prostředí Apple pro vývoj aplikací
2. Navrhněte ukázkovou aplikaci pro IOS
3. Popište postup stažení a instalace aplikace, napište krátký návod k použití

Seznam doporučené literatury:

Christian Keur and Aaron Hillegass: iOS Programming: The Big Nerd Ranch Guide. 2016 Big Nerd Ranch, LLC.
Start Developing iOS Apps (Swift). The Apple course for developers. Available online:
https://developer.apple.com/library/archive/referencelibrary/
GettingStarted/DevelopiOSAppsSwift/index.html#//apple_ref/doc/uid/TP40015214-CH2-SW1

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Vladimír Hlaváč, Ph.D., U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.04.2019** Termín odevzdání bakalářské práce: **12.06.2019**

Platnost zadání bakalářské práce: _____

Ing. Vladimír Hlaváč, Ph.D.
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Ing. Michael Valášek, DrSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

26 -04- 2019

Datum převzetí zadání

Podpis studenta

Poděkování

Zde bych chtěl poděkovat všem kteří poskytli podklady a odborné znalosti. Také bych chtěl poděkovat vedoucímu práce za odborné vedení a konzultování. V neposlední řadě svojí rodině za materiální a morální podporu.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s použitím literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.

Datum: Podpis:

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta Strojní

MOBILNÍ APLIKACE PRO VYHLEDÁVÁNÍ UČEBEN

Bakalářská práce
Duben 2019
Ondřej Bimka

Abstrakt

Předmětem bakalářské práce „Mobilní aplikace pro vyhledávání učeben“ je vytvořit aplikaci pro vyhledání učeben, popsat její fungování a program ve kterém byla vytvořena.

Klíčová slova

Aplikace, Xcode, vývoj, program, mobilní

Abstract

The subject of the bechelor thesis „Mobile application for a classroom look-up“ is to create an application for a classroom look-up, describe it’s functionality and the program in which has the app been created.

Keywords

Application, Xcode, development, program. mobile

Obsah

Použité značení:

XML	Extensible Markup Language
JSON	JavaScript Object Notation
MB	MegaByte
KB	KiloByte
Framework	Platforma, která poskytuje základ na kterém lze vytvářet další programy
Storyboard	Soubor s vizuálním zobrazením vytvářené aplikace
String	Řada znaků vytvářející dále použitelný objekt
Delegate	Funkce ovládající chování objektu
DataSource	Zdroj používaných dat
Segue	Přechod mezi okny aplikace uskutečňující navigaci
Screenshot	Snímek obrazovky
Optional	Objekt, který nemusí nabývat žádné hodnoty
Outlet	Výstup ze storyboard vytvořený v kódu k jeho úpravě
ViewContoroller	Okno aplikace
Third-party software	Software volně dostupný, nebo koupený od 3. osoby.

1. Úvod

Budova fakulty strojní na Karlově náměstí je rozlehlá a obsahuje mnoho místností. Občas bývá obtížné jednotlivé posluchárny najít, a proto jsem se rozhodl vytvořit tuto aplikaci, která by měla jejich hledání zrychlit a zjednodušit.

Celý proces vytvoření aplikace nemusí být úplně zřejmý. V úvodu této práce se zaměřím na seznámení se s programem a programovacím jazykem. Dále se budu věnovat základnímu postupu při vytváření aplikace a následné distribuci. Na závěr postupně popíši vytvořenou navigační aplikaci, včetně popisu kódu, jeho funkce a také další soubory, které aplikace obsahuje. Při programování jsem se snažil vycházet z použitých zdrojů [1],[2] a [3]. Tyto zdroje bych také doporučil pro začínající i pokročilé vývojáře.

2. Software a programovací jazyk

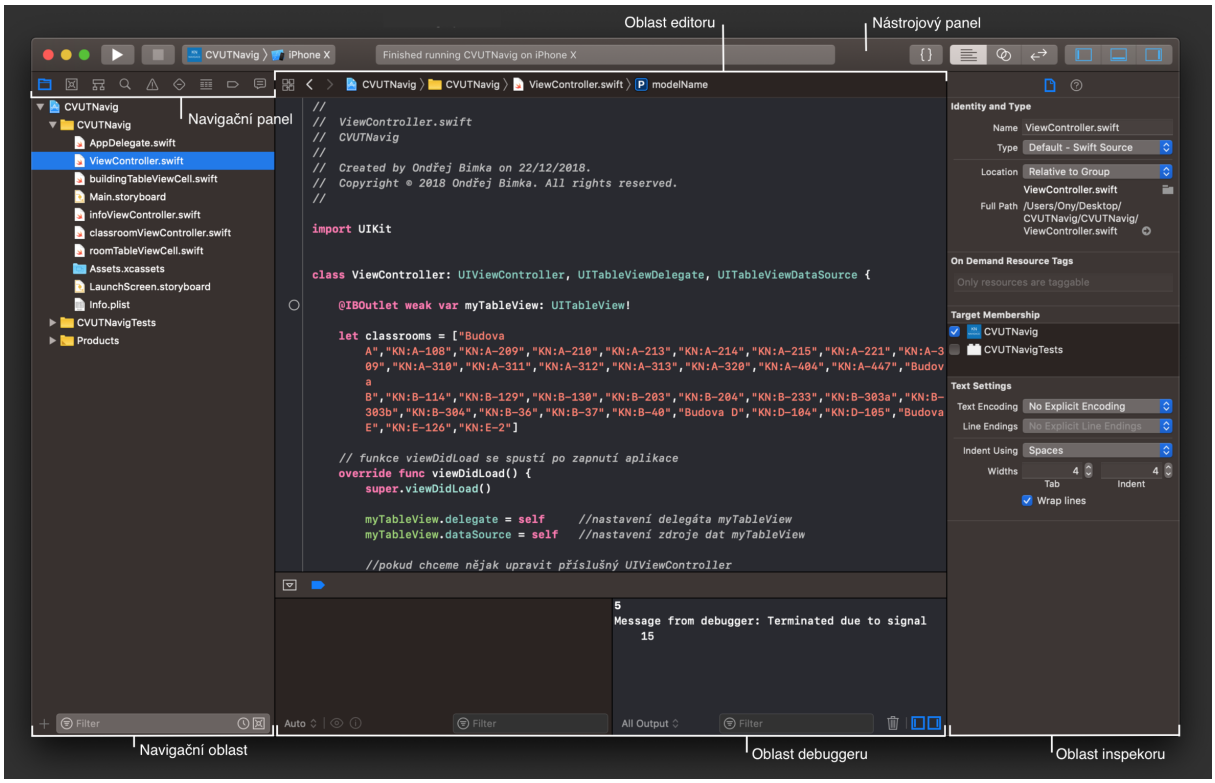
Mobilní aplikace pro systém iOS se dají vytvářet pomocí různých softwarů. Nejznámější je pravděpodobně Xcode, který byl vyvinut společností Apple. Tento program lze stáhnout zadarmo přímo v App Storu. Aplikace se zde dají programovat pomocí dvou programovacích jazyků, Objective-C a Swift.

2.1 Xcode

V tomto programu se dají vytvářet aplikace na iOS, watchOS, tvOS, macOS, různé frameworky atd. Poskytuje vše co je k vývoji aplikace potřeba. Od vytváření aplikace, její optimalizaci, až po její zveřejnění. Jako většina programů, tak i Xcode se stále vyvíjí, poslední verze už neprocházejí tak radikálními změnami, jako verze dřívější.

2.1.1 Uživatelské rozhraní Xcodu

Uživatelské rozhraní není nijak složité. Uprostřed se nachází hlavní pracovní okno a na krajích jsou pomocné a navigační panely. Navigace v projektu se uskutečňuje pomocí přepínání jednotlivých oken. Ve složitějších projektech o více souborech lze použít vyhledávač. V debuggeru se zobrazují upozornění, print funkce a chyby. Oblast inspektoru slouží k různému nastavení a zobrazení informací.

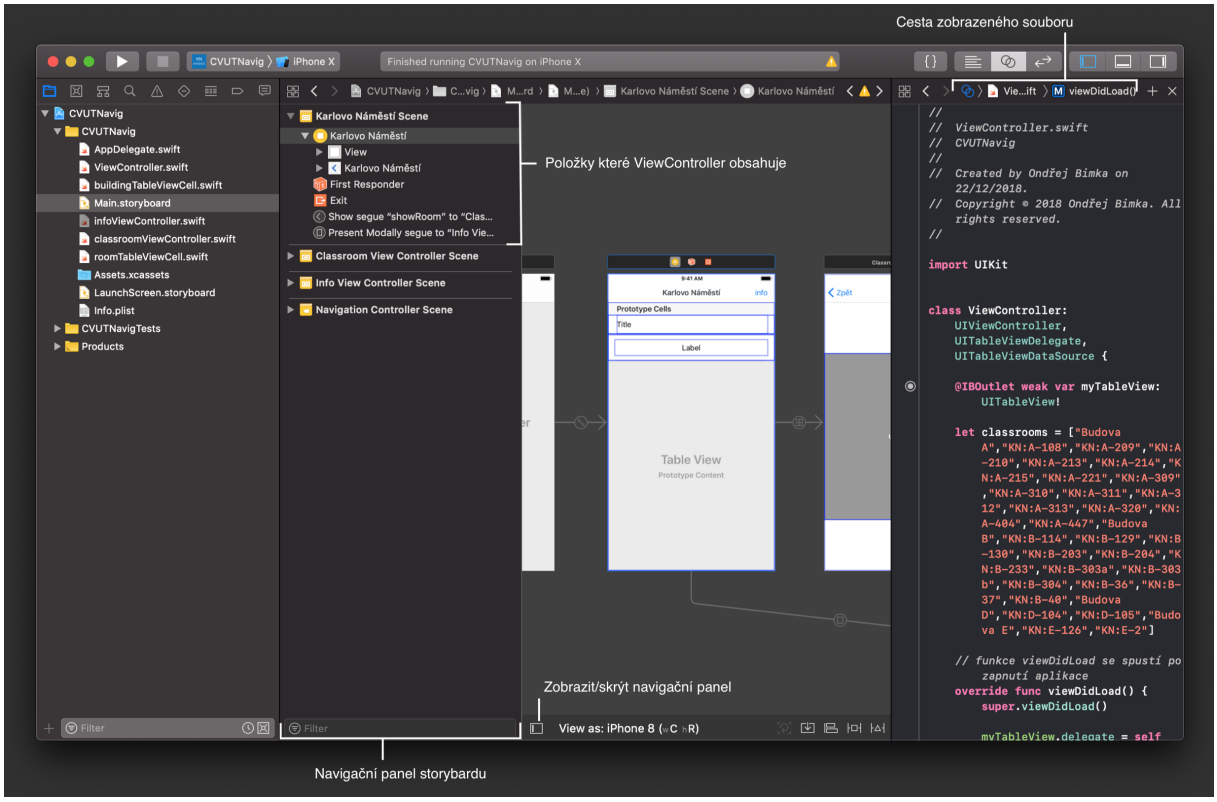


Obr. 1. Základní popis programu Xcode.

V horní části se nachází nástrojový panel. V něm se spouští a vypíná aplikace. Dále se zde nastavuje zobrazení oken v oblasti editoru a nastavení simulačního schématu, včetně volby simulovaného zařízení. V okně aktivity se nachází případná upozornění na nesrovnalosti kódu. Po kliknutí na výstrahu nás Xcode přesměruje ke zdroji chyby. Existují dva typy upozornění. Výstražné žluté, které Xcode při spouštění aplikace vypíše a aplikaci spustí a červené, které nedovolí spuštění aplikace, nebo aplikaci shodí.

Obr. 2. Popis nástrojového panelu

Pokud se nacházíme ve storyboardu, můžeme zobrazit další navigační panel pro ViewControllery a také lze využít často používané kombinované zobrazení ViewControlleru a jeho kódu.

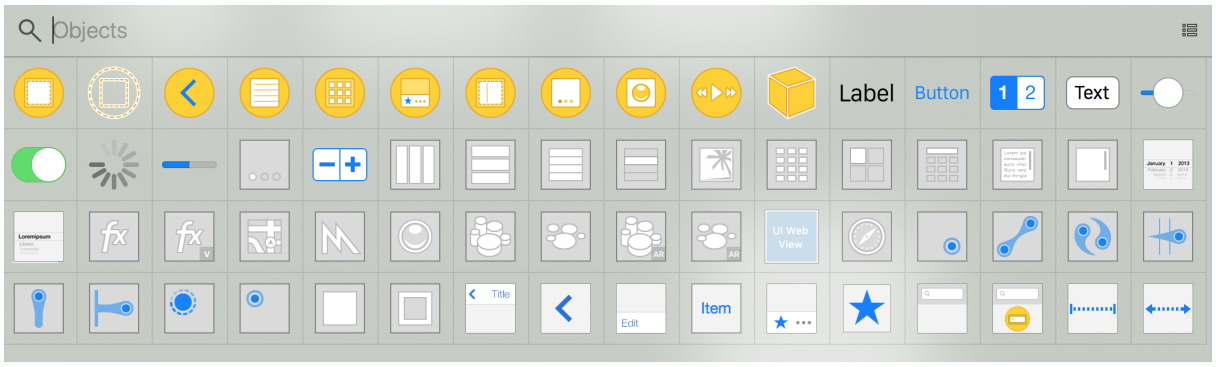


Obr. 3. Ukázka programu Xcode ve storyboard

2.1.2 Základní soubory a objekty v Xcode

- ViewController – jeden z nejdůležitějších stavebních prvků běžných aplikací. Každá aplikace obsahuje alespoň jeden. Představuje okno které vidíme.
- NavigationController – automaticky vytvoří navigační panel, který se dá různě nastavit.
- Assets.xcassests – do této složky se umisťují objekty, jako obrázky, ikony textury a používané v aplikaci.
- Main.storyboard – jedná se o hlavní storyboard kam umisťujeme jednotlivá okna (UIViewControllery). Pokud děláme složitější aplikaci je lepší vytvořit další storyboard, aby všechny ViewControllery nebyly umístěny v Main.storyboard.
- LaunchScreen.storyboard – zde nastavujeme co se nám zobrazí mezi kliknutím ikony a plným spuštěním aplikace.
- Info.plist - V této složce je základní nastavení aplikace. Lze ji zobrazit jako property list, nebo jako XML code.

Knihovna s objekty, které lze vložit do Main.storyboard.

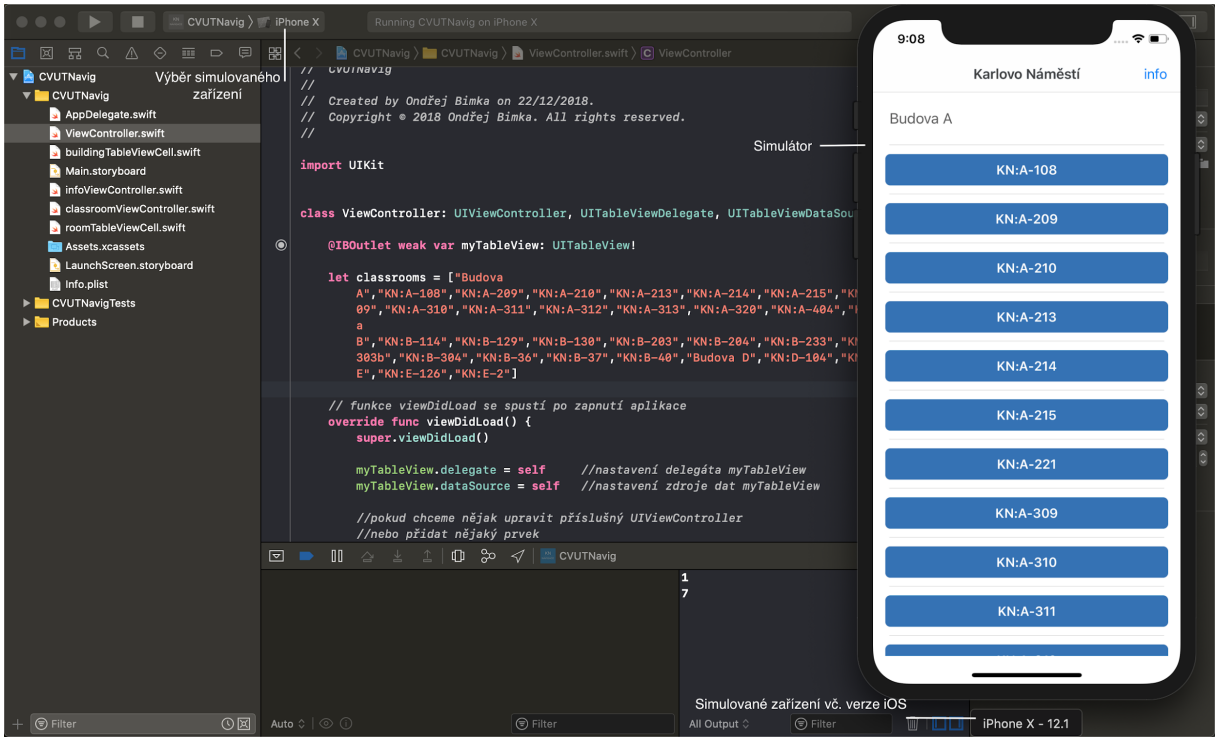


Obr. 4. Knihovna s objekty vložitelnými do storyboard.

Tyto objekty se po přidání do storyboardu dají nastavit přímo (v pravém sloupci Xcodu se po kliknutí na daný objekt zobrazí možnosti nastavení), nebo po připojení do příslušné „class“ pomocí kódu.
Není problém si vytvořit jakýkoliv vlastní objekt, ale pokud to jde, je lepší použít již definované objekty.

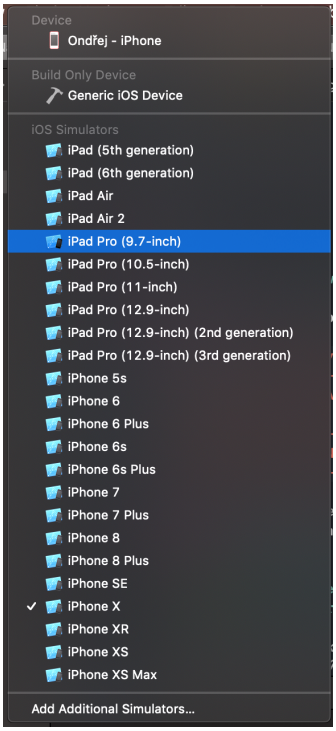
2.1.3 Simulace v Xcode

Xcode obsahuje simulátory jednotlivých zařízení. Simulátor se chová stejně jako reálné zařízení, avšak má svoje nevýhody a to například v chybějící simulaci fotoaparátu. Nastavení simulovaného zařízení se nachází v horním panelu. Simulátor se ovládá pomocí myši a klávesových zkratek. Při úpravě projektu je potřeba simulátor znovu zapnout, aby byla změna vidět.



Obr. 5. Ukázka simulátoru.

Při simulaci na reálném zařízení je nutné toto zařízení připojit k počítači pomocí nabíjecího kabelu. Aby šlo aplikaci simulovat v zařízení musí verze iOS v telefonu odpovídat verzi na kterou je aplikace vyvíjena. Připojené zařízení by se mělo zobrazit v nabídce simulovaných zařízení.



Obr. 6. Seznam zařízení k simulaci.

Po spuštění simulace se celá aplikace nahraje do zařízení a zůstane tam ve stejném stavu dokud nezahájíme simulaci znovu s nějakou úpravou, nebo aplikaci nesmažeme. Po odpojení reálného zařízení od počítače zůstane aplikace v zařízení a lze jí tak testovat kdekoli. Stručně řečeno, celá aplikace se při simulaci nahraje a funguje stejně jako ta, kterou bychom stáhli z App Store.

Pokud dojde při simulaci k chybě, aplikace spadne a Xcode nám zdroj chyby ukáže. Jsou i chyby, které Xcode nezvládne přesně lokalizovat, v tomto případě v debuggeru napíše o jaký druh chyby se jedná, aby bylo snazší chybu najít.

2.2 Swift

Jedná se o relativně jednoduchý open source programovací jazyk, který vznikl v roce 2014 a patří k nejrychleji rostoucím jazykům. Vychází z Objective-C, C a dalších programovacích jazyků. Je "type-safe", to znamená, že pokud kód vyžaduje nějaký typ hodnoty, nelze mu přiřadit jiný typ.
Swift, stejně jako Xcode prochází neustálým vývojem, ale stejně jako u Xcodu nejsou nynější změny tak znatelné jako dříve. Např. když se přecházelo z verze Swift 2 na Swift 3, tak musela být půlka celé aplikace pozměněná. Některé funkce se aktualizují automaticky, ale některé věci se musí přepsat „ručně". Při přechodu ze Swiftu 3 na Swift 4 nebyla změna tak radikální jako v předchozím případě.
Nyní se pracuje na verzi Swift 5. Beta verze Swiftu 5 je už dostupná v iBooks i v oficiálních materiálech od Applu.

2.2.1 Základy Swiftu

U Swiftu se nemusí za kódem používat „ ; " (ale může) a nemusíme specifikovat o jaký typ se jedná, ale Swift to pozná sám.

```
let string = "nějaký text"
```

V tomto případě nemusíme specifikovat, že se jedná o string , ale Swift typ automaticky rozpozná.

Stejně jako v programovací jazyk C, používá Swift proměnné pro ukládání a používání hodnot. Swift navíc obsahuje konstanty, což jsou hodnoty, které se nemění. Příklad konstant a proměnných se nachází v následujícím příkladu.

```
//Proměnné a konstanty
let konstanta = "Tento text nelze změnit"
konstanta = "Změněná konstanta"  //Nelze změnit        Cannot assign to value: 'konstanta' is a 'let' constant

var proměnná = "Změnitelný text"
proměnná = "Změněná proměnná"  //Toto je teď hodnota proměnné
```

Z příkladu je zřejmé, že konstanty nelze měnit, a proto se v kódu při pokusu o změnu konstantní hodnoty zobrazí upozornění o chybě.

Základní prvky

- Int – celá čísla

```
let příkladInt = 3
```

- Double – desetinná čísla. Preferováno před Float.

```
let příkladDouble = 3.32
```

- Float – desetinná čísla. Přenější na méně desetinných míst než Double

```
let příkladFloat:Float = 3.32
```

pokud by příklad Float nebylo specifikováno jako Float, byla by to hodnota Double.

- Bool – může nabývat pouze hodnoty true (pravda, 1), nebo false (nepravda, 0).
- String – textová data

Komentáře se vytvářejí pomocí tohoto znaku: "//" (vytvoří komentář na daném řádku), nebo „/*" a komentář pokračuje dokud není ukončen pomocí „*/". Komentář nemá žádný vliv na zbytek kódu a slouží jako poznámka nebo informace pro programátora, nebo toho kdo bude kód číst.

```
//Příklad komentáře

/* Příklad víceřádkového
   komentáře   */
```

Základní collection types.

- Array – řazené hodnoty stejného typu. Hodnota se může opakovat.

```
let mojeŘada = ["auto","motorka","kolo","koloběžka","autobus","auto"]
```

- Set – neřazená řada unikátních hodnot stejného typu. Hodnota se nesmí opakovat.

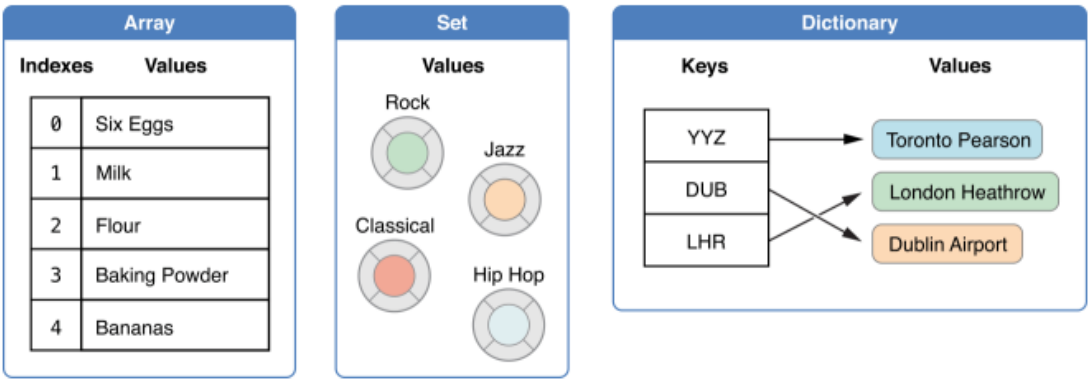
```
let můjSet:Set = ["auto","motorka","kolo","koloběžka","autobus"]
```

- Dictionary – každá hodnota obsahuje svůj vlastní klíč.

```
let můjDictonary = [1:"uživatel1",2:"uživatel2",4:"uživatel4"]
```

můjDictionary je typu [Int:String].

Pro názornější vysvětlení je zde obrázek popisující collection types.



Obr. 7. Názorná ukázka collection types.

Funkce se ve Swiftu vytváří a následně spouští takto:

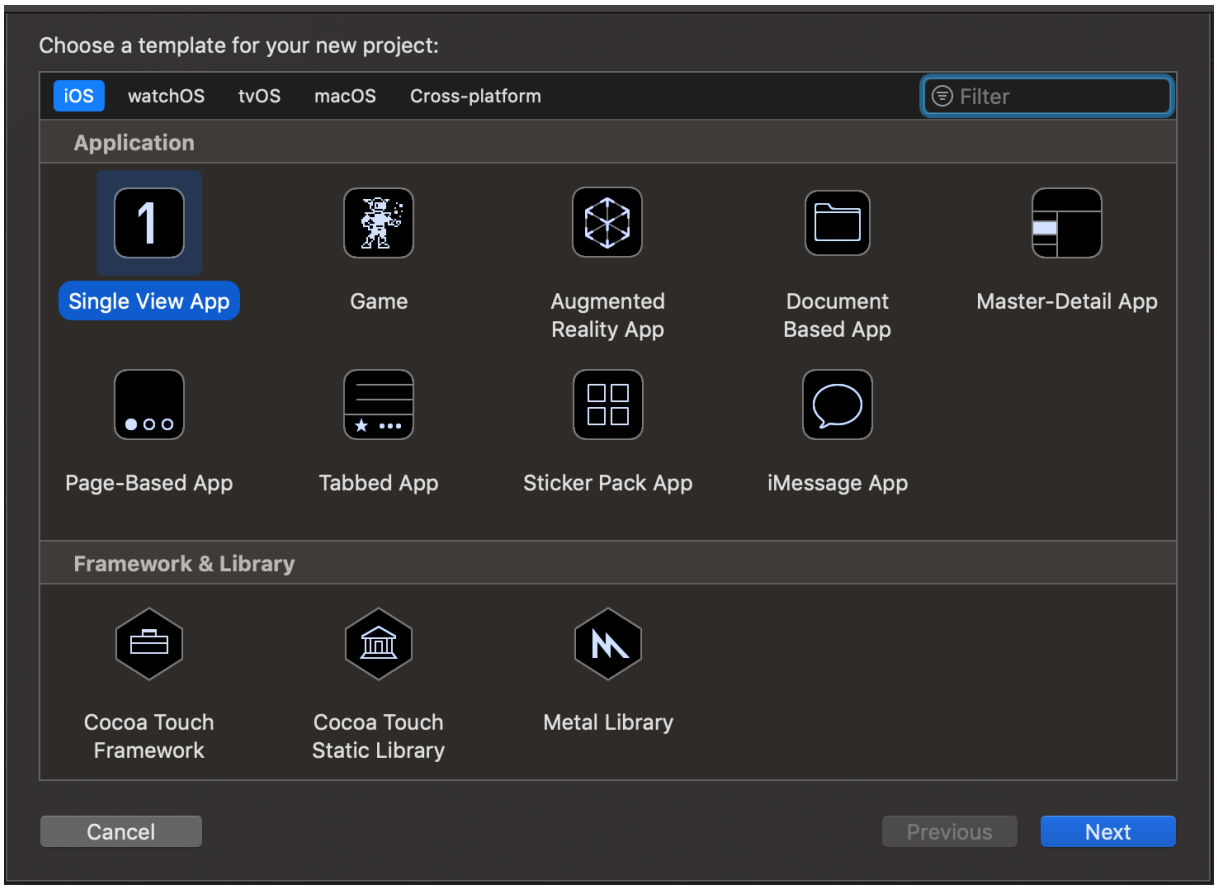
```
func mojeFunkce() {
    print("Vítej světe")
}

mojeFunkce() // Vítej světe
```

2.2.2 Propojení Swiftu a Objective-C v jedné aplikaci

Vzhledem k tomu, že různé frameworky mohou být psány v Objective-C, ale aplikace může být psaná ve Swiftu, používají se k tomuto účelu soubory zvané „bridging-headers“. Princip je velice jednoduchý. Tyto soubory nám po jednoduchém nastavení propojí Objective-C soubory s naší aplikací.

3. Vytvoření nové aplikace

Při vytváření nové aplikace v Xcode si nejprve zvolíme, jaký typ aplikace budeme vytvářet. Dále vyplníme potřebná pole a klikneme na tlačítko vytvořit nový projekt. Podle zvoleného typu se nám vytvoří základní šablona pro daný typ aplikace, která obsahuje úplně základní složky specifické pro danou aplikaci. Všechny složky se dají kdykoli do projektu vytvořit zpětně, proto na zvolené šabloně nezáleží, ale pouze proces zjednoduší.



Obr. 8. První okno při vytváření nového projektu.

3.1 První kroky při vytváření jednoduché aplikace

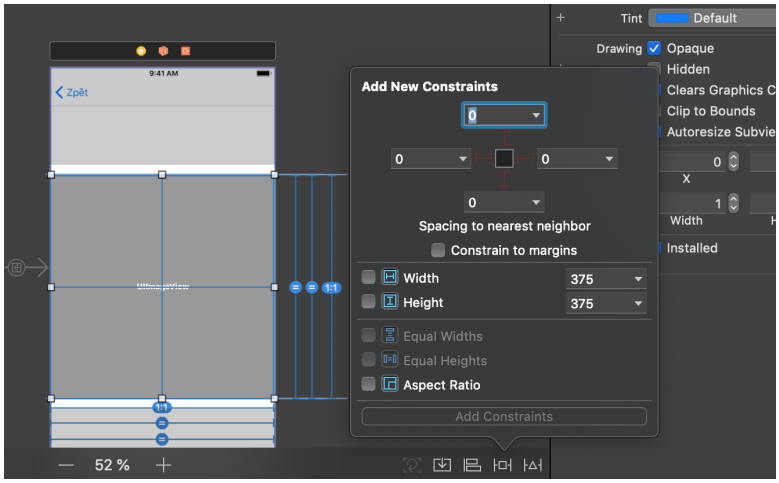
Při vytváření běžné aplikace se dá použít šablona „single view application“. Jedná se o nejjednodušší šablonu, která nám vytvoří pouze základní „Main.storyboard“ s jedním ViewControllerem ke kterému je automaticky přiřazen ViewController.swift, dále LaunchScreen.storyboard, Assets.xcassets, Info.plist a AppDelegate.swift.

Při vytváření nových ViewControllerů se postupuje tak, že ho vytvoříme ve storyboardu a vytvoříme k němu nový .swift file, ve kterém se vytvoří příslušná class a do té se píše kód.

Pokud chceme například UIViewController, který obsahuje UITableView, nebo UICollectionView. Můžeme je vytvořit z normálního ViewControlleru, nebo se dají vytvořit rovnou. Výhoda postupu, kdy je vytvoříme rovnou je ta, že jsou tam všechny základní funkce už přednastaveny a nemusíme nic řešit. Pokud vytvoříme nejdříve běžný ViewController, je nutné stanovit delegate a dataSource funkce.

3.2 Nastavení rozměrů

Vzhledem k tomu, že lze vytvořit jednu aplikaci pro všechna zařízení je nejlepší nastavit rozměry obecně, tak aby byly rozměry na všech zařízeních v pořádku. Rozměry se ve storyboardu nastavují tak, že zvolíme prvek jehož rozměry chceme nastavit a přidáme rozměrové délky.



Obr. 9. Ukázka okna pro přídání rozměrů.

Pokud nějaký objekt vytváříme pomocí kódu nastavuje se jeho rozměr pomocí CGrect a to následovně.

```
let mojeView = UIView()
mojeView.frame = CGRect(x: 0, y: 0, width: 100, height: 100)
```

X a y jsou počáteční body, kde se mojeView bude nacházet od levého horního rohu obrazovky a width a height nám stanoví šířku a výšku myView.

3.3 Navigace v aplikaci

Vstupní bod, neboli „entry point“ nám definuje, který ViewController se nám po spuštění zobrazí jako první. Dále je navigace mezi různými ViewControllery možná provést přímo „ručně“, to znamená, že zmáčknutím pravého tlačítka myši a následným přetažením z jednoho ViewControlleru na druhý se nám vytvoří segue, která při nastavených podmínkách navigaci vykoná (například zmáčknutí určeného tlačítka). Navigace se dá nastavit i pomocí kódu. Častá je taky možnost kombinace kódu a „ruční metody“, kdy vytvoříme seque ve storyboardu, dáme jí jméno a následně jí podle jména vyvoláme pomocí funkce.
Jak průběh navigace vypadá, lze nastavit pomocí kódu, nebo přímo ve storyboardu. Lze si vybrat z několika způsobů animace, nebo si lze vytvořit vlastní.

3.4 Přenos dat v aplikaci mezi ViewControllery

Na posílání dat mezi jednotlivými ViewControllery je více možností:

- Použitím „prepareForSegue“ funkce.
- Pomocí delegáta – složitější možnost ale umožňuje ovládat i funkce.
- Pomocí NSNotifications.
- Další metody…

Jedna z dalších metod posílání dat mezi jednotlivými ViewControllery, kdy data vložíme přímo do proměnné v jiném ViewControlleru, který poté pomocí funkce zobrazíme.

```
func sendData(){

    let storyboard: UIStoryboard = UIStoryboard(name: "SupportStoryboard", bundle: nil)
    let vc: contactViewController = storyboard.instantiateViewController(withIdentifier:
        "contactViewController") as! contactViewController

    vc.userId = self.userId
    self.present(vc, animated: true, completion: nil)
}
```

3.5 Uložení dat

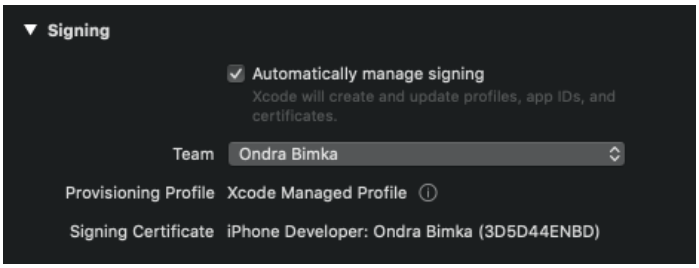
Pokud máme data, která chceme uložit, můžeme zvážit více způsobů. Pokud ukládáme malý objem dat, jako je například uživatelské jméno nebo session token, tak k tomuto účelu použijeme UserDefaults. Tato funkce je přímo pro tento typ dat určená.
Zde se nachází příklad použití UserDefaults.

```
//Uložení "jméno_k_uožení" pod klíč "username"
UserDefaults.standard.set("jméno_k_uložení", forKey: "username")

//Získání "jméno_k_uložení" pomocí UserDefaults
let username = UserDefaults.standard.object(forKey: "username") as! String
```

Pokud potřebujeme větší objem dat, lze použít CoreData, které nám umožní vytvořit námi definovanou databázi přímo v telefonu a data z ní lze získat i po zavření aplikace. Poslední možnost je použití nějaké externí databáze, se kterou pak komunikace probíhá pomocí JSON a PHP skriptů. Poslední typ se používá hlavně, když máme aplikaci, která má registrované uživatele a sdílený obsah.

4. Distribuce

K distribuci aplikací na App Storu je nutné mít zakoupený Apple Developer Program. Před začátkem distribuce je nutné přiřadit svůj developerský účet do kolonky „signing“, Xcode už potom vše přiřadí automaticky. Aplikace se pak vydávají a spravují přes iTunes Connect, na který se po přihlášení dostaneme přes internetový vyhledávač. V iTunes Connect pak můžeme aplikace spravovat, nebo získávat statistiky o počtu stažení.



Obr. 10. Přiřazení developerského účtu k aplikaci.

4.1 Developer program

Apple Developer Program stojí pro developera (jednotlivce) v přepočtu 100 dolarů ročně. Umožňuje umístit vytvořené aplikace na App Store a uživatel také obdrží přístup ke všem beta verzím.

4.2 Umístění aplikace na App Store pomocí iTunes Connect

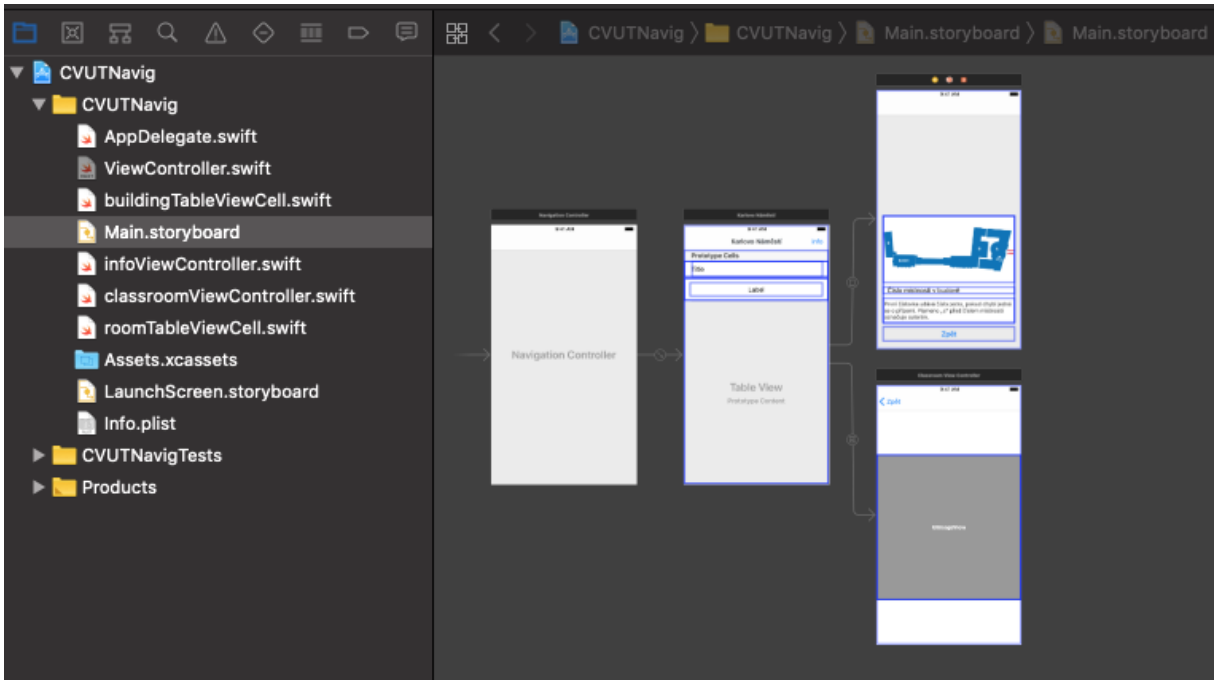
Po dokončení aplikace jí musíme archivovat. Dále jí musíme distribuovat a pokud splňuje požadované parametry tak se nám po zpracování objeví možnost přidat build v iTunes Connect. Po přidání několika screenshotů z aplikace, vyplnění popisu, přidání ikony, případně dalších věcí, jako defaultní user, pokud aplikace vyžaduje přihlašování. Můžeme aplikaci předložit ke schválení, které obvykle trvá několik dnů a pokud je vše v pořádku, tak se aplikace objeví na App Storu. Pokud aplikace nesplňuje nutné podmínky, je zamítnuta a nesrovnalost se musí opravit. Potom se celý proces opakuje, dokud není aplikace schválena.

4.3 Vydání nové verze

Pokud chceme vydat novou verzi existující aplikace je proces podobný, jako při jejím umisťování na App Store. U nové verze v Xcodu změníme číslo verze a číslo buildu. Zarchivujeme, validujeme, přídáme build na iTunes Connect, případně změníme popis a napíšeme co obsahuje daná verze za novinky a nakonec předložíme novou verzi ke schválení.

5. Popis a fungování CVUTNAvig aplikace

Tato aplikace slouží ke snadnějšímu nalezení místností na Karlově Náměstí. Skládá ze tří ViewControllerů. Dále obsahuje potřebné soubory, jako Assets.xcassets, Info.plist a Launchscreen.storyboard.

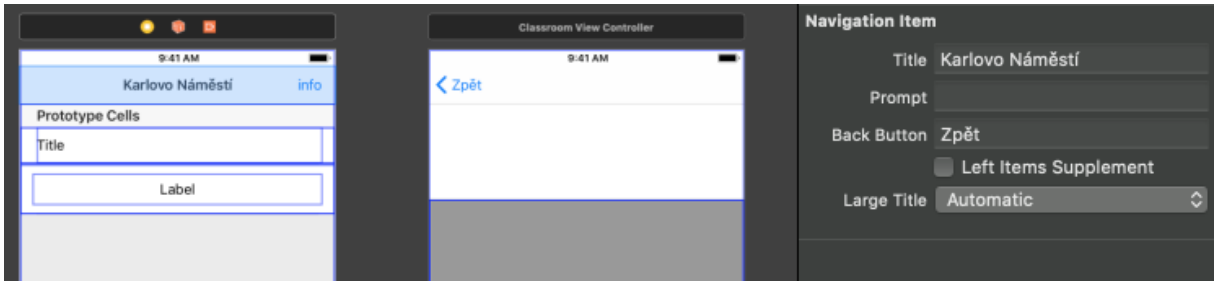


Obr. 11. Ukázka souborů CVUTNavig aplikace

Po zapnutí aplikace se zobrazí vstupní ViewController, který obsahuje horní navigační panel a UITableView, který se skládá z jednotlivých buněk. Po rozkliknutí buňky se nám zobrazí jiný ViewController, který se jmenuje ClassroomViewController. V ClassroomViewControlleru je vyznačená hledaná místnost, která je zobrazena na obrázku, který se dá případně přiblížit. Poslední UIViewController se zobrazí po kliknutí na info tlačítko, nacházející se v pravém horním rohu na úvodní stránce. Obsahuje informační text a obrázek s celkovým plánem budovy na Karlově Náměstí.

5.1 Navigace

Pro přehlednou navigaci je hlavní ViewController vložen do UINavigationViewControlleru. Ten vytvoří navigační panel, který lze nastavit přímo ve storyboardu.

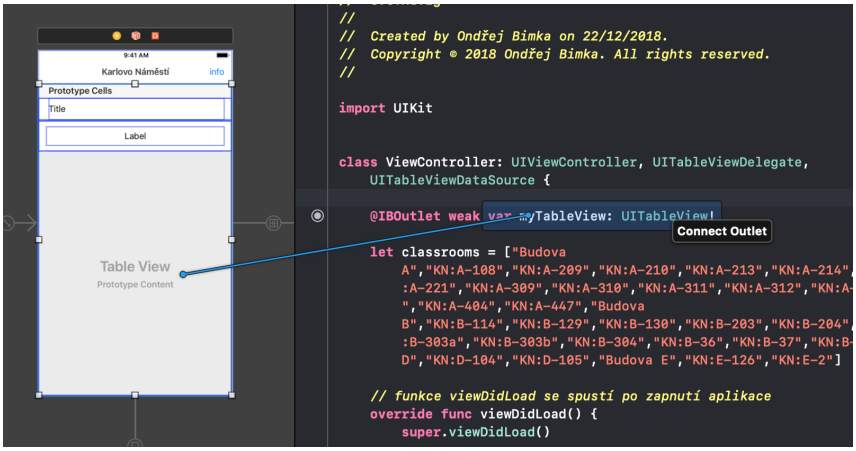


Obr. 12. Ukázka nastavení navigačního panelu.

Při použití seguee typu “Show(e.g. Push)” se navigační panel zobrazí i v zobrazeném ClassroomViewControlleru. Zpět se lze dostat zmáčknutím zpětného tlačítka, nebo táhnutím prstem z pravého okraje obrazovky.
InfoViewController je zobrazen pomocí „Present modally“ seguee. V tomto případě se navigační panel nevytvoří.

5.2 ViewController a zdrojový kód

Nejprve jsem do ViewControlleru ve storyboardu z knihovny přidal info tlačítko a dále TableView, který by nám bez dalšího nastavení nic nezobrazoval. Pokud mám nějaký přidaný prvek v UIViewControlleru a chci ho upravit pomocí kódu, tak na něj kliknu pravým tlačítkem myši tažením do okénka kódu se nám spojení automaticky vytvoří. Po vytvoření spojení lze objekt upravovat pomocí kódu.



Obr. 13. Propojení ViewControlleru s kódem.

Následně jsem definoval řadu classrooms, která obsahuje budovy a místnosti v číselném pořadí.

```
let classrooms = ["Budova
    A","KN:A-108","KN:A-209","KN:A-210","KN:A-213","KN:A-214","KN:A-215","KN:A-221","KN:A-309
    ","KN:A-310","KN:A-311","KN:A-312","KN:A-313","KN:A-320","KN:A-404","KN:A-447","Budova
    B","KN:B-114","KN:B-129","KN:B-130","KN:B-203","KN:B-204","KN:B-233","KN:B-303a","KN:B-30
    3b","KN:B-304","KN:B-36","KN:B-37","KN:B-40","Budova D","KN:D-104","KN:D-105","Budova
    E","KN:E-126","KN:E-2"]
```

Pro zobrazení dat v TableView je dále nezbytné přidat UITableDelegate a UITableDataSource. UIViewController je tam automaticky vždy, když vytváříme ViewController .swift soubor.

```
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {
```

Dále nastavit delegate a dataSource pro myTableView. Self odkazuje na „class“ jménem ViewController ve které se kód nachází. Od této chvíle si myTableView bere data a chová se podle funkcí, které se nacházejí v ViewControlleru. Spoustu funkcí je už předdefinovaných a pokud není potřeba něco hodně specifického nejsou potřeba vytvářet nové.

```
// funkce viewDidLoad se spustí po zapnutí aplikace
override func viewDidLoad() {
    super.viewDidLoad()

    myTableView.delegate = self      //nastavení delegáta myTableView
    myTableView.dataSource = self    //nastavení zdroje dat myTableView

    //pokud chceme nějak upravit příslušný UIViewController
    //nebo přidat nějaký prvek
    //dělá se to v této funkci

    //Např zde můžeme nastavit barvu pozadí
    self.view.backgroundColor = UIColor.white

    //nebo barvu nějakého prvku
    self.myTableView.backgroundColor = UIColor.white
}
```

Následující funkci používáme pro zobrazení dat a pro úpravu designu buněk. Nejprve je potřeba buňku zaregistrovat a po potřebné úpravě vrátit pomocí return. U budov jsem změnil pouze defaultní text, který lze použít, pokud není nutná složitější úprava. Buňka totiž sama o sobě základní text obsahuje. Pro místnosti jsem vytvořil nový UILabel, který jsem nazval classroomLabel a ten jsem nastavil s modrým pozadím, tak aby se místnosti odlišily od budov.

```
//Jedna z "delegate" gunkcí, která umožňuje úpravu jednotlivých buněk
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

    //IndexPath.row určuje pořadí buněk. Jsou zero based, proto začíná od nuly
    if indexPath.row == 0 || indexPath.row == 16 || indexPath.row == 29 || indexPath.row == 32 {

        let cell = tableView.dequeueReusableCell(withIdentifier: "buildingCell", for: indexPath) as! buildingTableViewCell //"Registrace" buňky
        cell.textLabel!.font = UIFont.systemFont(ofSize: 18)   //Font textu
        cell.textLabel!.textAlignment = .left                  //Rovnání textu
        cell.textLabel!.text = classrooms[indexPath.row]       //Text buňky
        cell.textLabel!.textColor = UIColor(red: 90/255, green: 90/255, blue: 90/255, alpha: 1) //Barva textu
        return cell

    } else {

        let cell = tableView.dequeueReusableCell(withIdentifier: "roomCell", for: indexPath) as!   roomTableViewCell  //"Registrace" buňky
        cell.classroomLabel.font = UIFont.boldSystemFont(ofSize: 16)  //Font textu
        cell.classroomLabel.textColor = UIColor.white                 //Barva textu
        cell.classroomLabel.layer.masksToBounds = true               //Nastavení layeru kvůli dalším úpravám
        cell.classroomLabel.layer.borderColor = UIColor.cvutBlue.cgColor  //Barva okraje
        cell.classroomLabel.backgroundColor = .cvutBlue                 //Barva pozadí
        cell.classroomLabel.layer.borderWidth = 1.5                     //Tloušťka okraje
        cell.classroomLabel.layer.cornerRadius = 6                      //Zaoblení hrany
        cell.classroomLabel.layer.shadowRadius = 6                      //Zaoblení stínu
        cell.classroomLabel.text = classrooms[indexPath.row]            //Text buňky
        cell.classroomLabel.layer.shadowColor = UIColor(red: 100/255, green: 100/255, blue: 100/255, alpha: 0.8).cgColor  //Barva stínu
        cell.classroomLabel.layer.shadowOffset = CGSize(width: 1.0, height: 1.0)     //Poloha stínu
        cell.classroomLabel.layer.shadowOpacity = 1.0           //Průhlednost stínu
        return cell

    }

}
```

V tomto případě máme pouze jednu řadu dat na zobrazení, z tohoto důvodu chceme pouze jednu sekci, proto return 1.

```
//Tato funkce nastavuje počet sekcí.
func numberOfSections(in tableView: UITableView) -> Int {
    return 1
}
```

Následující funkce určuje počet buněk v sekci. Je potřeba aby počet buněk odpovídal počtu prvků v řadě classrooms. K tomuto účelu slouží funkce count, která nám vypočítá počet prvků v řadě, jako

číselnou hodnotu. Počet buněk bude poté vždy roven počtu prvků v řadě classrooms. Nakonec tento počet vrátíme pomocí return.

```
//Tato nám určuje počet buněk.
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return classrooms.count
}
```

Abych od sebe více odlišil místností a budovy, změnil jsem výšku buněk ve kterých se nacházejí. Buňky budov jsou vyšší, než buňky místností.

```
//Tato nám nastavuje výšku buněk.
func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
    if indexPath.row == 0 || indexPath.row == 16 || indexPath.row == 29 || indexPath.row == 32 {
        return 65
    }else{
        return 60
    }
}
```

Po kliknutí na buňku se ukáže ViewController obsahující zvýrazněnou místnost. Segue je definovaná v Main.storyboardu. Vzhledem k tomu, že při kliknutí na název budovy by se nemělo nic dít, proto return.

```
//Tato nám určuje co by se mělo dít po kliknutí na buňku.
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    if indexPath.row == 0 || indexPath.row == 16 || indexPath.row == 29 || indexPath.row == 32 {
        return
    } else {
        performSegue(withIdentifier: "showRoom", sender: self) //Tento řádek uskuteční pojmenovanou seque
    }
}
```

Pro zobrazení dat v jiném ViewControlleru je potřeba data odeslat. V tomto případě odesíláme název zvolené místnosti do cílového classRommViewControlleru pomocí prepareForSegue funkce. Showroom je název seguee, která nám přepne controllery. IndexPath.row nám říká kolikátá buňka v pořadí byla zvolena, díky čemuž vybereme z řady classrooms příslušnou zvolenou místnost. Vybraný text je odeslán do cílového ViewControlleru, kde se uloží pod proměnnou className.

```
//Navigační funkce která pošle data do classroomViewControlleru
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "showRoom" {
      if let indexPath = self.myTableView.indexPathForSelectedRow {
        let controller = segue.destination as! classroomViewController
        print(indexPath.row)
        controller.className = classrooms[indexPath.row]
        }
    }
}
```

Přímo v zařízení vypadá vstupní ViewController takto.

Obr. 14. Screenshot ViewControlleru ze simulátoru.

Horní navigační panel byl přidán přímo v Main.storyboard pomocí NavigationControlleru. Jeho výhoda spočívá v tom, že se automaticky zobrazuje i v dalších ViewControllerech, dělá tak navigaci přehlednější.

5.3 ClassRoomViewController a zdrojový kód

ClassRoomViewController obsahuje UIScrollView, ve kterém je umístěný UIImageView, z důvodu možnosti přiblížení obrázku s vyznačenou místností. Dále se zde nachází proměnná className, do které pomocí segue fuknce z úvodního ViewControlleru uložíme přijatý název místnosti. Jedná se o název, proto jsem jako typ zvolil string. Otazník znamená, že se jedná o optional value, takže může být className prázdná, aniž by došlo ke spadnutí aplikace. Pokud bychom chtěli otevřít prázdnou optional hodnotu dojde k chybě.

```
class classroomViewController: UIViewController, UIScrollViewDelegate {

    var className:String? //Proměnná do které uložíme jméno místnosti

    @IBOutlet weak var planImage: UIImageView!

    @IBOutlet weak var myScrollView: UIScrollView!
```

Následuje funkce viewDidLoad, kde se nachází nastavení delegáta a přiblížení pro myScrollView. Dále je zde relativně dlouhý kód, který však obsahuje pouze nastavení designového prvku. Podle druhu zařízení se upravuje pouze velikost tohoto prvku. Pro samostatné fungování aplikace není nějak významný.

```
override func viewDidLoad() {
    super.viewDidLoad()

    myScrollView.minimumZoomScale = 1.0   //Max možné přiblížení obrázku
    myScrollView.maximumZoomScale = 2.0   //Min možné přiblížení obrázku
    myScrollView.delegate = self          //Delegate pro myScrollView

    planImage.contentMode = .scaleAspectFit //Nastavení zobrazení obrázku

    //Následující kód vytváří přechodový efekt  z modré do bílé
    let forGradient = UIView()
    let gradiendUp = CAGradientLayer()

    //Rozpoznání zařízení a následná úprava velikosti přechodú
    if (UIDevice.modelName == "iPhoneXs" || UIDevice.modelName == "iPhoneXr" || UIDevice.modelName == "iPhoneXsMax" ||
        UIDevice.modelName == "iPhoneX") {

        view.layoutIfNeeded()
        forGradient.frame = CGRect(x: 0, y: self.view.bounds.maxY - 230, width: UIScreen.main.bounds.width, height: 230)
        gradiendUp.frame = CGRect(x: 0, y: 0, width: forGradient.bounds.width, height: forGradient.bounds.height)

    } else if (UIDevice.current.userInterfaceIdiom == .pad) {
        view.layoutIfNeeded()
        forGradient.frame = CGRect(x: 0, y: self.view.bounds.maxY - 100, width: UIScreen.main.bounds.width, height: 130)
        gradiendUp.frame = CGRect(x: 0, y: 0, width: forGradient.bounds.width, height: forGradient.bounds.height)
    } else {
        view.layoutIfNeeded()
        forGradient.frame = CGRect(x: 0, y: self.view.bounds.maxY - 130, width: UIScreen.main.bounds.width, height: 130)
        gradiendUp.frame = CGRect(x: 0, y: 0, width: forGradient.bounds.width, height: forGradient.bounds.height)
    }
    forGradient.backgroundColor = UIColor.clear
    let secondColor = UIColor(red: 1, green: 1, blue: 1, alpha: 1).cgColor
    let firstColor = UIColor.cvutBlue.cgColor
    gradiendUp.colors = [secondColor,firstColor]
    gradiendUp.locations = [0.0,1.0]
    gradiendUp.borderColor = UIColor.black.cgColor
    forGradient.layer.insertSublayer(gradiendUp, at: 1)
    view.insertSubview(forGradient, at: 0)
    //konec kódu pro přechod
}
```

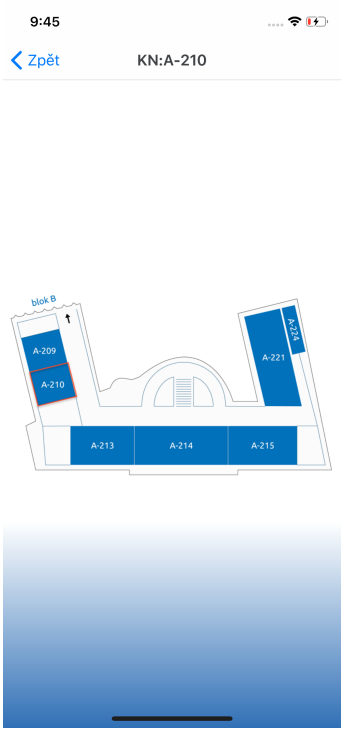
Aby byl nadpis místnosti i obrázek aktuální ještě před zobrazením classroomViewControlleru použijeme funkci viewWillAppear. Obrázky odpovídající místnostem se jmenují stejně jako příslušná místnost, proto lze název místnosti rovnou použít jako název obrázku. Vzhledem k tomu, že je className optional, tak je potřeba se nejdříve přesvědčit, zda není prázdná. Pokud má className hodnotu, tak se zobrazí příslušný obrázek a titulek v horní části zařízení.

```
//Funkce viewWillAppear
override func viewWillAppear(_ animated: Bool) {
    if className != nil {              //Kontrola, jestli className proměnná není prázdná
        self.navigationItem.title = self.className    //Nastavení nadpisu na navigationItem
        planImage.image = UIImage(named: className!)  //Nastavení obrázku.
    }else{
        return
    }
}
```

Nakonec pomocí funkce viewForZooming nastavím, v jakém případě povolit přiblížení obrázku. Vzhledem k velikosti iPadů u nich není zoom potřeba, proto return nil, který nevrátí žádné view k zoomování. Pokud se nejedná o pad, vrátí nám tento kód view s obrázkem, které lze zoomovat.

```
//Předdefinovaná delegate funkce pro UIScrollView
func viewForZooming(in scrollView: UIScrollView) -> UIView? {

    if (UIDevice.current.userInterfaceIdiom == .pad) {
        return nil //U ipadu není zoom potřeba
    } else {
        return self.planImage
    }
}
```

V simulátoru vypadá classRoomViewController takto.



Obr.15 . Screenshot ClassRoomViewControlleru ze simulátoru.

V dolní části lze vidět přechod z modré do bíle, který je definovaný ve ViewDidLoad funkci. Dále nahoře lze vidět navigační panel, který je zde automaticky i se zpětným tlačítkem. Pouze titulek se musí nastavit tak, aby odpovídal zvolené místnosti.

5.4 InfoViewController a zdrojový kód

InfoViewController slouží jako informační. Obsahuje plán celé budovy, vysvětlení názvů místností a zpětné tlačítko. Všechny tyto prvky jsem přidal do ViewController v .storyboardu, proto jsem je do kódu připojil pomocí outletů.

```
class infoViewController: UIViewController {

    //Outlety ze storyboard
    @IBOutlet weak var backView: UIView!

    @IBOutlet weak var planImageView: UIImageView!

    @IBOutlet weak var myLabel: UILabel!

    @IBOutlet weak var myTextView: UITextView!

    @IBOutlet weak var backButton: UIButton!
```

Ve funkci viewDidLoad nastavím vzhled zpětného tlačítka a přiřadím mu funkci, kterou má spustit, při jeho zmáčknutí. Dále je zde nastavení vzhledu backView, ve kterém jsou umístěné všechny prvky tohoto viewControlleru. Je to z toho důvodu, aby vyniklo to co má být vidět vůči částečně průhlednému pozadí. Dole se nachází znemožnění skrolování pro text, protože by nemělo být potřeba.

```
  override func viewDidLoad() {
      super.viewDidLoad()

      //Nastavení zpětného tlačítka
      backButton.addTarget(self, action: #selector(goBack), for: .touchUpInside)
      backButton.layer.masksToBounds = true
      backButton.backgroundColor = UIColor.cvutBlue
      backButton.layer.cornerRadius = 6
      backButton.setTitleColor(UIColor.white, for: .normal)

      //Nastavnení pozadí
      backView.layer.borderWidth = 1
      backView.layer.borderColor = UIColor.lightGray.cgColor
      backView.layer.cornerRadius = 6
      backView.layer.masksToBounds = true

      myTextView.isScrollEnabled = false //Znemožnění scrollování
  }
```

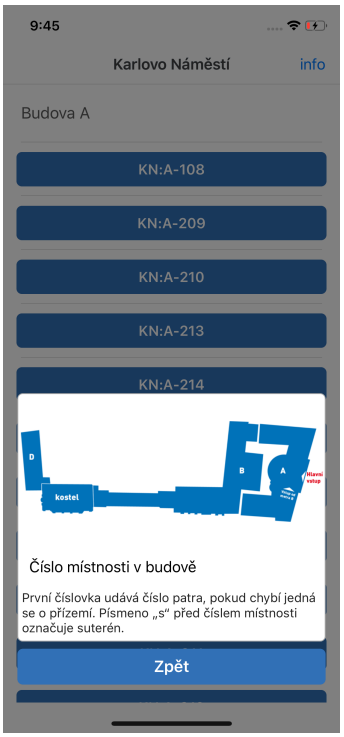
Tento UIViewController je zobrazen pomocí sequee „Present Modally“, proto je použita funkce dismiss. Aby byl přechod animovaný je zde animated true, avšak nic dalšího by se dít nemělo, proto je completion nil.

```
  //Navigační fukce
  @objc func goBack() {
      self.dismiss(animated: true, completion: nil)
  }
```

Pro postupné ztmavování barvy pozadí, je jeho barva nejprve úplně průhledná a potom se pomocí UIView.animate podle nastaveného času ztmavuje do výsledné barvy, která je v tomto případě tmavá, částečně průhledná.

```
//Tato funkce se uskuteční než se nám zobrazí obsah
override func viewWillAppear(_ animated: Bool) {
    view.backgroundColor = UIColor(red: 0, green: 0, blue: 0, alpha: 0.0) //Průhledné pozadí
    UIView.animate(withDuration: 1.2, animations: {
        self.view.backgroundColor = UIColor(red: 0, green: 0, blue: 0, alpha: 0.5) //Částečně průhledné pozadí
    })
}
```

Výsledek v simulátoru vypadá takto. Vzhledem k částečně průhlednému pozadí, lze stále vidět vstupní ViewController.



Obr. 16. Screenshot InfoViewControlleru ze simulátoru.

5.5 Další složky v aplikaci

Aplikace obsahuje ještě další složky, které plní určitou funkci a bez kterých by aplikace nemohla fungovat.

5.5.1 AppDelegate.swift

Tento soubor se vytvoří automaticky při vytvoření aplikace. Má dvě hlavní funkce.

- Definuje AppDelegate class, který vytváří okno s obsahem a poskytuje funkce pro zareagování a změnu stavu aplikace. Například při telefonátu se změní stav a pomocí funkce na to lze reagovat.

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
    [UIApplication.LaunchOptionsKey: Any]?) -> Bool { return true }
func applicationWillResignActive(_ application: UIApplication) {}
func applicationDidEnterBackground(_ application: UIApplication) {}
func applicationWillEnterForeground(_ application: UIApplication) {}
func applicationDidBecomeActive(_ application: UIApplication) {}
func applicationWillTerminate(_ application: UIApplication) {}
```

Obr. 17. Toto jsou funkce, které se v AppDelegate vyskytují. Podle jejich názvů si lze představit jejich význam

- Vytváří vstupní bod do aplikace a run loop, která koordinuje v jakém pořadí budou probíhat příchozí funkce

5.5.2 Extensions použité v aplikaci

Extensions nám umožňují rozšířit existující třídu, strukturu, protokol a další. Dají se umístit do nově vytvořené .swift složky, nebo přímo do nějaké viewcontroller.swift složky, mimo hlavní „class“. Tato aplikace obsahuje pouze dvě extensions.

První nám rozšiřuje UIColor o novou barvu, kterou jsem pojmenoval cvutBlue.
Tato barva odpovídá používané modré na webových stránkách fakulty strojní.

```
extension UIColor {

    class var cvutBlue: UIColor {
        return UIColor(red: 49/255, green: 112/255, blue: 182/255, alpha: 1)
    }

}
```

Použití je následovné.

```
cell.classroomLabel.backgroundColor = .cvutBlue                    //Barva pozadí
```

Druhá extension je složitější. Slouží k určení, o jaký model telefonu se jedná. Funguje tak, že rozšiřuje UIDevice, aby vracel název modelu telefonu. Vzhledem k tomu, že nové modely iPhonů mají vzhledem k chybějícímu dolnímu tlačítku jiné rozložení obrazovky, je potřeba u některých elementů vytvořených kódem upravit jejich velikost a umístění.

```
extension UIDevice {

    static let modelName: String = {
        var systemInfo = utsname()
        uname(&systemInfo)
        let machineMirror = Mirror(reflecting: systemInfo.machine)
        let identifier = machineMirror.children.reduce("") { identifier, element in
            guard let value = element.value as? Int8, value != 0 else { return identifier }
            return identifier + String(UnicodeScalar(UInt8(value)))
        }

        func mapToDevice(identifier: String) -> String {
            switch identifier {
            case "iPhone8,1":                                   return "iPhone 6s"
            case "iPhone8,2":                                   return "iPhone 6s Plus"
            case "iPhone9,1", "iPhone9,3":                      return "iPhone 7"
            case "iPhone9,2", "iPhone9,4":                      return "iPhone 7 Plus"
            case "iPhone8,4":                                   return "iPhone SE"
            case "iPhone10,1", "iPhone10,4":                    return "iPhone 8"
            case "iPhone10,2", "iPhone10,5":                    return "iPhone 8 Plus"
            case "iPhone10,3", "iPhone10,6":                    return "iPhoneX"
            case "iPhone11,2":                                  return "iPhoneXs"
            case "iPhone11,4", "iPhone11,6":                    return "iPhoneXsMax"
            case "iPhone11,8":                                  return "iPhoneXr"
            case "iPad3,4", "iPad3,5", "iPad3,6":               return "iPad 4"
            case "iPad4,1", "iPad4,2", "iPad4,3":               return "iPad Air"
            case "iPad5,3", "iPad5,4":                          return "iPad Air 2"
            case "iPad6,11", "iPad6,12":                        return "iPad 5"
            case "iPad7,5", "iPad7,6":                          return "iPad 6"
            case "iPad2,5", "iPad2,6", "iPad2,7":               return "iPad Mini"
            case "iPad4,4", "iPad4,5", "iPad4,6":               return "iPad Mini 2"
            case "iPad4,7", "iPad4,8", "iPad4,9":               return "iPad Mini 3"
            case "iPad5,1", "iPad5,2":                          return "iPad Mini 4"
            case "iPad6,3", "iPad6,4":                          return "iPad Pro 9.7 Inch"
            case "iPad6,7", "iPad6,8":                          return "iPad Pro 12.9 Inch"
            default:                                            return identifier
            }
        }
        return mapToDevice(identifier: identifier)
    }()
}
```

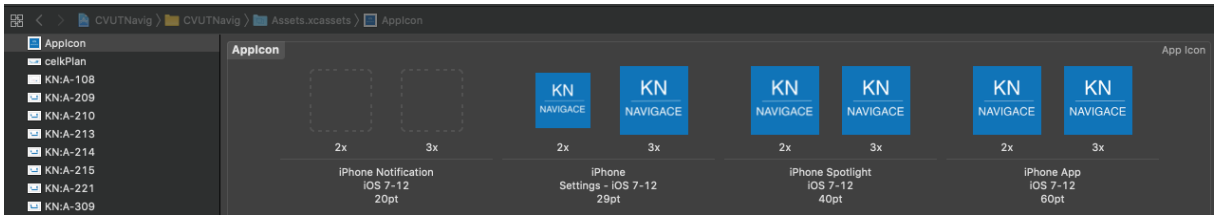
Použití vypadá následovně.

```
//Rozpoznání zařízení a následná úprava velikosti přechodú
if (UIDevice.modelName == "iPhoneXs" || UIDevice.modelName == "iPhoneXr" ||
    UIDevice.modelName == "iPhoneXsMax" || UIDevice.modelName == "iPhoneX") {
```

Dále upravím co je potřeba, aby byl splněn požadovaný vzhled.

5.5.3 Assets.xcassets

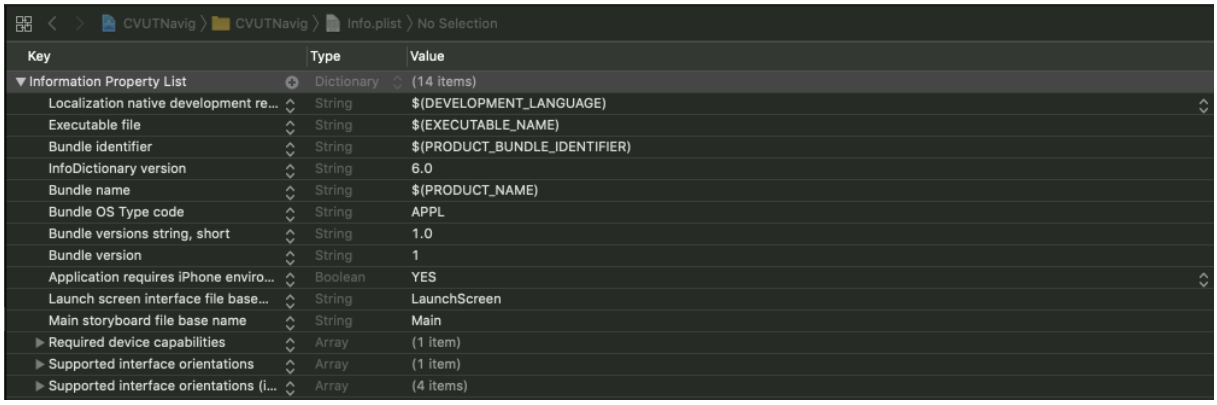
Zde se nacházejí jednotlivé obrázky s vyznačenými místnostmi a ikona aplikace, která se musí vložit v různých velikostech, podle toho na co se bude používat. Vzhledem k tomu, že tato aplikace neposílá žádná upozornění, může iPhone Notification ikona chybět, nebude totiž používána.



Obr. 18. Ukázka Assets.xcassets složky z CVUTNavig.

5.5.4 Info.plist

Tento soubor obsahuje základní nastavení, jako číslo verze, jaký storyboard je hlavní atd.



Obr. 19. Ukázka Info.plist složky z CVUTNavig.

Základní věci se nastaví automaticky. Důležitou úlohu má tento soubor, pokud chceme například získat uživatelovu polohu, nebo přístup ke kameře. Právě sem se nejprve musí umístit povolení a také důvod proč a na co přesně přístup chceme. Pokud se člověk pokusí získat přístup v kódu bez předchozího vložení do tohoto souboru, je aplikace zamítnuta. V poslední době se kontroluje i jaký má požadavek smysl a pokud není dostatečně specifikován, je aplikace také zamítnuta.

Zde je příklad žádosti o povolení přístupu do knihovny obrázků a přístupu ke kameře. Takto vypadá část Info.plist jako XML kód. Klíč je druh o jaké povolení se jedná a pod ním se nachází text který se uživatelovi zobrazí při zažádání o přístup.

```
<key>NSCameraUsageDescription</key>
<string>$(PRODUCT_NAME) requires to access your camera, if you want to take pictures with this app. </string>
<key>NSPhotoLibraryUsageDescription</key>
<string>$(PRODUCT_NAME) requires access to the photo library. To pick your new pictures.</string>
```

Obr. 20. Ukázka Info.plist v XML kódu.

$(PRODUCT_NAME) je název aplikace který si Info.plist vezme z nastavení aplikace. Název je definovaný v úvodním nastavení aplikace.

5.5.5 LaunchScreen.storyboard

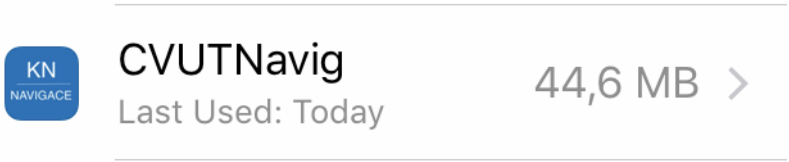
Vzhledem k tomu, že se aplikace nenastartuje po kliknutí na její ikonu okamžitě, je do projektu přidána složka LaunchScreen.storyboard, která se zobrazí okamžitě po zmáčknutí ikony a zmizí ve chvíli, kdy je aplikace připravena k fungování. Launch screen se nedá upravovat pomocí kódu. Lze do ní pouze vkládat objekty, které lze upravit pouze v základním nastavení.

Obr. 21. Ukázka launch screen ze simulátoru.

Pro tuto aplikace je launch screen pouze nastavená barva pozadí s vloženým obrázkem, ve kterém je znak FS CVUT.

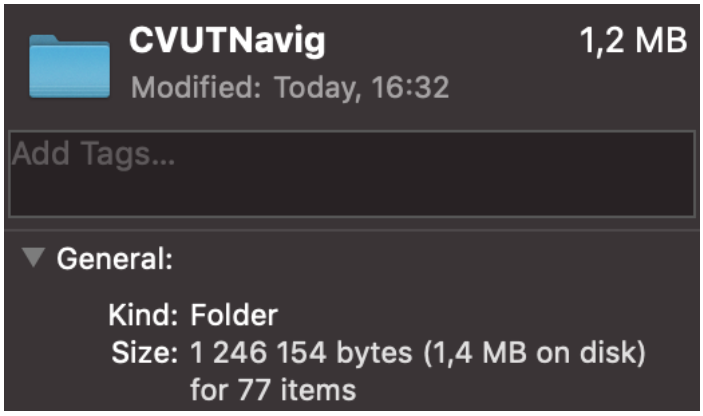
5.6 Velikost aplikace

CVUTNavig zabírá při nahrání do telefonu pomocí simulátoru 44,6 MB. To je na takto malou aplikaci poměrně dost, avšak pokud by byla aplikace nahrána na App Store, potom by po jejím stáhnutí to měla zabírat kolem 25 MB.



Obr. 22. Velikost aplikace na reálném zařízení.

Na disku zabírá složka s aplikací 1,4 MB. Z toho 1,1 MB obrázky budovy a zbytek další složky, jako .swift soubory, jejichž velikost se pohybuje řádově v jednotkách KB.

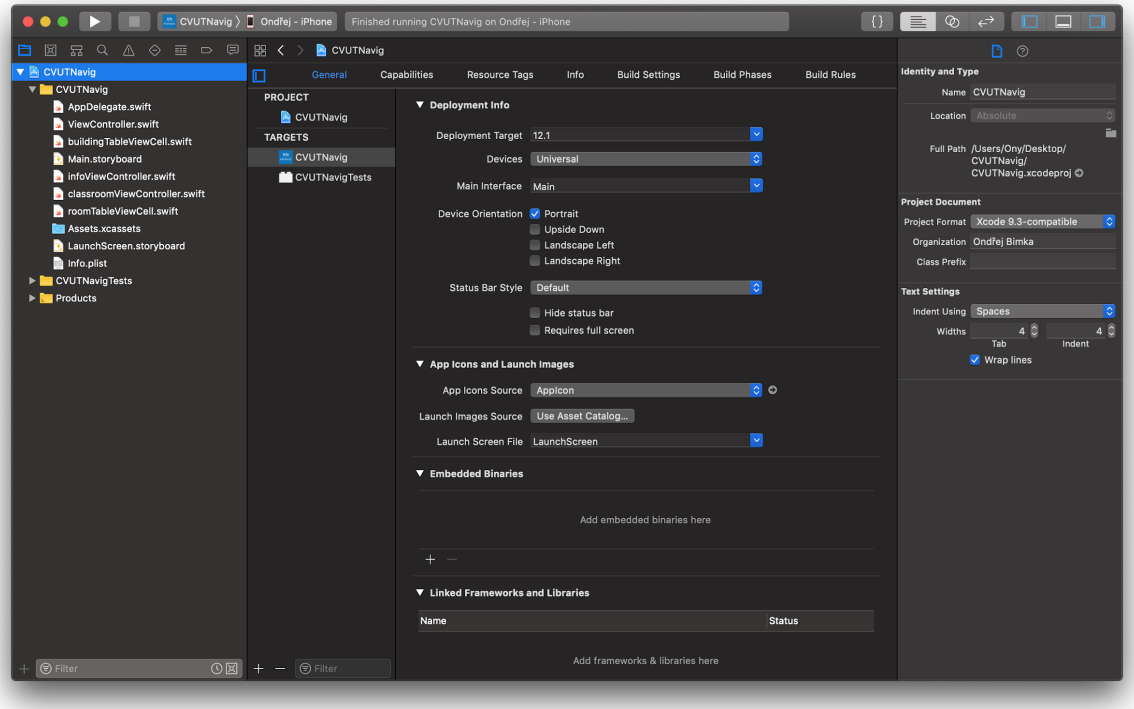


Obr. 23. Velikost složky aplikace v počítači.

5.7 Další možnosti nastavení v aplikaci

Pokud klikneme na soubor projektu, zobrazí se nám další možnosti nastavení. V případě CVUTNavig nebylo třeba nic přenastavovat. Vše bylo nastaveno správně.

Tato nastavení se upravují, pokud používáme nějaké frameworky, při nastavení bridging-headeru, nebo nějakého third-party softwaru. Většina věcí se však nastavuje automaticky, proto je lepší tato nastavení spíše neměnit.



Obr. 24. Ukázka možnosti dalšího nastavení.

6. Závěr

Cílem této práce bylo seznámit čtenáře s aplikací CVUTNavig a vysvětlit základní postup při jejím vytváření. Tento postup by měl být obecně aplikovatelný na jakoukoli aplikaci.

Snažil jsem se aby byla aplikace co nejnázornější a nejpřehlednější. Doufám, že po přečtení této práce si čtenář udělá obrázek o fungování a vytváření aplikace. Jako dokumentaci jsem používal hlavně oficiální podklady od Apple.

CVUTNavig byla naprogramována programovacím jazykem Swift za použití programu Xcode. Věřím, že poznatky získané při vytváření této bakalářské práce využiji i ve svém budoucím životě, či zaměstnání.

Tato aplikace by byla také vhodná k rozšíření do všech budov fakulty strojní. Dále by se dal k místnostem přidat i jejich rozvrh, nebo detailněji zobrazit navigaci k místnostem.

7. Použité zdroje

[1] KEUR, Christian a Aaron HILLEGASS. *IOS programming: the Big Nerd Ranch guide*. Sixth edition. Atlanta, GA: Big Nerd Ranch, [2016]. ISBN 978-0134682334.

[2] Jump Right In. *Start Developing iOS Apps (Swift)* [online]. San Francisco (CA): Apple, c2018 [cit. 2019-04-6].Dostupné z: https://developer.apple.com/library/archive/referencelibrary/GettingStarted/DevelopiOSAppsSwift/index.html#//apple_ref/doc/uid/TP40015214-CH2-SW1

[3] Swift. *Welcome to Swift.org* [online]. San Francisco (CA): Apple, c2019 [cit. 2019-04-6]. Dostupné z: https://swift.org

Obsah přiloženého CD:
- Text bakalářské práce ve formátu PDF
- Návod k vytvořené aplikaci
- Zdrojové texty vytvořené aplikace
- PowerPoint soubor s prezentací