Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Software pro sledování osobní knihovny

2017

Jiří Budík

Vedoucí práce: Mgr. Tomáš Kühr, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Jiří Budík
Název práce:	Software pro sledování osobní knihovny
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2017
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Tomáš Kühr, Ph.D.
Počet stran:	37
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Jiří Budík
Title:	Personal Library Tracker
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2017
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Tomáš Kühr, Ph.D.
Page count:	37
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Práce se zabývá nástroji určenými k vývoji mobilních aplikací pro operační systém iOS. Dále také tvorbou konkrétní aplikace pro účely sledování osobní knihovny a evidence vypůjčených knih přátelům s možností získat informace o knize z veřejně dostupných databází pomocí čtečky čárových kódů. Nakonec popisuje použití výsledné aplikace.

Synopsis

The thesis deals with the tools for development of mobile applications for the iOS operating system. In addition, it also deals with creating a specific application for tracking a personal library and lending books to friends with the ability to get information about a book from publicly accessible databases using a barcode reader. Finally, it describes the use of the resulting application.

Klíčová slova: iOS; aplikace; osobní knihovna; čtečka čárových kódů

Keywords: iOS; application; personal library; barcode reader

Rád bych poděkoval Mgr. Tomáši Kührovi, Ph.D. za trpělivost, cenné rady a připomínky v průběhu vypracování bakalářské práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Tato práce se zabývá tvorbou aplikace pro mobilní platformu iOS firmy Apple. V první části je představen operační systém iOS a jednotlivé vrstvy, ze kterých se skládá. Další část se zabývá některými běžnými nástroji určenými pro vývoj a distribuci aplikací a seznamuje čtenáře se základními prvky programovacího jazyka Swift. Dále jsou popsány dostupné knihovny a některé jejich části, které jsou použity při vývoji konkrétní aplikace. Následuje příručka pro programátora zachycující postup tvorby konkrétní aplikace a část, která popisuje její možnosti a chování z pohledu uživatele. Na závěr jsou zmíněny některé náměty na možná vylepšení aplikace.

1.1 Cíle práce

Cílem práce je navržení a realizace nástroje pro správu osobní knihovny uživatele a sledování výpůjček knih přátelům či členům širší rodiny. Software má být implementován jako aplikace pro zařízení iPhone. Práce s aplikací má být usnadněna možností využívat čtečku čárových kódů a pohodlně vyhledávat knihy ve veřejně dostupných databázích.

2 Systém iOS

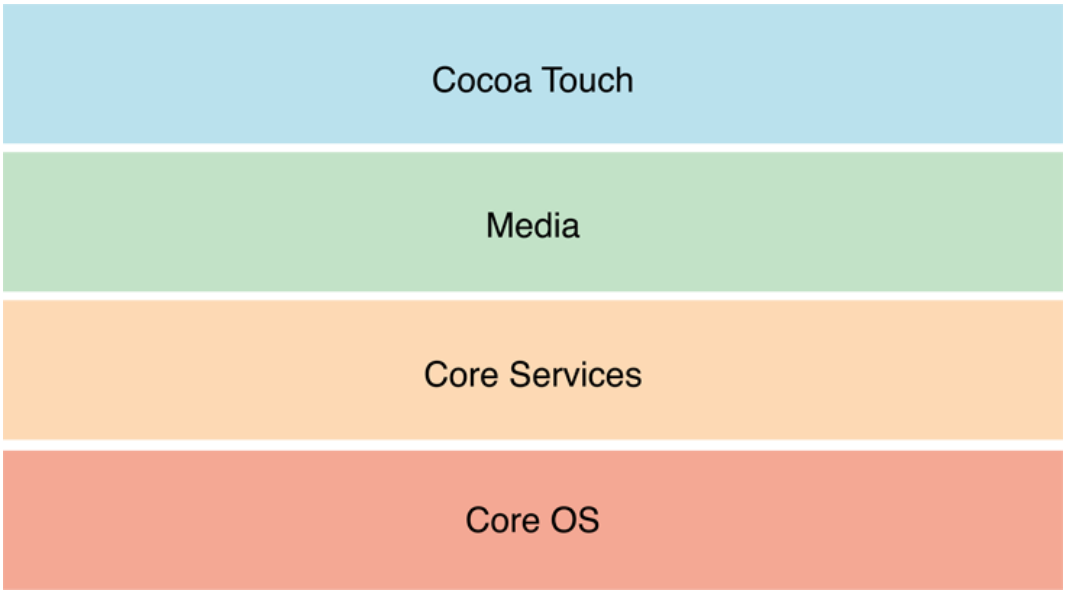
Tato kapitola pokrývá základní informace o systému a popisuje jednotlivé vrstvy jeho architektury.

2.1 Základní popis

iOS je operační systém vyvinutý firmou Apple. Jeho první verze byly vydávány od roku 2007 pod názvem iPhone OS a od roku 2010 až dodnes nese název iOS[19]. Oproti desktopovému macOS se jedná o odlehčenou verzi systému určenou pro mobilní zařízení. Neobsahuje veškerou funkcionalitu macOS, ale přidává podporu dotykového ovládání a běží na zařízeních iPad, iPhone a iPod, které pochází taktéž z dílny Apple. Z toho plyne jistá výhoda firmy odladit operační systém pro konkrétní produkty.

2.2 Architektura iOS

Aplikace nepřistupují k hardwaru přímo, ale komunikují skrze systémová rozhraní, na které lze nahlížet jako na čtyři základní vrstvy (znázorněno na obrázku 1). O těch pojednávají následující podkapitoly.



Obrázek 1: Vrstvy iOS [1]

2.2.1 Core OS

Core OS je jádrem systému a zároveň nejníže položenou vrstvou poskytující nízkoúrovňové funkce většině ostatním technologiím, které na ní leží. I když nejsou většinou v aplikacích využívány přímo, velmi pravděpodobně je využívají vysokoúrovňové komponenty systému. Vrstva se stará především o správu paměti, souborový systém, bezdrátovou komunikaci, konektivitu skrze Bluetooth nebo Lightning konektor či třicetipinový konektor[2]. Dále o vestavěné bezpečnostní prvky jako jsou certifikáty, veřejné a soukromé klíče, ověření uživatele pomocí Touch ID[1] nebo generování kryptografických pseudonáhodných čísel[2].

2.2.2 Core Services

Tato vrstva poskytuje služby vyšší úrovně. Její frameworky obstarávají základní práci s datovými typy, URL, vlákny, porty atd. Dále například zajišťuje funkcionalitu lokačních služeb, uložiště iCloud[2], In-App[3] nákupů, podporu zpracování XML dokumentů a SQLite databázi[3]. Pod tuto vrstvu spadá přístup k databázi kontaktů uživatele a také ukládání strukturovaných dat a mapování na programové objekty, tzv. Core Data[3], které jsou v rámci naší práce hojně využívány.

2.2.3 Media

Media vrstva obecně zahrňuje grafické, zvukové a obrazové technologie. Pod grafické technologie spadá pokročilá podpora animací, hardwarově akcelerované vy-

[1]Touch ID - technologie na rozpoznávání otisků prstu
[2]iCloud - cloudová služba od společnosti Apple
[3]In-App - nákupy uvnitř aplikace

kreslování 2D a 3D objektů, čtení a zápis většiny rozšířených grafických formátů, engine pro vykreslování textu[4]. Zvukové technologie poskytují sadu rozhraní pro správu přehrávání a záznam zvuku, umožňují přehrávat systémové zvuky a upozornění, vibrovat a přehrávat vícekanálový či streamovaný zvuk nebo také poskytují přístup k iTunes knihovně a přehrávání skladeb[4]. Frameworky pro obraz pak umožňují přístup k obrázkové knihovně uživatele, přehrávání videí přes celou obrazovku nebo jen částečně a o záznam obrazu se stará rozhraní AV Foundation[4], které zajišťuje jednu z hlavních funkcionalit v rámci této práce.

2.2.4 Cocoa Touch

Před samotnou aplikací se na nejvyšší úrovni nachází poslední vrstva Cocoa Touch, která obsahuje nejdůležitější frameworky včetně knihovny UIKit. Ta je společně s knihovnou Foundation jednou z nejdůležitějších pro vývoj aplikací a poskytuje infrastrukturu pro implementaci grafického rozhraní aplikace a interakci s uživatelem[5]. Vrstva zahrnuje další vysokoúrovňové služby, mezi které patří multitasking, šifrování souborů, notifikační centrum, rozpoznávání gest, Auto Layout[4], AirDrop[5], Handoff[6], používání mapové komponenty a standardní systémová zobrazení určená pro náhled a úpravu kontaktů, vkládání událostí do kalendáře, psaní e-mailu či SMS, náhled a otevření souboru, pořízení fotky nebo videa či výběr z fotogalerie[5].

3 Vývojářské nástroje

Spolu s iOS zařízeními jsou od roku 2008 vydávány softwarové balíčky určené pro vývojáře, takzvané Software Development Kit[20]. Slouží k vytváření, instalaci, spouštění a testování nativních aplikací pro iOS a to pomocí obsažených frameworků a použitím příslušného programovacího jazyka. Jím byl dříve především Objective-C[7], ale v dnešní době je velmi upřednostňován jazyk Swift, který firma Apple představila v roce 2014.

Součástí balíčku je i vývojové prostředí Xcode, kterému budou věnovány následující podkapitoly včetně způsobu distribuce hotové aplikace a závěr této části bude patřit základní syntaxi programovacího jazyka Swift.

3.1 Xcode

Vývojové prostředí Xcode je úzce spjato s Cocoa[8] a Cocoa Touch frameworkem a poskytuje velmi přívětivé prostředí pro vytváření aplikací pro zařízení Mac, iPhone, iPad, Apple Watch a Apple TV[6]. Nabízí intuitivní grafický editor pro

[4]Auto Layout - způsob, jakým je definováno dynamické uživatelské rozhraní
[5]AirDrop - služba pro přenos souborů mezi Apple zařízeními pomocí Wi-Fi nebo bluetooth
[6]Handoff - služba pro sdílení dat a úloh některých aplikací mezi Apple zařízeními
[7]Objective-C - objektově orientovaný jazyk implementovaný jako rozšíření jazyka C
[8]Cocoa - sada frameworků zajišťující běhové prostředí pro aplikace v systému macOS

tvorbu uživatelského rozhraní. Přitom lze pomocí asistenčního editoru v podokně sledovat a upravovat příslušný zdrojový kód. Umožňuje táhnutím myši jednoduše propojovat ovládací prvky uživatelského rozhraní s implementačním kódem[6].

3.2 Interface Builder

Editor Interface Builder, který je vestavěný v Xcode, usnadňuje návrh úplného uživatelského rozhraní bez nutnosti psaní jakéhokoli kódu. Umožňuje přesouvat okna, tlačítka, textová pole a další objekty přímo na plátno návrhu a vytvořit tak funkční uživatelské rozhraní. Vzhledem k tomu, že Cocoa a Cocoa Touch jsou navrženy pomocí architektury Model-View-Controller, je možné nezávisle navrhovat uživatelská rozhraní odděleně od jejich implementace.[7] Nyní se zaměříme na pár důležitých prvků Interface Builderu.

3.2.1 Storyboards

Každý projekt iOS aplikace tvořené v Xcode automaticky obsahuje soubor Storyboard zachycující celkový obraz aplikace. Základní stavební jednotkou pro prvky uživatelského rozhraní v iOS je „View Controller“ jakožto objekt třídy `UIViewController`[8] nebo jeho potomci, kterými jsou například „Table View Controller“ a „Collection View Controller“ sloužící pro zobrazování určité sady dat či „Tab Bar Controller“[7] pro přepínání mezi jednotlivými „View Controllery“. Hlavní náplní Storyboard je znázornit, jaké typy „View Controllerů“ jsou použity, a jakým způsobem jsou mezi sebou propojeny[8]. V rámci Storyboard se do jednotlivých „View Controllerů“ bežně umisťují ostatní objekty uživatelského rozhraní, kterými jsou například tlačítka, štítky nebo textová pole[8] (znázorněno na obrázku 2 vlevo).

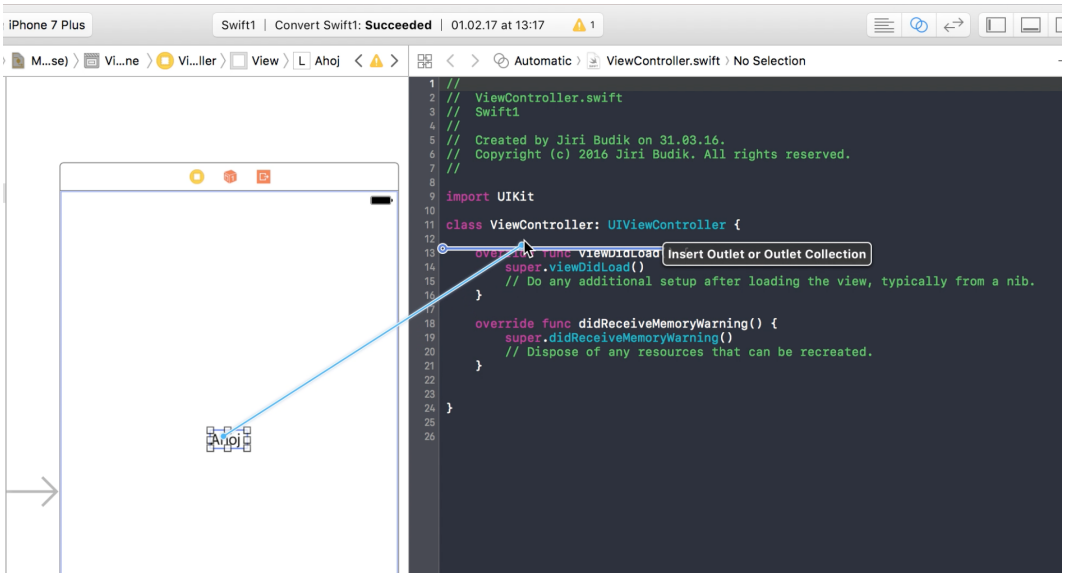
3.2.2 Auto Layout

Auto Layout neboli automatické rozvržení je založeno na představě, že každý objekt uživatelského rozhraní může mít definovaná takzvaná „constraints“ tedy omezení změny zobrazení, v reakci na změny nadřazeného zobrazení nebo dalších prvků uživatelského rozhraní[7]. Nastavená omezení hrají také velkou roli při změně velikosti displeje, anebo při změně orientace displeje z portrétového na „landscape“ mód. Pomocí Interface Builderu lze automaticky vytvořit všechna „constraints“, která zajistí sadu kompatibilních pravidel pro správné zobrazení[7].

3.2.3 Assistent editor

Pomocný „Assistant“ editor je okno v Interface Builderu, které zobrazuje zdrojový kód při práci se Storyboard a umožňuje napojit prvek uživatelského rozhraní přetažením myši do místa zdrojového kódu v editoru pro implementaci jeho chování[8] (znázorněno na obrázku 2 vpravo). Xcode navíc nabídne vytvořit „action“, tedy metodu, kterou bude prvek spouštět nebo takzvaný „outlet“, což

je proměnná navázaná na prvek pro uchování dat[7]. Tímto způsobem může být poskytnuta logika pro uživatelské rozhraní[7].



Obrázek 2: Storyboard a Assistent editor

3.3 iOS Simulator

Díky Software Development Kit lze v Xcode použít simulátor iOS zařízení a pomocí něj vytvářet, instalovat, spouštět a odlaďovat aplikace a tím zefektivnit proces vývoje[17]. Xcode tedy umožňuje používat vytvářenou aplikaci na libovolných typech zařízení iPhone nebo iPad, aniž by byly fyzicky připojené k počítači. Simulátor pochopitelně neumožňuje telefonovat nebo používat fotoaparát, ale nabízí spoustu užitečných prvků, jako jsou například simulace doteku či fyzického otočení, přístup do fotogalerie, simulace geografické polohy, simulace nedostatku paměti a především může vývojáři sloužit k odhalování chyb či nedostatků uživatelského rozhraní[8].

3.4 App Store a distribuce aplikace

Obchod App Store pro iOS je platforma pro šíření digitálního obsahu vytvořená firmou Apple určená pro mobilní zařízení s operačním systémem iOS. Umožňuje uživatelům vyhledávat a stahovat aplikace vytvořené pomocí vývojářského balíčku pro iOS. App Store byl poprvé spuštěn v roce 2008 a nabízel zhruba 500 aplikací. Počátkem roku 2017 obsahoval už přes 2 miliony dostupných aplikací. Vývojářům také nabízí možnost stanovit si vlastní ceny svých aplikací, nebo je nabízet zdarma, anebo využít variantu in-app nákupů.[16]

Pro běžného uživatele existuje jediná legální cesta, jak si stáhnout do svého zařízení aplikaci a to je právě prostřednictvím obchodu App Store. Z pohledu

vývojáře se zároveň jedná o nejsnazší způsob, jak distribuovat aplikaci směrem k uživateli. Jakmile je aplikace dokončena, prostředí Xcode ji umožňuje nahrát přímo na App Store. Celkový proces distribuce je však ve skutečnosti o něco delší, a je třeba splnit několik náležitostí, které budou nastíněny v následujících krocích. Za předpokladu, že programátor má v projektu nahrané všechny požadované rozměry ikony aplikace a neplánuje provádět už žádné změny v kódu, je dále zapotřebí:

1. Mít vytvořený placený vývojářský účet, přihlásit se do něj na stránkách https://developer.apple.com a přejít do sekce *Certificates, Identifiers and Profiles*.

2. Vytvořit *App ID*, kde je třeba vyplnit *Bundle ID*, které je uvedeno v Xcode v nastavení projektu v sekci *Identity*.

3. V aplikaci *Keychain Access* si přes průvodce vyžádat certifikační klíč od certifikační autority a uložit ho na disk.

4. Ve vývojářském účtu vytvořit certifikát typu *iOS App Development*. Při vytváření bude požadováno nahrání certifikátu z předchozího bodu. Vygeneruje se certifikát, který stačí stáhnout a otevřít v aplikaci *Keychain Access*.

5. Vytvořit certifikát typu *App Store and Ad Hoc*. Při vytváření lze opět z disku nahrát certifikát z bodu 3. Vygeneruje se certifikát, který stačí stáhnout a otevřít v aplikaci *Keychain Access*.

6. Vytvořit *Provisioning Profile* typu *App Store*. Při vytváření je třeba z nabídky zvolit *App ID* z bodu 2 a v dalším kroce zvolit certifikát z bodu 5.

7. Přihlásit se na stránkách https://itunesconnect.apple.com, založit novou aplikaci a vyplnit veškeré požadované údaje a uložit.

8. V nastavení Xcode v sekci *Accounts* je třeba mít přidaný vývojářský účet z bodu 1. V nastavení projektu v sekci *Signing* je třeba zaškrnout *Automatically manage signing* a za tým mít vybraný název vývojářského účtu.

9. V Xcode přepnout zařízení iOS simulátoru na *Generic iOS Device* a v horní nabídce *Product* zvolit možnost *Archive*. Po archivaci je třeba provést validaci pomocí *Validate*. Po úspěšné validaci už stačí zvolit *Upload to App Store*.

10. Vrátit se na https://itunesconnect.apple.com a v nastavení založené aplikace v sekci *Build* vybrat nahranou aplikaci, která je výsledkem předešlého kroku.

11. Na závěr kliknout na *Submit for Review*.

Po splnění poslední kroku už zbývá jen čekat na vyjádření ze strany Apple, která buď aplikaci schválí, anebo požádá autora o provedení konkrétních změn. Proces schvalování běžně probíhá v řádu několika dnů.

3.5 Swift

Následující podkapitoly obsahují několik informací sloužících k seznámení čtenáře s některými základními rysy programovacího jazyka, který byl použit v rámci této práce.

3.5.1 Základní popis

Swift je programovací jazyk od společnosti Apple, který je využíván při vývoji na platformách macOS a iOS a od prosince roku 2015 je navíc open source[9][18]. Jedná se o multi-paradigmový jazyk, který ke kompilaci používá LLVM[10] a umožňuje v rámci jednoho programu použít kromě Swiftu i další jazyky jako jsou C, Objective-C či Objective-C++[10]. Swift byl původně zamýšlen jako náhrada za zmíněný Objective-C se snahou eliminovat některé chyby ze strany programátora, a proto, narozdíl od Objective-C, nepodporuje například práci s ukazateli a pro práci s pamětí je zde využíváno automatického počítání referencí (ARC)[10].

3.5.2 Syntaxe

Syntaxe je místy podobná jazykům C nebo Objective-C. Není potřeba speciálně importovat vstupně/výstupní knihovny nebo knihovny obstarávající práci s řetězci. Odpadá také nutnost použít funkci *main* a nemusí se ani používat středník na konci každého příkazu.[9]

Nyní se zaměříme na základní konstrukce jazyka. Zdrojem většiny následujících informací je dokumentace dostupná z [9]. Pro konstanty slouží klíčové slovo `let` a pro proměnné se používá `var`. Hodnota konstanty nemusí být známa v době kompilace, ale může jí být přiřazena pouze jednou. Konstanta i proměnná musí být stejného typu jako jim přiřazená hodnota, přičemž typ může být zadán explicitně, anebo automaticky díky takzvané typové inferenci. Hodnoty nelze implicitně přetypovávat. Je třeba explicitně vytvořit novou instanci kýženého typu. Pro tvorbu polí a asociativních polí se používají hranaté závorky a přes ně se také přistupuje k jejich prvkům pomocí indexu nebo klíče.

K vytváření podmínek se používá `if` a `switch` a ke tvorbě cyklů slouží `for-in`, `while` a `repeat-while`. Podmínka nebo řídící proměnná v cyklu nemusí být ohraničena kulatými závorkami, ale jejich těla musí být ve složených závorkách. Lze použít spojení `if let` při práci s hodnotami, které mohou chybět. Takové hodnoty jsou nazývané „optionals“ a běžně se označují otázníkem na konci jejich typu. Platí, že buď obsahují hodnotu, anebo obsahují `nil` indikující chybějící hodnotu. Dalším způsobem práce s „optional“ hodnotami je použití

[9] Open source - otevřený zdrojový kód
[10] LLVM (Low Level Virtual Machine) - implementace optimalizujícího překladače

```
1  //deklarace proměnné a konstanty
2  var myVariable = 42
3  myVariable = 50
4  let myConstant = 42
5
6  //implicitní a explicitní deklarace typu
7  let implicitDouble = 70.0
8  let explicitDouble: Double = 70
9
10 //práce s polem a asociativním polem
11 var shoppingList = ["catfish", "water", "tulips", "blue paint"]
12 shoppingList[1] = "bottle of water"
13
14 var occupations = ["Malcolm": "Captain", "Kaylee": "Mechanic"]
15 occupations["Jayne"] = "Public Relations"
```

Zdrojový kód 1: Ukázka základní syntaxe jazyka Swift (1)

operátoru `??`, kde se v případě chybějící hodnoty použije výchozí hodnota za operátorem.

```
1  //procházení pole cyklem a větvení pomocí podmínky
2  let individualScores = [75, 43, 103, 87, 12]
3  var teamScore = 0
4  for score in individualScores {
5     if score > 50 {
6           teamScore += 3
7     } else {
8           teamScore += 1
9     }
10 }
11 print(teamScore) //11
12
13 //použití optional hodnoty a výchozí hodnoty
14 let nickName: String? = nil
15 let fullName: String = "John Appleseed"
16 let informalGreeting = "Hi \(nickName ?? fullName)"
```

Zdrojový kód 2: Ukázka základní syntaxe jazyka Swift (2)

Funkce se deklarují pomocí klíčového slova `func`. Následuje seznam parametrů v kulatých závorkách. V případě, že má funkce návratovou hodnotu, tak se ještě deklaruje její typ a mezi typem a seznamem paramatrů je použit oddělovač v podobě šipky ->. Pomocí takzvaných „tuple“ může funkce vracet výsledek složený z více hodnot. K jednotlivým hodnotám lze pak přistupovat přes klíč nebo index. Swift dále podporuje vnořené funkce nebo funkce vyššího řádu.

Pro vytváření tříd slouží klíčové slovo `class` následováno názvem třídy. Deklarace vlastností se provádí stejně jako u konstant a proměnných s rozdílem, že

```
1  //funkce vracející n-tici hodnot a způsob, jak se k nim přistupuje
2  func calculateStatistics(scores: [Int]) -> (min: Int, max: Int, sum:
       Int) {
3     var min = scores[0]
4     var max = scores[0]
5     var sum = 0
6
7     for score in scores {
8        if score > max {
9           max = score
10       } else if score < min {
11          min = score
12       }
13       sum += score
14    }
15    return (min, max, sum)
16 }
17
18 let statistics = calculateStatistics(scores: [5, 3, 100, 3, 9])
19 print(statistics.sum) //120
20 print(statistics.2) //120
```

Zdrojový kód 3: Ukázka základní syntaxe jazyka Swift (3)

to probíhá v kontextu třídy. Stejné je to v případě metod a funkcí. K inicializaci objektů slouží konstruktor s názvem `init`. Při dědění tříd se vedle názvu třídy napíše ještě název jejího rodiče oddělený dvojtečkou. Zděděné metody lze následně přepisovat pomocí klíčového slova `override`. Instance třídy se vytváří přidáním kulatých závorek na konec názvu třídy. K vlastnostem a metodám instance se pak přistupuje pomocí tečkové notace.

4 Aplikace

Primárním cílem práce bylo vytvoření aplikace pro platformu iOS, která slouží k evidenci osobní knihovny a sledování výpůjček přátelům. Tato sekce nejprve pojednává o vybraném názvu naší aplikace a poté se zabývá popisem frameworků, na kterých je postavena.

4.1 Volba názvu

Jak je známo, anglický jazyk je celosvětově považován za mezinárodní jazyk, a tak první a zatím jediná verze naší aplikace byla tvořena s anglickou lokalizací. Z toho vyplývá i zvolený vystihující název aplikace, kterým je *Personal Library*, což lze do češtiny přeložit jako *Osobní Knihovna*. V kapitole 3.4 je uvedeno, že obchod App Store obsahuje více jak 2 miliony aplikací. Přitom platí, že každá musí mít unikátní název a *Personal Library* se tam dosud nevyskytl, čehož bylo

využito při registraci názvu v rámci nahrávání aplikace do App Store.

4.2 Použité knihovny

Následující podkapitoly jsou věnovány pomyslným stavebním kamenům naší aplikace, které představují standardně dostupné knihovny od firmy Apple. Budou popsány některé jejich vybrané části, které byly použity při tvorbě aplikace.

4.2.1 Foundation

Tato knihovna zajišťuje přístup k základním datovým typům, kolekcím a službám operačního systému a tím definuje základní vrstvu funkčností pro aplikaci. Foundation framework se dále stará o správné fungování ukládání dat, zpracování textu, práce s datem a časem, třídění a filtrace či síťových služeb. Třídy, protokoly a datové typy, obsažené v této knihovně, jsou používány při vývoji napříč systémy macOS, iOS, watchOS a tvOS, jak uvádí [11].

Foundation tedy obsahuje třídy a struktury pro práci se základními datovými typy jako je například `Int` nebo `Double`. Dále zahrnuje tvorbu a práci s textovými řetězci s podporou sady Unicode[11]. V rámci kolekcí obstarává pole, asociativní pole, množiny a další specializované kolekce pro procházení skupiny objektů či hodnot. Pro filtraci poskytuje například třídu `NSPredicate` a pro třídění `NSSortDescriptor`. Dále třídy a struktury pro interakci s URL adresami a komunikaci se servery s použitím standardních internetových protokolů. Podporuje také nástroje pro archivaci a serializaci dat. Zmiňme například třídu `JSONSerialization`, která má na starosti serializaci dat ve formátu JSON[12].

4.2.2 UIKit

UIKit framework poskytuje potřebnou infrastrukturu pro iOS nebo tvOS aplikace. Pomocí architektury oken a zobrazení zajišťuje implementaci uživatelského rozhraní, obsluhu událostí vyvolaných pomocí Multitouch[13] a dalších typů uživatelských vstupů a především zajišťuje hlavní nekonečnou smyčku potřebnou pro správu interakcí mezi uživatelem, systémem a aplikací[12]. Mezi další funkce, které framework nabízí, patří podpora animací, kreslení a obrazového výstupu, podpora informací o aktuálním zařízení, podpora vyhledávání, podpora zpřístupnění nebo správa zdrojů[12].

UIKit nám tedy poskytuje veškerá „Views“ neboli zobrazení, díky kterým je umožněno zobrazovat obsah na obrazovce a tím usnadnit interakci s uživatelem a k tomu ještě takzvané zobrazovače neboli „View Controllers“, které pomáhají spravovat jednotlivá zobrazení a strukturu uživatelského rozhraní.

Zahrnuje třídy `UIViewController` poskytující infrastrukturu pro správu zobrazení naší aplikace, `UINavigationController` řídící navigaci v rámci

[11] Unicode - norma pro kódování znaků použitelná pro většinu písem
[12] JSON - datový formát určený pro přenos dat, který je platformově nezávislý
[13] Multitouch - schopnost snímacího zařízení zachtit více dotyků najednou

hierarchického obsahu zobrazovačů, `UITabBarController` sloužící jako řadič zobrazení, které spravuje rozhraní pomocí radio-style[14] výběru. Dále třídy `NSLayoutAnchor` a `NSLayoutConstraint` udávající vztah mezi dvěma prvky uživatelského rozhraní, použité při rozvržení zobrazení technologií Auto Layout, `UIImagePickerController`, která poskytuje základní a omezené (v porovnání s knihovnou AVFoundation) možnosti pořízení nebo výběru snímků a videí z fotogalerie zařízení, `UIAlertController` zobrazující dialogová okna, `UITapGestureRecognizer` reagující na jeden či více doteků ze strany uživatele, `UILabel` zobrazující text, který běžně slouží jako popisek nějakého ovládacího prvku, `UITextField` zobrazující editovatelný text, `UIButton` jakožto ovládací prvek, jehož výběrem lze spustit nějakou část kódu, `UIImageView` sloužící k zobrazení jednoho obrázku nebo sekvence obrázků, `UIPickerView` používající metaforu hracího automatu k zobrazení jedné nebo více sad hodnot. Ještě zmiňme `UICollectionViewController` reprezentující View Controller, jehož obsah se skládá z objektu třídy `UICollectionView`. Ta slouží k zobrazení setříděné kolekce dat a má mnoho společného se třídou `UITableView`, ale poskytuje navíc například horizontální posouvání a celkově více možností pro rozložení zobrazených buněk.

4.2.3 Core Data

Pomocí Core Data lze spravovat objekty reprezentující model z Model-View-Controller architektury. Poskytuje ucelená a automatizovaná řešení pro běžné úkony spojené s životním cyklem objektu, diagramem tříd a persistentním uložením dat[13]. Core Data výrazně snižuje množství potřebného kódu k zajištění vrstvy, která se stará o zmíněný model, což je způsobeno díky následujícím vestavěným funkcím, které už není třeba implementovat, testovat nebo optimalizovat[13]:

- Propagace změn včetně zachování konzistence vztahů mezi objekty.
- Líné vyhodnocování objektů a sdílení dat pomocí metody copy-on-write[15] ke snížení režijních nákladů.
- Automatické ověřování hodnot u vlastností. Spravované objekty mají rozšířené standardní zakódování klíč-hodnota o metody, které ověřují, zda jsou jednotlivé hodnoty v přípustném rozmezí.
- Seskupování, filtrace a organizace dat v paměti a v uživatelském rozhraní.
- Podpora pro ukládání objektů do externích datových uložišť.
- Namísto psaní SQL dotazů je možné vytvářet komplexní dotazy s použitím třídy `NSPredicate` v rámci požadavku pro načtení objektů.

[14] Radio-style - používá přepínací tlačítka pro výběr jediné možnosti
[15] Copy-on-write - technika při které dochází k duplikaci dat až v okamžiku jejich změny

Knihovna například poskytuje třídu `NSManagedObject` implementující veškeré základní chování, které vyžaduje objekt modelu Core Data. Dále třídy `NSFetchedResultsController`, jejíž objekt se používá ke správě výsledků načtených z Core Data a zobrazení dat uživateli, `NSFetchRequest` popisující kritéria pro načítání dat z persistentního uložiště, `NSPersistentContainer` reprezentující kontejner, který zapouzdřuje uložiště Core Data v rámci aplikace, `NSManagedObjectContext` sloužící k provádění změn persistentně uložených objektů včetně jejich vytváření a odstraňování.

4.2.4 AVFoundation

Framework AVFoundation kombinuje čtyři hlavní technologické oblasti, které dohromady zahrnují širokou škálu úkolů pro zachycování, zpracovávání, slučování, kontrolu, import a export audiovizuálních médií na platformách společnosti Apple[14].

V rámci naší práce byla využita oblast knihovny, která umožňuje snímání obrazu skrze vestavěný fotoaparát. K tomu bylo použito několik následujících tříd. `AVCaptureDevice` má na starosti snímání vstupu a nabízí ovládací prvky pro hardwarově specifické funkce zachycení. Objekt třídy `AVCaptureSession` obstarává aktivitu zachycení a koordinuje tok dat ze vstupního zařízení do zvoleného místa pro uložení zachyceného výstupu, kterým je v našem případě objekt třídy `AVCaptureMetadataOutput` sloužící jako záznamový výstup pro zpracování časově označených metadat vytvořených v rámci zachytávací relace. Pak lze z těchto metadat pomocí `AVMetadataMachineReadableCodeObject` detekovat čárový kód. Ještě zmiňme, že společně s `AVCaptureSession` se používá třída `AVCaptureVideoPreviewLayer`, která obstarává zobrazení zachytávaného obrazu.

4.2.5 ContactsUI

Knihovna ContactsUI poskytuje nástroje, které usnadňují zobrazování, úpravu, výběr a vytváření kontaktů v rámci aplikace[15].

Z frameworku byla použita třída `CNContactPickerViewController`. Její objekt zajišťuje například zobrazení pro výběr kontaktů pomocí zobrazovače `CNContactViewController`, ze kterého může uživatel vybírat jeden nebo více kontaktů. K jednotlivým vlastnostem kontaktu je přístupováno prostřednictvím třídy `CNContact`, která poskytuje různé informace jako je jméno, příjmení, datum narození, fotka či telefonní číslo.

5 Programátorská příručka

Tato kapitola popisuje, jak je aplikace vytvořena z pohledu programátora. V přiloženém adresáři projektu je pro každou třídu stanoven soubor s koncovkou

„swift“. Každý soubor je pojmenován podle třídy v něm obsažené. U souborů, které mají v názvu suffix „+Help“, se jedná o rozšíření daných tříd.

Při tvorbě aplikace nebyl použit Interface Builder, což má za následek větší množství kódu. Na začátku byl odstraněn soubor `Main.storyboard`. Jediné místo, kde bylo využito Storyboards, je při zobrazení obrázku loga aplikace (zobrazeno na obrázku 3) v průběhu jejího spouštění v rámci souboru `Launch Screen.storyboard`.

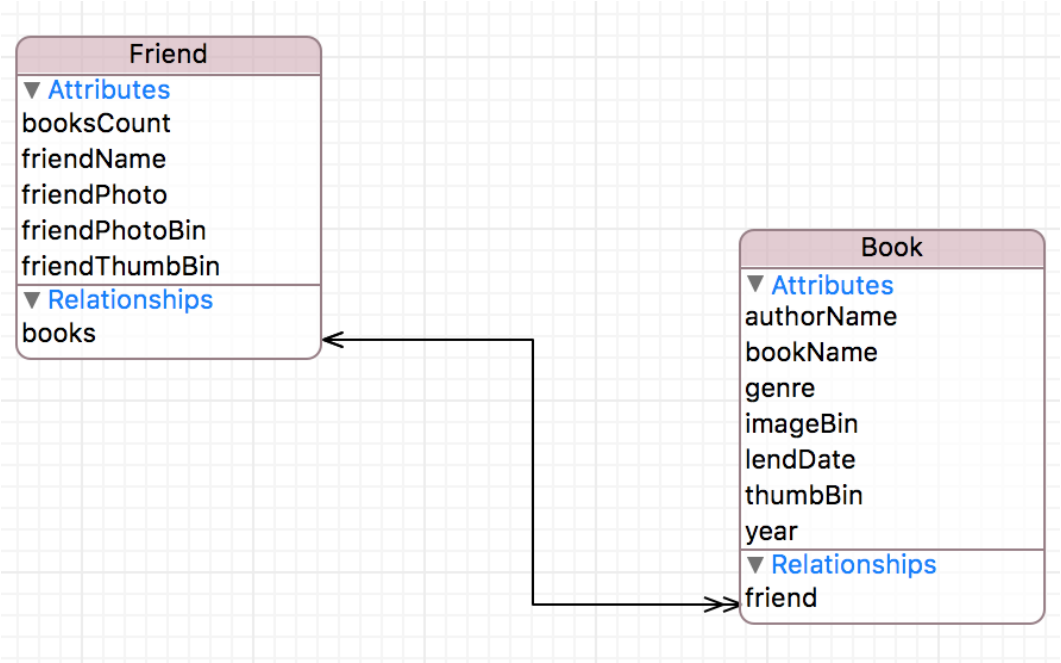
Obrázek 3: Logo aplikace

Následující podkapitoly kladou důraz na popsání sémantiky významných částí kódu a jsou inspirované Model-View-Controller architekturou. Byla tedy snaha rozdělit aplikaci na část datovou, logickou a část zahrnující uživatelské rozhraní.

5.1 Data

Pro správné využití aplikace bylo zapotřebí persistentně uchovávat data. O to se stará technologie Core Data. Samotná data jsou modelována pomocí entit, vzájemných vztahů a atributů. Aplikace pracuje se dvěma hlavními entitami. Jednou z nich je `Book` představující model knihy a její atributy typu `String`, jako jsou například název knihy, autor, či datum vydání nebo atribut typu `Data` určený

pro obrázek knihy a `NSDate` pro datum zapůjčení. Druhou je entita `Friend` reprezentující osobu, která má také několik atributů, mezi kterými je i informace o zapůjčených knihách. Ta je dána vztahem mezi entitami. `Book` může mít vazbu pouze na jednu osobu. V opačném případě `Friend` může mít vazbu hned na několik `Book`. Potom se tedy jedná o vztah 1:N a je zde použita množina typu `NSSet`. Znázornění modelu dat na obrázku 4. Po vytvoření datového modelu, tedy nastavení entit, vztahů a atributů, umožňuje vývojové prostředí Xcode automaticky vygenerovat příslušné třídy. V tomto případě jsou vytvořeny třídy `Book` a `Friend`, které jsou zároveň podtřídami třídy `NSManagedObject`. Ty reprezentují pomyslný model z Model-View-Controller architektury.



Obrázek 4: Data model

5.2 Logika

Při absenci Storyboards je nutno nastavit úvodní scénu pro stav po načtení aplikace. To je provedeno v hlavní třídě `AppDelegate`, kde je upravena metoda *application*, viz zdrojový kód níže.

Vytvoří se okno o rozměrech displeje, nastaví se jeho viditelnost a přidělí se mu počáteční scéna jakožto kořenový „View Controller“. V rámci této aplikace je jím `TabVC`, potomek třídy `UITabBarController` starající se o přepínání mezi dvěma hlavními záložkami aplikace, které jsou definovány ve třídách `BooksVC` a `FriendsVC`.

Nyní se zaměříme na první z nich, třídu `BooksVC`. Ta je potomkem třídy `UICollectionViewController` a slouží především k napojení dat na objekt třídy `UICollectionView`, tedy seznam obsahující jednotlivé buňky, které

```
1 func application(_ application: UIApplication,
     didFinishLaunchingWithOptions launchOptions: [
     UIApplicationLaunchOptionsKey: Any]?) -> Bool {
2        window = UIWindow(frame: UIScreen.main.bounds)
3        window?.makeKeyAndVisible()
4        window?.rootViewController = TabVC()
5        return true
6 }
```

Zdrojový kód 4: Metoda *application*

data zobrazují. K tomu je třeba implementovat několik fundamentálních metod ze třídy `UICollectionViewController` a také jednu metodu z protokolu `UICollectionViewDelegateFlowLayout`. Jednou z nich je *collectionView* s parametrem *sizeForItemAt* vracející pevný rozměr buňky. Další metodou je *collectionView* s parametrem *numberOfItemsInSection* určující celkový počet buňek, které se mají zobrazit. Nakonec musí být implementována *collectionView* s parametrem *cellForItemAt* vracející objekt třídy `BookCell`, která je potomkem třídy `UICollectionViewCell` a definuje podobu buňky. Tato třída musí být zároveň registrována ihned po načtení scény ve stěžejní metodě *viewDidLoad*. V téže metodě je zasílána zpráva *performFetch* objektu třídy `NSFetchedResultsController`, která zajistí načtení dat z persistentního uložiště na základě názvu entity. Může být doplněna podmínka, kterou musí načítaná data splňovat a pole klíčů podle nichž dojde k jejich setřídění. V rámci *viewDidLoad* je definována i horní navigační lišta, která obsahuje několik tlačítek. Činnost tlačítek pro třídění obstarávají dvě metody. Jednak *handleSort*, která pošle zprávu *handleSettings* objektu třídy `SortSettings`, což vytvoří `NSObject` zobrazující kolekci s možnostmi pro výběr třídícího klíče a metoda *handleAscend*, která pouze přepíná atribut *ascend* typu `Bool`. Tlačítko pro přidání buňky obstarává metoda *handleNew*. Vytvoří objekt třídy `BookNewVC` a přepne scénu přidáním obrazovky na vrchol zásobníku, viz zdrojový kód níže.

```
1 func handleNew() {
2       let bookNewVC = BookNewVC()
3       navigationController?.pushViewController(bookNewVC, animated:
           true)
4 }
```

Zdrojový kód 5: Metoda *handleNew*

O výběr jednotlivých buňek se stará metoda *collectionView* s parametrem *didSelectItemAt*, která vytvoří objekt třídy `BookInfoVC`, nastaví mu atribut *bookObj* podle vybrané buňky a přepne scénu.

Třída *BookNewVC* je potomkem `UIViewController`. Má na starosti například zadané vstupy do textových polí nebo výběr obrázku pomocí *picker*,

instance třídy `UIImagePickerController`. Na *imageView* typu `UIImage` je nastaveno rozpoznání doteku `UITapGestureRecognizer`, které zavolá metodu *handleImageTap*, což vyvolá dialogové okno pomocí `UIAlertController` pro výběr následující akce. Možnost pořídit fotografii zajišťuje metoda *handleImageCapture* a výběr z fotogalerie pak *handlePicker*. Samotné nastavení obrázku nakonec probíhá v rámci metody *imagePickerController* s parametrem *didFinishPickingMediaWithInfo*. `BookNewVC` obstarává také výběr žánru pomocí třídy `GenreSettings`. Na štítek *genreText* typu `UILabel` je nastaveno rozpoznání doteku s metodou *handleGenre*, která posílá zprávu *handleSettings* objektu *genreSettings*. Ta, podobně jako tomu je u `SortSettings`, zobrazí kolekci s buňkami typu `MenuCell`. Při výběru buňky se v metodě *collectionView* s parametrem *didSelectItemAt* nastaví žánr a následně *handleDismiss* zařídí zmizení nabídky. Důležitou úlohu má metoda *handleCreate*, která při použití tlačítka *createButt* vytvoří nový objekt podle názvu entity a na základě datového modelu, přiřadí atributům hodnoty na vstupu, uloží do persistentního uložiště, odebere obrazovku z vrcholu zásobníku a přejde na úvodní scénu. V tomto případě se tedy jedná o objekt třídy `Book` a její vlasnosti. Některé ukládané hodnoty mohou být vzdáleně dostupné na příslušné URL adrese ve formátu JSON. Tyto informace je možno stáhnout a automaticky nastavit. K tomu slouží následné metody navázané na tlačítcích v rámci navigační lišty. První z nich je *handleJson*, která přes dialogové okno přijímá vstup v podobě ISBN[16] kódu a dále ho předává metodě *fetchJsonGB* jako parametr typu `String`. Tam je použit v rámci struktury `URLRequest`, která je následně brána jako parametr metody třídy `URLSession`. Ta získá potřebná data a pomocí metody *jsonObject* třídy `JSONSerialization` jsou vrácena ve formátu JSON a ke konkrétním hodnotám je pak přistupováno skrze asociativní pole. Pokud po aplikaci *fetchJsonGB* nepřibydou v textových polích třídy `BookNewVC` žádné hodnoty, zavolá se ještě metoda *fetchJsonOL*, která pracuje podobným způsobem, ale liší se především URL adresou, kterou je naplněna struktura `URLRequest`. Z navigační lišty je dále možné spustit metodu *handleScan*. Ta vytvoří instanci třídy `ScannerVC` a předá její obrazovku na vrchol zásobníku. `ScannerVC` používá kromě UIKit také knihovnu AVFoundation a má na starosti snímání čárového kódu pomocí vestavěného fotoaparátu. V případě úspěšného nalezení je metodě *found* předán řetězec s kódem. Ten je následně argumentem metody *fetchJsonGB* a opakuje se výše zmíněný postup.

Třída `BookInfoVC` oproti `BookNewVC`, která objekty `Book` vytváří, umožňuje především zobrazit hodnoty vlastností daného objektu, provádět jejich úpravy nebo dokonce celý objekt odstranit z persistentního uložiště. Po vytvoření instance třídy `BookInfoVC`, ke kterému dochází při výběru buňky v rámci úvodní `BooksVC`, je instanci nastavena vlastnost *bookObj* obsahující vybraný objekt třídy `Book`. Ta obsahuje potřebné informace k vyplnění prvků, jež `BookInfoVC` používá při zobrazení daných hodnot. Stejně jako třída `BookNewVC` používá `UITapGestureRecognizer` k rozpoznání doteku a obstarává interakci s prvky

[16]ISBN - číselný kód určený pro jednoznačnou identifikaci knižních vydání

jako jsou textová pole, štítek pro výběr žánru či objekt typu `UIImage` pro nastavení obrázku. Navíc umožňuje vytvořit vazbu na objekt třídy `Friend`, což je vztah mezi entitami popsaný v předchozí kapitole. K tomu je zde implementován prvek třídy `UIPickerView`, tedy jakýsi seznam reprezentující všechny setříděné uložené objekty třídy `Friend`, které jsou načteny podobně jako tomu bylo u `BooksVC` pomocí třídy `NSFetchedResultsController`, kde je její instanci *fetchedResultController2* zaslána zpráva *fetchedObjects*. Metoda *pickerView* s parametrem *didSelectRow* vrací index právě vybrané hodnoty, podle kterého lze jednoznačně určit konkrétní objekt. Toho využívá metoda *handleLend* navázaná na prvek `UIButton`, který je určen k nastavení výše zmíněné vazby. Pokud není vlastnost *bookObj.friend* nastavena na hodnotu `nil`, ale na objekt třídy `Friend`, pak je seznam skryt pomocí vlastnosti *isHidden* a místo něj jsou k dispozici prvky `UIImage` a `UILabel`, které jsou naplněny daty reprezentujícími nastavený objekt. U těchto prvků je při dotyku volána metoda *handleFriendInfo*, která vytvoří instanci třídy `FriendInfoVC`, přiřadí vlastnosti *friendObj* příslušnou hodnotu a přepne scénu. Většina prvků používaných k modifikaci hodnot vlastností objektu `Book` mají zpočátku zakázanou interakci ze strany uživatele pomocí atributu *isUserInteractionEnabled*, který je nastaven na proměnnou *permission* typu `Bool`. Navigační lišta obsahuje tlačítko s metodou *handleEdit*. Ta nastaví *permission* a vyvolá opětovné načtení scény pomocí *viewDidLoad*. Díky změně *permission* je zpřístupněna interakce s prvky a na navigační liště jsou nastavena nová tlačítka respektive nové metody umožňující zahodit změny vrácením do stavu před posledním načtením scény díky *handleCancel* nebo možnost uložit provedené změny pomocí *handleSave*, která persistentně uloží *bookObj*, anebo naopak odstranit celý objekt skrze metodu *handleDelete*. V té je použito dialogové okno typu `UIAlertController` s možnostmi potvrzení a odvolání akce přes instance třídy `UIAlertAction`.

Druhou z hlavních záložek definuje třída `FriendsVC`, která dědí ze třídy `UICollectionViewController` stejně jako `BooksVC` a v základu toho mají spoustu společného. Od náležitostí spojených se správným fungováním objektu `UICollectionView` přes funkcionalitu tlačítek v rámci navigační lišty. Nyní se zaměříme na některé odlišnosti. Seznam je tvořen buňkami typu `FriendCell` a načítaná data pomocí objektu *fetchedResultController* jsou typu `Friend`. Metoda *handleNew* vytváří instanci třídy `FriendNewVC` a při výběru buňky obrazovka přechází na instanci třídy `FriendInfoVC` s nastaveným slotem *friendObj*.

Co se třídy `FriendNewVC` týče, tak ta, oproti `BookNewVC`, obstarává pouze štítek, textové pole a obrázek. V navigační liště pak tlačítko s metodou *handleSave* vytvářející nový objekt typu `Friend` obdobně jako *handleCreate* ze třídy `BookNewVC` a tlačítko pro výběr záznamu ze seznamu kontaktů pomocí metody *handleContacts*, která využívá třídu `CNContactPickerViewController` z knihovny ContactsUI. Z té je pak také použita metoda *contactPicker* s parametrem *didSelect*, v rámci které dochází k naplnění jednotlivých prvků daty z vybraného záznamu.

`FriendInfoVC` pracuje se stejnými prvky jako třída `FriendNewVC` a me-

tody jednotlivých tlačítek v navigační liště jsou totožné s těmi v `BookInfoVC`. Je zde navíc prvek `UIButton`, který v případě, kdy aktuálně vybraný objekt *Friend* nemá vazbu na žádný objekt `Book`, má pouze informativní charakter. Pokud je však množina *friendObj.books* neprázdná, tlačítko lze použít a volá metodu *handleBooksView*, která přepne scénu na novou instanci třídy `FriendBookSetVC`.

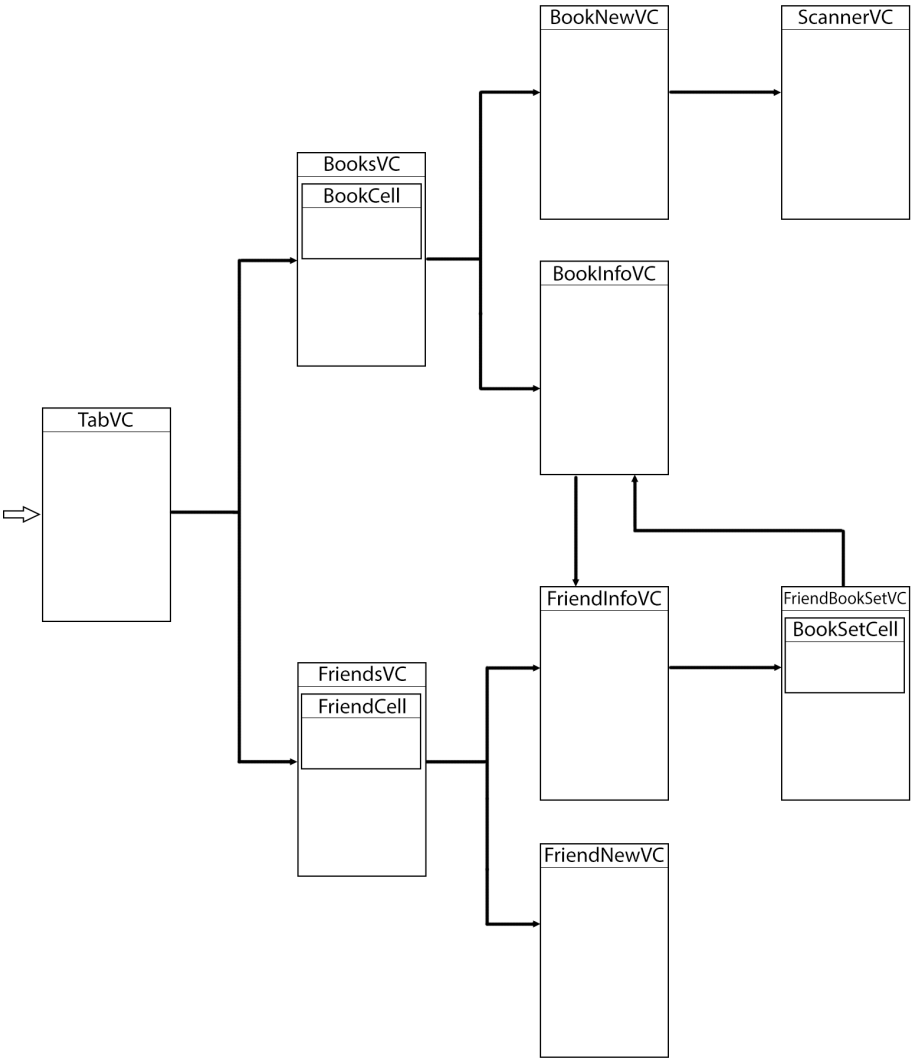
Ta je potomkem `UICollectionViewController` a primárně slouží k zobrazení prvků výše zmíněné množiny pomocí buněk třídy `BookSetCell`. V rámci metody *collectionView* s parametrem *cellForItemAt* jsou jednotlivé buňky naplněny daty. K těm je tentokráte přistoupeno přes vlastnost *friendObj.books*, z které je vytvořeno pole typu `[Book]` a jeho prvky jsou následně setříděny podle atributu *lendDate*. Buňky používají štítek *daysLabel* k zobrazení uplynulého času, který je počítán prostřednictvím metody *getElapsedDate* ze třídy `BooksVC` beroucí jako argument *lendDate* typu `Date` a vracející informaci o čase ve formě textového řetězce. Při výběru buňky se přechází na instanci třídy `BookInfoVC` podobně jako v případě `BooksVC`.

5.3 Uživatelské rozhraní

Samotný vzhled naší aplikace, jež by se dal zahrnout do kategorie view v rámci Model-View-Controller architektury, je tvořen obrazovkami, které jsou reprezentovány třídami zmíněnými v předchozí kapitole. Ty jsou totiž, ať už přímými nebo nepřímými, potomky `UIViewController`, jakožto hlavní zobrazovací komponenty. Jejich hierarchie a vzájemné propojení je znázorněno na obrázku 5.

V počáteční třídě `TabVC` je při vytváření záložek použit konstruktor třídy `UINavigationController`. Ta je zásadní například pro správné fungování záložek, dále přepínání mezi obrazovkami, anebo také zajišťuje přítomnost navigační lišty s popiskem aktuální obrazovky, či tlačítko zpět, které vrátí zobrazení na předchozí scénu.

Každá obrazovka zahrnuje několik prvků sloužících k zobrazení obsahu nebo interakci s uživatelem. Těmi jsou instance třídy `UIView` nebo její potomci. Zmiňme například samotné `UIView` použité pro vykreslení oddělovací čáry, `UILabel` pro popisky, `UITextField` pro textová pole, `UIButton` pro tlačítka nebo `UIImageView` pro vykreslení obrázku. V případě tříd `BooksVC`, `FriendsVC` a `FriendBookSetVC`, kde se jedná o obrazovky, které jsou typu `UICollectionViewController`, je obsah zobrazován pomocí objektů třídy `UICollectionViewCell` tedy jednotlivých buněk a ty v sobě opět implementují zmíněné prvky. V každém případě je vždy zapotřebí prvky nejprve deklarovat a poté umístit na obrazovku. To se děje v rámci metody *setupViews*, kde jsou prvky hierarchicky vkládány pod kořenové `UIView` pomocí metody *addSubview* beroucí vždy jeden konkrétní prvek jako argument. Následně je definováno vzájemné rozložení prvků použitím technologie Auto Layout, které se dá, při použití Interface Builderu, nastavit automaticky. V našem případě musíme každému prvku nastavit vlastnost *translatesAutoresizingMaskIntoConstraints* na hodnotu



Obrázek 5: Schéma propojení „View Controllerů“

`false`. Dále je zapotřebí určit ukotvení z každé strany vůči sousedícím prvkům pomocí `NSLayoutAnchor`, nastavit konstanty pro vzájemné rozestupy a případně určit pevnou šířku a výšku daného prvku. Prvek je pak z každé strany vymezen vůči svému okolí a každé jednotlivé vymezení je třeba ještě aktivovat pomocí vlastnosti *isActive*, která se nastaví na hodnotu `true`.

6 Uživatelská příručka

Tato část představuje návod pro uživatele. Popisuje, jakým způsobem aplikaci nainstalovat na zařízení iPhone. Dále se zabývá možnostmi aplikace a funkcemi prvků, které nabízejí jednotlivé obrazovky.

6.1 Instalace

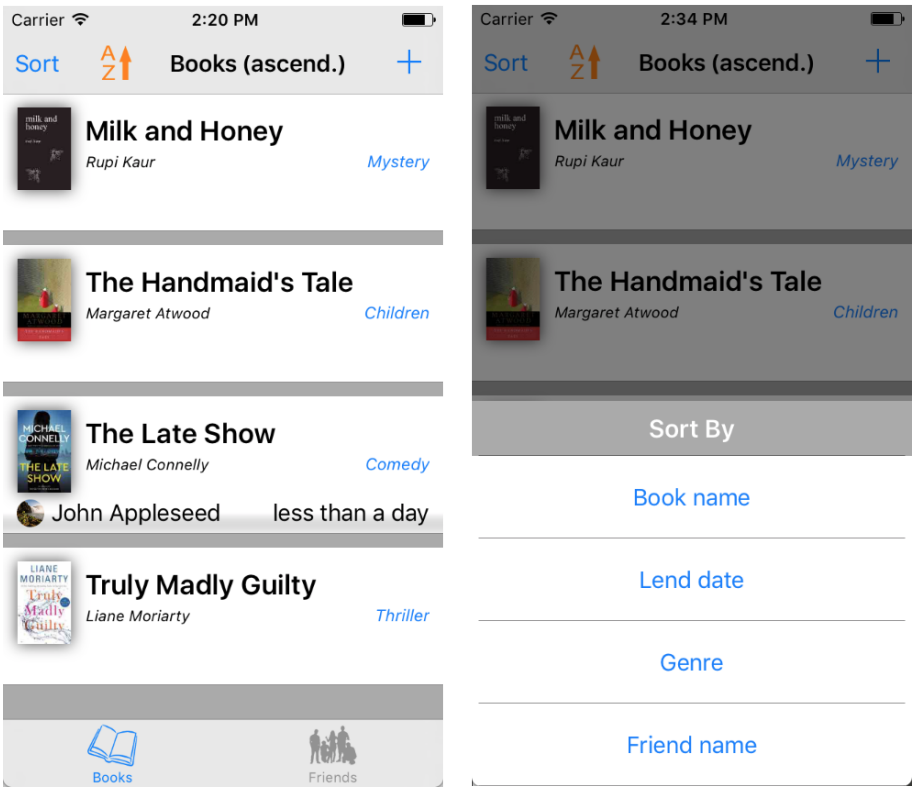
Instalace probíhá tradičním způsobem prostřednictvím obchodu App Store, který je součástí základního softwarového vybavení každého zařízení od firmy Apple. Pro snazší vyhledání aplikace stačí k jejímu názvu *Personal Library* přidat ještě klíčové slovo „upol“. Instalace zabírá necelých 8 MB úložného prostoru a vyžaduje verzi systému 10.0 a vyšší.

6.2 Úvodní obrazovka

Po spuštění aplikace je aktivní úvodní záložka se sekcí „Books“ tedy hlavní obrazovka se seznamem knih. Každý záznam může zobrazovat přední obal knihy, její název, autora a kategorii. V případě, že je kniha zapůjčena, zobrazí se navíc tmavý pruh se jménem osoby, miniatura její fotografie a uplynulá doba od zapůjčení (znázorněno na obrázku 6 vlevo). Výběrem záznamu se zobrazí detail knihy. Na spodní liště je možnost přepnout na druhou záložku reprezentující sekci „Friends“. Horní lišta obsahuje několik tlačítek. Tlačítko „Sort“ zobrazí seznam klíčů a uživatel může vybrat, podle kterého mají být knihy seřazeny (znázorněno na obrázku 6 vpravo). Dále je zde tlačítko „AZ“ pro přepínání abecedního uspořádání a tlačítko „+“ pro přidání nové knihy. Horní lišta navíc obsahuje název klíče a typ uspořádání, podle kterého je seznam aktuálně setřízen. Výchozí hodnotou je název knihy a vzestupné uspořádání.

6.3 Přidání knihy

Uživatel může při vytváření nového záznamu ručně zadat název knihy, jejího autora a datum vydání. Při zadávání žánru se zobrazí seznam pro výběr žánru. Lze také nastavit fotografii přebalu knihy a to výběrem modře orámované oblasti, kde se nachází výchozí obrázek „No Cover“. Tím se zobrazí nabídka pro pořízení nové fotografie, anebo výběr z fotogalerie. Při prvním použití bude aplikace vyžadovat oprávění přístoupit k fotoaparátu či do fotogalerie, což platí napříč celou aplikací. Spodní lišta umožňuje přepínání záložek. Horní lišta obsahuje tlačítko pro návrat na předchozí obrazovku. Dále pak dvě tlačítka umožňující vyplnit údaje o knize automaticky pomocí internetu. Výběrem ikonky lupy se zobrazí pole, do kterého může uživatel zadat kód ISBN označující knihu, na základě kterého budou informace vyhledávány (znázorněno na obrázku 7 vlevo). Aplikace nejdříve zkusí informace zjistit z databáze „Google Books“ (znázorněno na obrázku 7 vpravo). Při nenalezení vyzkouší ještě otevřenou databázi „Open Library“. Po selhání opětovného pokusu se zobrazí informace „Nothing found“ a vyhledávání tím končí. Oproti tomu, při úspěšném nalezení jsou dostupné informace automaticky vyplněny. Uživatel je může dále upravovat a pomocí tlačítka „Create“ uloží změny, vytvoří se nový záznam a zobrazení přejde na hlavní obrazovku se seznamem knih. Poslední tlačítko na horní liště má ikonku fotoaparátu. Tím se zpřístupní snímání čárového kódu knihy pomocí fotoaparátu. Při úspěšné detekci kódu zařízení zavibruje a aplikce se zeptá, zda má tento kód použít pro

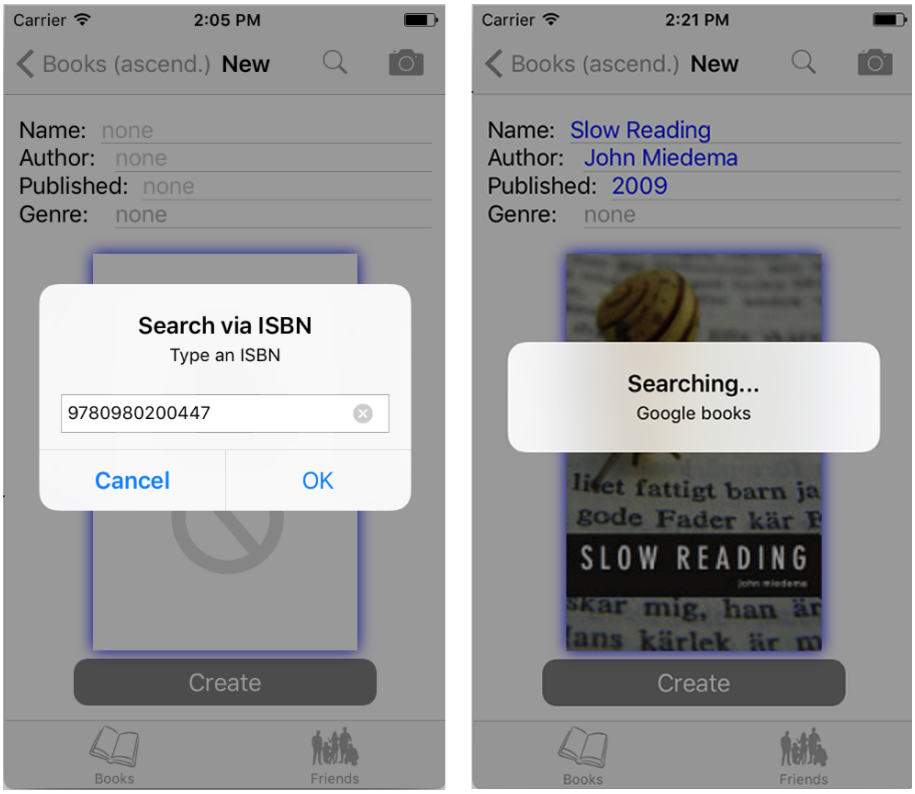


Obrázek 6: Seznam knih a výběr třídícího klíče

následné vyhledávání. Potvrzením se spustí obdobný proces, jako tomu bylo při ručním zadávání ISBN.

6.4 Detail knihy

Tato sekce zobrazuje název knihy, autora, datum vydání, žánr a obal knihy. Zapůjčenou knihu navíc indikuje ztmavené pozadí a fotografie se jménem přítele. Ty slouží zároveň jako tlačítko pro zobrazení detailu přítele. Horní lišta obsahuje tlačítko pro návrat na předchozí obrazovku, název knihy a tlačítko „Edit“, kterým se přepne prostředí do módu úprav. V něm lze provádět úpravy veškerých údajů, které jsou zvýrazněny tmavě modrou barvou. Především však umožňuje evidovat, zda a komu je kniha zapůjčena. K tomu slouží tlačítko, které má v případě zapůjčeného stavu název „Get back“ a simuluje navrácení knihy (znázorněno na obrázku 8 vpravo). V opačném případě je navíc k dispozici rolující seznam obsahující jména přátel a tlačítko nese název „Lend“, které simuluje zapůjčení knihy osobě s vybraným jménem (znázorněno na obrázku 8 vlevo). V rámci „Edit“ módu není k dispozici spodní lišta a horní lišta neobsahuje tlačítko zpět, aby se aplikace nemohla dostat do nekonzistentního stavu. Pro návrat je třeba nejprve mód úprav ukončit. Lze tak docílit použitím tlačítka „Cancel“ a tím zahodit provedené úpravy nebo použitím „Save“ naopak uchovat provedené změny a v poslední řadě je zde možnost knihu nenávratně odstranit pomocí



Obrázek 7: Zadávání kódu a vyhledávání

ikonky koše. V tomto případě se aplikace uživatele nejprve zeptá, zda chce tuto akci opravdu provést. Po smazání záznamu se obrazovka přesune zpět na úvodní seznam knih.

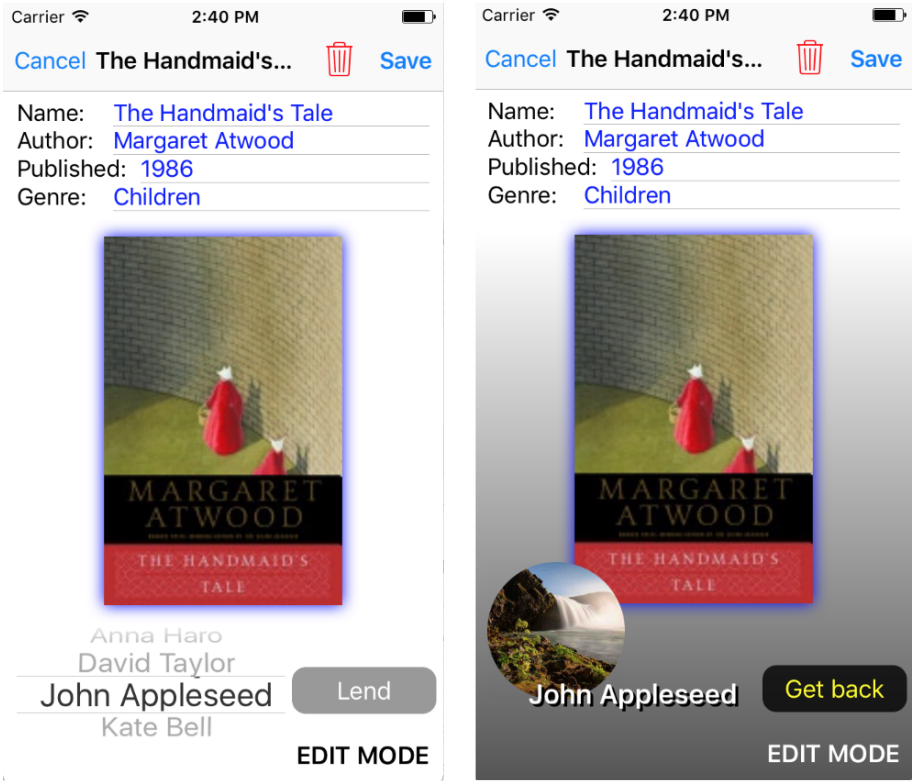
Pokud je při vstupu do této sekce zjištěno, že seznam přátel neobsahuje žádný záznam, zobrazí se uživateli varování. Pro správné fungování simulace zapůjčení knihy je zapotřebí, aby seznam přátel obsahoval alespoň jeden záznam.

6.5 Seznam přátel

S aktivní záložkou „Friends“ se zobrazuje seznam přátel, kde každý záznam obsahuje fotografii přítele, jeho jméno a počet aktuálně vypůjčených knih (znázorněno na obrázku 9 vlevo). Výběrem záznamu se obrazovka přesune na detail přítele. Na horní liště jsou tlačítka „Sort“ pro výběr klíče k setřídění seznamu, „AZ“ pro přepnutí uspořádání a „+“, které vyvolá obrazovku pro přídání nového záznamu. Lišta navíc obsahuje název aktuálně zvoleného klíče a uspořádání.

6.6 Přidání přítele

Tato obrazovka obsahuje pole pro vyplnění jména přítele. Pro jednodušší zadávání jména a příjmení je nastaveno, že každé další slovo začíná automaticky velkým písmenem. Ukončit zadávání je možno klávesou „Enter“. Dále je k dispo-

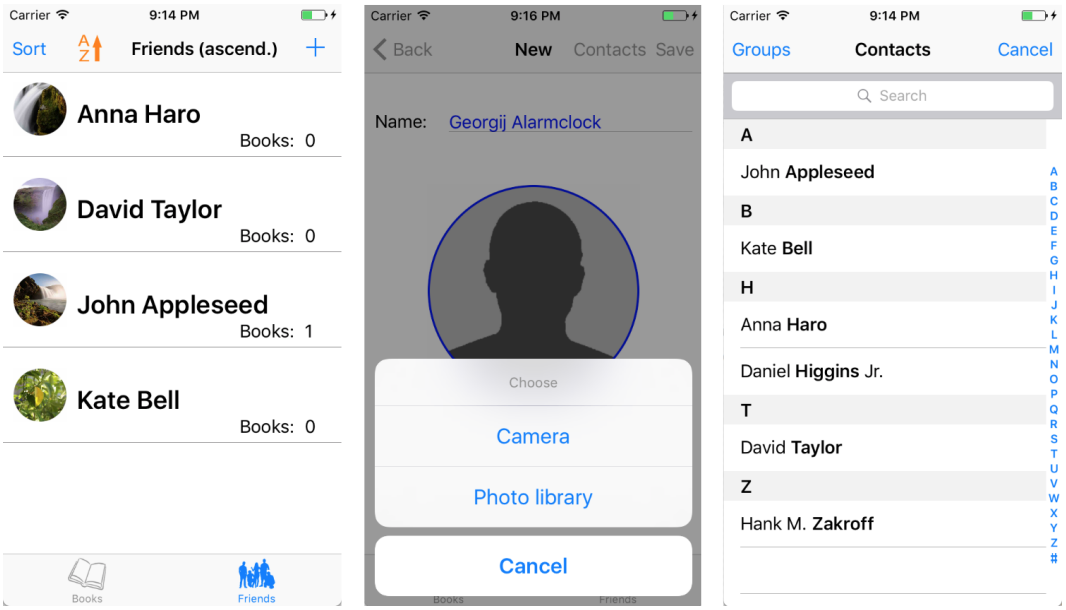


Obrázek 8: Mód úprav a zapůjčení knihy

zici oblast pro výběr fotografie ohraničená tmavě modrým kroužkem. Při výběru si uživatel zvolí, zda použít existující obrázek z fotogalerie nebo pomocí fotoaparátu pořídit fotografii novou (znázorněno na obrázku 9 uprostřed). Spodní lišta obsahuje hlavní záložky. Horní lišta nabízí tlačítko pro návrat na předchozí obrazovku, dále tlačítko „Contacts“, u kterého si aplikace při prvním použití vyžádá přístup ke kontaktům. Umožňuje totiž uživateli výběr kontaktu z adresáře a na jeho základě automaticky vyplní pole se jménem a fotografii (znázorněno na obrázku 9 vpravo). Lišta ještě nabízí tlačítko „Save“. Při jeho použití dojde k uložení změn, vytvoření nového záznamu a přesunutí obrazovky zpět na seznam přátel.

6.7 Detail přítele

V této sekci je zobrazeno jméno a fotografie přítele. Níže se nachází tlačítko „View lent books“, které přepne zobrazení na seznam vypůjčených knih (znázorněno na obrázku 10 vlevo). V případě, že aktuálně žádné knihy vypůjčené nemá, tlačítko změní název na „No books to view“ a má pouze informativní charakter. Spodní lišta je k dispozici. Horní lišta umožňuje návrat na předchozí obrazovku, dále zobrazuje jméno přítele a je zde tlačítko „Edit“ pro přepnutí prostředí do módu úprav. V tomto módu se skryje spodní lišta a tlačítko vypůjčených knih. Prvky, které jsou zvýrazněny tmavě modrou barvou, lze upravovat. Horní lišta nyní



Obrázek 9: Seznam přátel, volba přidání fotografie a výběr z kontaktů

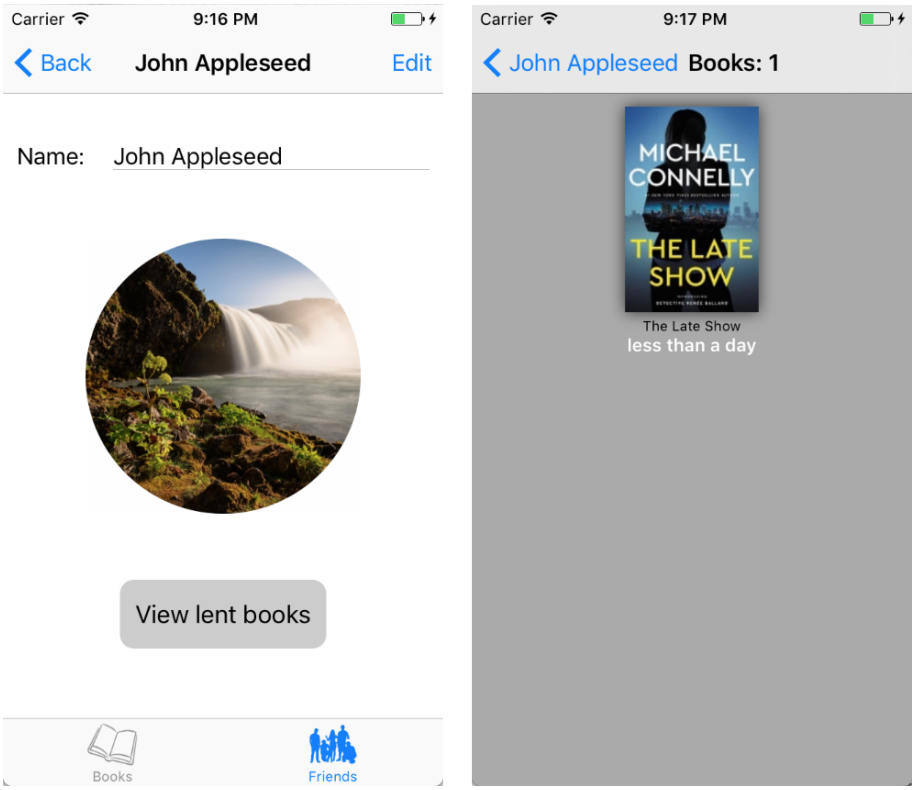
obsahuje možnost „Cancel“ pro opuštění „Edit“ módu bez uložení změn. Oproti tomu tlačítko „Save“ opustí mód a uloží provedené změny. A zbývá červená ikonka koše, která nenávratně smaže přítele a přepne zobrazení zpět na hlavní seznam přátel. Před samotným odstraněním je uživatel dotázán, zda chce tuto akci provést.

6.8 Seznam vypůjčených knih

Obrazovka zahrnuje seznam knih seřazených sestupně podle data zapůjčení. Každý záznam obsahuje obal knihy, její název a uplynulou dobu od jejího zapůjčení. Spodní lišta není k dispozici. Na horní liště je tlačítko pro návrat zpět a číslo udávající celkový počet knih v seznamu (znázorněno na obrázku 10 vpravo). Výběrem záznamu se zobrazení přesune na detail knihy.

7 Možná vylepšení aplikace

Každá aplikace se v průběhu času vyvýjí a snad na každé se dá stále něco vylepšovat. Naše aplikace bude zpočátku využívána zejména místními uživateli, a tak by jí v první fázi nejvíce pomohlo napojení na databázi, která by poskytovala údaje i o česky vydaných knihách. V případě, že by byly dostupné informace ve formátu JSON, byla by navíc následná implementace snadno proveditelná. Následovat by mohlo přidání jazykové mutace, například češtiny. Pro zobrazování seznamu knih je použita kolekce „collection view“, u které by se dalo využít jejich širokých možností pro modifikaci rozložení buněk a nabídnout tak uživateli například přepnutí zobrazení z klasického seznamu jdoucího od shora dolů



Obrázek 10: Detail přítele a seznam vypůjčených knih

na zobrazení buněk do mřížky. Nebo přidat možnost zobrazit buňku na velikost okna a aplikovat horizontální posuv, jakožto imitaci procházení fotek ve fotogalerii. Dalším vylepšením by mohla být automatická změna barvy pozadí nebo pruhu, který indikuje zapůjčenou knihu, v závislosti na uplynulé době od data zapůjčení. Přibýt by mohla také možnost, aby si uživatel u každé knihy nastavil své hodnocení, podle kterého by se daly knihy následně setřídit. Po poměrně snadných úpravách by se dala vytvořit i verze pro zařízení iPad.

Závěr

Na závěr lze konstatovat, že cíle práce byly naplněny. V oblasti vyhledávání knih ve veřejně dostupných databázích byla snaha navázat spolupráci s několika českými institucemi spravující knižní databáze a získat tak přístup k základním informacím o knihách vydaných v česku. Osloveny byly například Databazeknih.cz, Knihy Dobrovský, Československá bibliografická databáze, Národní agentura ISBN v ČR nebo Městská knihovna v Praze. Všichni dotázaní se omluvili, že poskytnout přístup k databázi není možné. Z Československé bibliografické databáze odpověděli, že z licenčních důvodů mohou poskytnout informace maximálně sta titulů, což by nebylo pro účely práce přínosné. Naštěstí otevřené databáze, které aplikace využívá, se neustále doplňují a občas v nich přibudou i české tituly.

Při tvorbě aplikace nebylo využito Interface Builderu ani Storyboards, které vývojové prostředí Xcode nabízí. Jsou to nástroje, které mohou v mnoha případech usnadnit a zrychlit programátorovi práci a to především při tvorbě uživatelského rozhraní. Nevyužitím těchto zdánlivých výhod bylo zjištěno, že k vývoji aplikace je zapotřebí většího množtví kódu, ale programátor není odstíněn od spousty implementačních detailů a snáze pochopí, jak některé technologie fungují. Další výhodou je znovupoužitelnost a lepší přenositelnost kódu. Bylo empiricky zjištěno, že doba kompilace bez přítomnosti Storyboards je zanedbatelně kratší a při práci na monitoru s menší úhlopříčkou displeje je přívětivější upravovat pouze kód, protože u rozsáhlejších projektů je v rámci Storyboards zobrazeno velké množství propojených scén a tím se může práce na projektu stát méně přehlednou.

Použitý programovací jazyk Swift je relativně mladý a neustále se vyvíjí, což má za následek poměrně časté vydávání novějších verzí. V průběhu tvorby naší aplikace došlo k aktulizaci hned dvakrát a pokaždé to přinášelo mnoho změn v již zaběhlé syntaxi. Vývojové prostředí Xcode sice nabízí nástroje pro převod zdrojového kódu na novější verzi, ale bohužel to nefunguje úplně spolehlivě, což nepatrně prodloužilo dobu vývoje aplikace.

Aplikace byla po jejím dokončení nahrána do obchodu App Store, schválena ze strany firmy Apple a následně zveřejněna. Celý proces nahrávání je v textu zachycen v několika krocích a mohl by posloužit jako stručný návod pro začínající vývojáře.

Conclusions

In conclusion, the goals of the thesis were fulfilled. In the field of searching books in publicly accessible databases it was an attempt to establish cooperation with several Czech institutions managing book databases to gain access to basic information on books published in the Czech Republic. For example, Databazeknih.cz, Dobrovský Books, the Czechoslovak Bibliographic Database, the National ISBN Agency in the Czech Republic or the Municipal Library in Prague. All respondents apologized for not being able to access the database. They replied from the Czechoslovak Bibliographic Database that they can provide information for a maximum of hundreds of titles for license reasons which would not be beneficial for thesis purposes. Fortunately, the open databases used by the application are constantly expanding and sometimes Czech titles are added too.

When creating an application neither Interface Builder nor Storyboards was used by the Xcode development environment. These are tools that can, in many cases, make it easier for the programmer to work especially when creating a user interface. By not exploiting these apparent benefits it has been discovered that more code is needed to develop an application but the programmer is not shielded from a lot of implementation details and easier understands how some technologies work. Another advantage is reusability and better code portability. It was empirically found that the time of compilation without Storyboards is negligibly shorter and when working on a monitor with smaller display it is more convenient to modify only the code because larger projects involve a large number of connected scenes within the Storyboards which can make project work less clear.

The used Swift programming language is relatively young and is constantly evolving resulting in fairly frequent publishing of newer versions. During the creation of our application it happened twice and there have always been many changes to the already existing syntax. The Xcode development environment provides tools to convert source code to a newer version but not fully reliably which slightly prolonged the time of developing the application.

After finishing the app it was uploaded to the App Store approved by Apple and subsequently released. The entire uploading process is captured in a few steps within the thesis text and could be use as a brief tutorial for novice developers.

A Obsah přiloženého CD/DVD

doc/

Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/

Kompletní zdrojové texty projektu Personal Library se všemi potřebnými zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelné verze aplikace.

Literatura

[1] *About the iOS Technologies* [online]. 2014-09-17. [cit. 2017-06-25]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

[2] *Core OS Layer* [online]. 2014-09-17. [cit. 2017-06-25]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html

[3] *Core Services Layer* [online]. 2014-09-17. [cit. 2017-06-25]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.html

[4] *Media Layer* [online]. 2014-09-17. [cit. 2017-06-26]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html

[5] *Cocoa Touch Layer* [online]. 2014-09-17. [cit. 2017-06-26]. Dostupné z: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html

[6] *Xcode IDE* [online]. 2017. [cit. 2017-06-27]. Dostupné z: https://developer.apple.com/xcode/ide/

[7] *Interface Builder* [online]. 2017. [cit. 2017-06-27]. Dostupné z: https://developer.apple.com/xcode/interface-builder/

[8] *Xcode* [online]. 2017-05-08. [cit. 2017-06-29]. Dostupné z: https://cs.wikipedia.org/wiki/Xcode

[9] *A Swift Tour* [online]. 2017-06-05. [cit. 2017-07-01]. Dostupné z: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/GuidedTour.html

[10] *Swift (programovací jazyk)* [online]. 2017-04-03. [cit. 2017-07-02]. Dostupné z: https://cs.wikipedia.org/wiki/Swift_(programovac%C3%AD_jazyk)

[11] *Foundation* [online]. 2017. [cit. 2017-07-02]. Dostupné z: https://developer.apple.com/documentation/foundation

[12] *UIKit* [online]. 2017. [cit. 2017-07-03]. Dostupné z: https://developer.apple.com/documentation/uikit

[13] *Core Data* [online]. 2017. [cit. 2017-07-05]. Dostupné z: https://developer.apple.com/documentation/coredata

[14] *AVFoundation* [online]. 2017. [cit. 2017-07-06]. Dostupné z: https://developer.apple.com/documentation/avfoundation

[15] *ContactsUI* [online]. 2017. [cit. 2017-07-06]. Dostupné z: https://developer.apple.com/documentation/contactsui

[16] *App Store (iOS)* [online]. 2017-07-30. [cit. 2017-08-02]. Dostupné z: https://en.wikipedia.org/wiki/App_Store_(iOS)

[17] *Features - Xcode IDE* [online]. 2017. [cit. 2017-08-03]. Dostupné z: https://developer.apple.com/xcode/features/

[18] *Swift Blog* [online]. 2015-12-05. [cit. 2017-08-04]. Dostupné z: https://developer.apple.com/swift/blog/?id=34

[19] *iOS: A visual history* [online]. 2013-09-16. [cit. 2017-08-05]. Dostupné z: https://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad

[20] *iOS SDK* [online]. 2017-07-07. [cit. 2017-08-06]. Dostupné z: https://en.wikipedia.org/wiki/IOS_SDK