Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Výpůjční informační systém malého vědeckého pracoviště

Vedoucí práce: doc. Ing. Vojtěch Merunka, Ph.D.
Autor: Bc. Zdeno Dubnička

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Zdeno Dubnička

Informatika

Název práce

Výpůjční informační systém malého vědeckého pracoviště.

Název anglicky

Borrowing information system of a small science department.

Cíle práce

Navrhnout a naprogramovat výpůjční aplikaci pro katedru vysoké školy, kde budou spravována veškerá literatura i zařízení, které je možné vypůjčit. Aplikace bude umístěna, spravována a obsluhována lokálně na katedře zaměstnancem katedry.

Metodika

Standardy softwarového inženýrství, především UML a zásady objektového návrhu.

Doporučený rozsah práce

60-80 stran

Klíčová slova

výpůjční systém, MacOSX, Objective-C

Doporučené zdroje informací

Guy Hart-Davis, AppleScript
Ondřej Čada, COCOA – Úvod do programování počítačů Apple
Ondřej Čada, Objektové programování
Stephen G. Kochan, Objective-C 2.0
Vojtěch Merunka, Objektové modelování

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

doc. Ing. Vojtěch Merunka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 30. 10. 2013

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 5. 12. 2013

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 28. 03. 2016

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Výpůjční informační systém malého vědeckého pracoviště" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2016 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Vojtěchu Merunkovi, Ph.D. za odborné vedení při zpracování této práce a za poskytnutí cenných rad.

Výpůjční informační systém malého vědeckého pracoviště

Souhrn

Tato práce pojednává o vývoji výpůjčního informačního systému pro katedru na FJFI ČVUT. Nejprve je popsáno pracoviště, na kterém bude systém využíván a hardware, který jej bude obsluhovat. Poté jsou shrnuty všechny požadavky, které bude uživatelka na systém mít. Po jejich shrnutí byl vybrán programovací jazyk vývoje. Vzhledem k tomu, že server je spuštěn na operačním systému Mac OSX od společnosti Apple, byly vybrány jazyky Objective-C a Swift. Dále bylo potřeba zvolit databázové řešení, které se postará o veškerou práci s daty systému. Vzhledem k jejich dnešní popularitě a možnostem rozšíření do budoucna byl zvolen cloudový databázový poskytovatel PARSE. Poté byla vybrána metoda samotného vývoje aplikace, kterou se stala česká metoda BORM, a mohlo být přistoupeno k samotnému vývoji. Byla provedena podrobná analýza problému a vypracován návrh jeho řešení. Ten byl následně rozpracován pro implementaci v konkrétním programovacím jazyce a bylo vytvořeno grafické uživatelské rozhraní, přes které bude aplikace ovládána. Závěr práce je věnován způsobům, jak je možné aplikaci rozšířit v budoucnu.

Klíčová slova: Výpůjční systém, Mac OSX, Objective-C, Swift, Xcode, vývoj software, cloud, databáze, App Store, grafické uživatelské rozhraní

Borrowing information system of a small science department.

Summary

This thesis deals with development of a borrowing system for a department on FJFI ČVUT. At first, particular department where system will be used is described together with the hardware, which will be serving the system. Afterwards there is a summary of all system requirements, that has been placed by the user. After that a particular programming language could have been chosen. Server is running Apple's Mac OSX and because of that, Objective-C and Swift were chosen. Right after that a database solution, which would handle all the data management needed to be chosen. Because of popularity of cloud solutions and their possibilities, as far as the system extensions in the future are concerned, cloud database provider PARSE was chosen. Then a development methodology BORM was picked to be used and the development itself could begin. Thorough analysis of the problem was performed and the design of the solution could be suggested. That was developed for implementation in particular language and the graphic user interface was designed. Through that the entire system will be handled. The end of this thesis is dedicated to possible ways, how the system can be extended with other functionality in the future.

Keywords: Borrowing system, Mac OSX, Objective-C, Swift, Xcode, software development, cloud, database, App Store, graphic user interface

Obsah

Seznam obrázků

1 Úvod

Stolní platforma Mac OSX od společnosti Apple se stává celosvětově stále úspěšnější. Operační systém Windows má stále náskok v celkovém počtu uživatelů, což je dáno především jeho firemním nasazením a také faktem, že jsou na něj uživatelé zvyklí a většinou to bývá první operační systém, se kterým přijdou do styku. Nicméně i přesto lze najít využití, kde je operační systém Mac OSX vhodnějším kandidátem, než právě Windows nebo Linux. To byl důvod, proč byl pro vědecké pracoviště na FJFI ČVUT vybrán jako malý server právě tento systém.

Jeho konfigurací se zabýval autor ve své bakalářské práci. Na tu také navazuje zadání této práce, kdy byl požadován Výpůjční Systém pro stejné pracoviště, který by mohl být obsluhován právě na tomto serveru. Z požadavků na systém totiž vyplynulo, že nebude třeba neobvykle vysoký výpočetní výkon ani nebudou kladeny jiné specifické požadavky na server a na stávajícím hardwaru nebude problém systém provozovat.

Práce je rozdělena do tří hlavních logických celků. První část popisuje obecná specifika vývoje aplikace pro Mac OSX. Jsou zde rozebrány možnosti, které vývojáři dnes mají, pokud se rozhodnou svou aplikaci vydat právě pro tento operační systém, podmínky pro zapojení se do celého ekosystému Apple a také možnosti, pokud by se tomuto zapojení chtěl vývojář vyhnout. Také jsou zde rozebrány dva oficiálně podporované jazyky a vývojové prostředí Xcode, které sám Apple nabízí a doporučuje pro vývoj aplikací pro svá zařízení.

Druhá část obsahuje počátek samotného vývoje Výpůjčního Systému. Nejprve je proveden sběr požadavků na systém a z něj je následně vypracován úvodní návrh aplikace. Po jeho akceptaci je následně zpracován podrobný návrh aplikace včetně jeho datového modelu a způsobu naprogramování. Také je zde popsán výběr databázového řešení spolu s odůvodněním.

Třetí část se již věnuje vývoji aplikace jako takové. Je zde uveden popis a vývoj uživatelského rozhraní aplikace a odůvodněn jeho výsledný vzhled. Na vzorové výpůjčce je zde také rozebráno chování aplikace. Na konci této části je zařazena samostatná kapitola

věnována možnostem, jak rozšířit aplikaci v budoucnu a návrh několika nových funkcionalit, ze kterých by systém mohl těžit.

2 Cíl práce a metodika

2.1 Cíl práce

Záměrem práce bylo navrhnout a naprogramovat řešení výpůjčního informačního systému pro vědecko-výzkumné pracoviště FJFI ČVUT. Cílem práce je plně využít server, který je již na pracovišti instalován a navrhnout pro něj optimálně řešenou aplikaci. Snahou autora bylo vytvořit aplikaci s jednoduchým a přehledným uživatelským rozhraním, kterou bude uživatel schopen ovládat intuitivně a zároveň také aplikaci optimalizovat pro konkrétní řešení databáze, aby bylo možné nakládat s daty co nejefektivněji. Posledním cílem autora bylo umožnit aplikaci správnou volbou komponentů její budoucí rozvoj a možné rozšiřování.

2.2 Metodika

Použité postupy a obecné informace obsažené v této práci vychází z převážné části ze zkušeností autora. Autor pracoval čtyři roky jako člen IT oddělení renomované nadnárodní pojišťovny, kde se podílel na uživatelské a technické podpoře produktů společnosti. Zde nasbíral zkušenosti, jak by mělo ideálně vypadat uživatelské rozhraní, aby byla práce s aplikací přímočará. Také působí dva roky jako vývojář cloudového ERP systému NetSuite, což mu poskytlo náhled na proces vývoje aplikací a programátorské zázemí. Autor se také aktivně věnuje platformě Apple od roku 2008 a o jazyk Swift se zajímá již od uvedení jeho beta verze. Veškeré ostatní informace byly získány studiem literatury, která je v práci beze zbytku citována. Klíčovým zdrojem informací byla oficiální dokumentace k programovacímu jazyku Swift vydaná přímo společností Apple: *The Swift Programming Language, Swift 2 Edition*.

Pro samotný vývoj bylo využito české metodiky vývoje softwaru BORM, která je v této práci také popsána. Nejprve byly sebrány požadavky uživatele na systém. Po jejich podrobné analýze bylo navrženo konečné řešení a vytvořen design samotného Výpůjčního Systému. Tento design byl následně implementován v konkrétním programovacím jazyku.

3 Přehled řešené problematiky

3.1 Obecný vývoj aplikací pro OSX a iOS

Od vydání svého chytrého telefonu iPhone v roce 2007 a jeho následnému otevření vývojářům pro vývoj aplikací třetích stran na začátku roku 2008 se Apple snažil umožnit vývoj aplikací co největšímu počtu vývojářů. Toho se snaží docílit co možná největším zjednodušením celého procesu vývoje aplikací.

To má své nesporné výhody. Apple tím, že poskytl kompletní vývojové prostředí s celým zázemím, umožnil vývojářům přístup ke vždy aktuální verzi jazyka. Také mají přístup k aktuálnímu debuggeru a k simulátorům poslední verze systému, na kterých mohou aplikaci spustit a ladit.

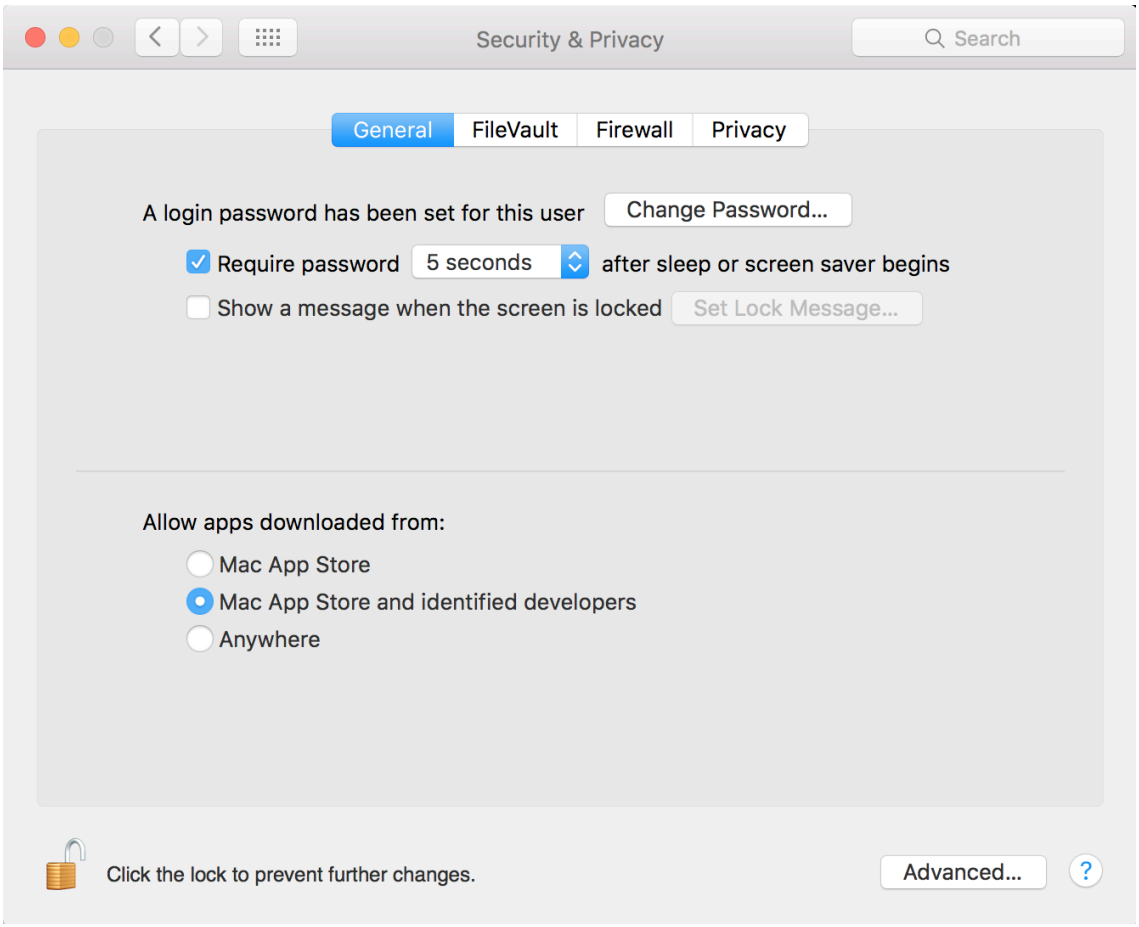
Toto rozhodnutí však vzbudilo také určitou kontroverzi, protože Apple se tak stal účastníkem a také jakýmsi dozorčím celého procesu vývoje a vydání aplikace (určitou výjimkou je platforma Mac – viz níže). Tuto skutečnost mohou vývojáři vnímat jako omezení své práce a určité svázání. V současné době tak prakticky nemá smysl používat jiné prostředky k vývoji než ty, které nabízí samotný Apple.

Pro vývoj pro platformu iOS je nutné použít vývojové prostředí Xcode, které Apple nabízí zdarma ke stažení. Registrovaný vývojář potřebuje Xcode k tomu, aby mohl aplikaci podepsat svým účtem a mohl ji poskytnout k distribuci do App Store.[1] Z toho vyplývá i nutnost použít pouze podporovaný programovací jazyk. Určitou výjimkou jsou webové aplikace, které je možné spouštět přes rozhraní internetového prohlížeče. Ty ale nemají přístup k systémovým funkcím zařízení a jejich funkcionalita je tím do značné míry omezená.

[1] Internetový obchod společnosti Apple, přes který mohou vývojáři distribuovat své aplikace uživatelům. Pro uživatele iOS je toto jediná oficiální možnost, jak do svého zařízení aplikace instalovat. Na Mac OSX je toto jediná povolená možnost ve výchozím nastavení systému. Výhodou distribuce je možnost nabídnout aplikaci všem uživatelům na světě zároveň. V případě placené aplikace si Apple účtuje marži 30% z ceny aplikace.

Aplikace na mobilní platformy společnosti není oficiálně možné stáhnout jinak, než přes oficiální obchod Apple App Store. V něm aplikace procházejí schvalovacím procesem, zda splňují nároky na bezpečnost, rychlost uživatelského rozhraní a co největší bezchybnost (pokud při testování aplikace zhavaruje, je odeslána vývojáři na dopracování). Toto podepsání a distribuce aplikace nelze provést jinak, než pomocí Xcode.

Platforma OSX je pro vývoj mnohem otevřenější. Apple samozřejmě doporučuje využívat jejich prostředků k vývoji, ale vývojáři na ně nejsou odkázáni. Desktopový operační systém společnosti Apple umožňuje tři stupně zabezpečení, které omezuje možnost instalace nových aplikací.



Obr. 1 - Způsob zabezpečení systému při instalaci aplikací.[2]

[2] Screenshot je dílem autora.

1. Mac App Store – tato možnost zabezpečí, že do počítače budou moc být nainstalovány pouze aplikace stažené přes App Store. To znamená, že aplikace prošly schvalovacím procesem společnosti Apple. Toto je standardní chování systému nastavené při jeho prvním spuštění.
2. Mac App Store a registrovaní vývojáři – tato možnost zajistí, že instalované aplikace jsou přímo z App Store, anebo vývojáři, kteří je vydali a podepsali jsou registrováni u společnosti Apple a v případě nežádoucího nebo nezákonného chování aplikace je možné tyto vývojáře vyhledat.
3. Odkudkoliv – tato možnost znamená nejmenší způsob zabezpečení systému počítače. Sám Apple ji nedoporučuje používat. Je možné instalovat aplikace vytvořené v jiných vývojových prostředích i programovacích jazycích. Nevýhodou je, že tyto aplikace mohou obsahovat závadný kód a mohou poškodit operační systém nebo uživatelská data.

Vývoj aplikace pomocí Xcode nabízí výhodu v podobě její možné distribuce pomocí aplikace App Store, což je také největší deviza celého ekosystému Apple z pohledu vývojáře. Zde mohou uživatelé dílčí aplikaci jednoduše najít, hodnotit a případně doporučit a také stahovat její aktualizace. Tím je velmi jednoduché nabídnout aplikaci ke stažení všem uživatelům daného zařízení najednou.

Uživatelská výhoda stažení aplikace přes App Store spočívá také v principu tzv. Sandboxu. Každá aplikace, která je distribuována přes oficiální obchod společnosti Apple musí tento princip dodržovat. V praxi to znamená, že veškerá data aplikace a veškeré soubory, které jsou s aplikací instalovány, jsou uchovány na jednom místě spolu se spustitelnými soubory. Pokud tato aplikace potřebuje přístup k datům vně svého Sandboxu, musí o to požádat operační systém přes poskytnuté kanály a to, ke kterým datům bude mít přístup, je uživatelsky definovatelné. Uživatel tedy má plnou kontrolu nad tím, k jakým datům a informacím bude aplikace moci přistupovat. Vedlejší výhodou je také jednoduchá odinstalace aplikace, kdy ji stačí pouze přesunout do koše a ten následně vysypat a veškerá data budou okamžitě smazána bez toho, aby „zanášela“ systém odpadem v podobě nepotřebných souborů.

Aplikace, o které pojednává tato práce nebude šířena pomocí App Store. Hlavním důvodem je její zakázková povaha. Aplikace je tvořena na míru konkrétnímu pracovišti a je nepravděpodobné, že by z ní uživatelé na jiných univerzitách mohli také těžit.

Zabezpečení instalace nebude také hrát roli, protože výsledná aplikace bude zkompilována přímo na stanici, na které bude v budoucnu běžet, takže ji není nutné podepisovat žádným vývojářským ID a bude jednodušší jí upravit, pokud by to bylo potřeba. Kompilace na konkrétním hardwaru také zajistí vyšší výkon a optimalizaci aplikace.

3.2 Výběr programovacího jazyka (Obj-C vs. Swift)

Pro vývoj aplikací pro stolní počítače společnosti Apple je možné vybírat z řady programovacích jazyků. Tato platforma je mnohem více otevřená vývojářům, než mobilní operační systém iOS (tam je možné vyvíjet aplikace striktně pouze v Xcode a s použitím C, Objective-C a Swift). Podpora a efektivita jednotlivých jazyků se ovšem razantně liší.

3.2.1 Java

Programovací jazyk Java je ze své podstaty spustitelný na prakticky všech platformách. Pro běh aplikací napsaných v jazyce Java je pouze nutné stáhnout Java Runtime[3] společnosti Oracle. Je to také jazyk, který byl přizpůsoben objektovému přístupu k programování a pro jeho nasazení mluví také jeho rozšíření, kdy je dnes s relativně velkým náskokem nejrozšířenějším programovacím jazykem. Jeho uživatelská základna je tedy velmi široká a případné další úpravy aplikace by nebylo problém vývojářům zadat.

Nevýhodou je poněkud rozporuplná optimalizace běhu těchto programů v operačním systému OSX. Ze zkušenosti autora jsou aplikace napsané v jazyce Java náročnější na systémové prostředky počítače. Kvůli přítomnosti další vrstvy pro běh aplikace tyto aplikace způsobují větší vytížení procesoru počítače a tím i jeho vyšší spotřebu. Ta u

[3] Aktuální verze Java Runtime je stažitelná na adrese:
http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

desktopového počítače nehraje tak významnou roli, ale pokud bude aplikace spuštěna na laptopu, jeho výdrž na baterii může být omezená.

3.2.2 Smalltalk

K vývoji pro platformu OSX je možné využít také programovacího jazyka Smalltalk a jeho vývojového prostředí VisualWorks. Tento jazyk je čistě objektový, takže zcela zapadá do filosofie společnosti Apple. Zdrojový kód jazyka Smalltalk je také velmi efektivní, co se svého zápisu týče. To znamená, že mu stačí velmi málo napsaného zdrojového kódu, aby byl schopen vykonat stejné množství práce, jako jiné jazyky. Pro práci s daty používá Lambda-kalkul[4] spolu s jeho obměnami (alfa-konverze a eta-redukce)[5] a je tedy pro vkládání a úpravu dat velmi příjemný a přímočarý.

Další jeho výhodou je jeho případné nasazení u klienta. Vývojové prostředí jazyka Smalltalk, VisualWorks, funguje jako samostatná jednotka (Merunka 56). Můžeme pro jeho běh použít analogii samostatného operačního systému. Veškerý obsah jazyka: všechny jeho knihovny, všechny funkce, všechna uživatelsky vložená data a dokonce i stav, v jakém se aplikace nachází v každém momentě, je obsažený v prostředí samotném. To znamená, že v každém momentě je možné práci přerušit, prostředí vypnout a po jeho opětovném nastartování jej opět najdeme v tom samém stavu jako před vypnutím.

VisualWorks, a tedy i Smalltalk jsou ze své podstaty multiplatformní (všechno potřebné k běhu je vždy součástí samotného prostředí) a je možné je spustit na přibližně 20 různých platformách a systémech.[6] Co je ovšem unikátní, je možnost ukončit práci na jednom operačním systému, prostředí vypnout a přenést na jiný systém. Na něm je možné opět prostředí spustit a ono bude pokračovat ze stejného stavu, v jakém skončilo před vypnutím.

Největší nevýhodou, která mluví proti použití jazyka Smalltalk pro účely této práce, je jeho malé rozšíření a malé povědomí vývojářů o něm. Tato skutečnost komplikuje jeho

[4] MERUNKA, Vojtěch, *Objektové modelování*. Str.17

[5] Tamtéž, str.21

[6] Tamtéž, str.57

nasazení kvůli jeho správě a údržbě v budoucnosti a velmi limituje jeho možné customizace, jelikož je základna vývojářů, která by toho byla schopná, velmi omezená.

3.2.3 Objective-C

Objective-C je jeden z nativních[7] a oficiálně společností Apple podporovaných programovacích jazyků, které lze použít pro vývoj aplikací pro operační systémy iOS a OSX. Byl využit k vývoji samotných operačních systémů a aplikací v nich, je tedy skvěle optimalizovaný pro běh na operačních systémech OSX a iOS.

Vychází také z jazyka Smalltalk, takže si ponechává jeho výhody, co se týče objektového přístupu k programování. Na druhou stranu je také kompletně zpětně kompatibilní s jazykem C, takže pro některé grafické aplikace nebo složitější aplikace, například pokud jde o využití speciálního hardwaru, je možné jej bez problémů využít.

Možnou nevýhodou jeho nasazení jsou určitá specifika při vývoji, která vychází z příbuznosti s jazykem C. Je tedy nutné myslet například na správu paměti a mít zdrojový kód velmi dobře otestovaný právě s ohledem na práci s pamětí, kdy dochází k velkému množství havárií.

3.2.4 Swift

Jazyk Swift je velmi moderní a nový jazyk. To pro něj může znamenat výrazné výhody, ale také určitá rizika a specifika při práci s ním, se kterými je nutné počítat, pokud se uživatel rozhodne pro vývoj v tomto jazyce.

Jako jazyk vytvořený společností Apple je zpětně kompatibilní s jazyky Objective-C a C, díky čemu nepřichází o žádnou funkcionalitu, ani o robustnost, kterou tyto jazyky nabízejí.

---

[7] Výraz nativní se používá pro vlastnost, kterou daný prvek má přirozeně. V tomto případě se společnost Apple podílí na samotném vývoji programovacího jazyka a jsou v něm vytvořeny i operační systémy Mac OSX a mobilní iOS.

Také je díky statutu nativního jazyka pro platformu iOS a OSX zajištěna jeho bezvadná optimalizace pro použití na těchto platformách.

Další nespornou výhodou plynoucí z jeho mládí je skutečnost, že byl vytvořen s ohledem na možnosti dnešního hardwaru a je schopen z nich relativně jednoduše a automaticky těžit (podpora 64-bit technologie, podpora vícejádrových procesorů atd.).

Jeho potenciální nevýhody plynou opět z jeho krátké historie. První je logická skutečnost, že tento jazyk neměl čas být bezchybně odladěn a stále na něm probíhá velmi výrazný a rychlý vývoj. Z toho důvodu se při programování mohou objevit některé chyby a nečekané překážky, které se u starších a osvědčených jazyků již neobjevují. Také se může stát, že jazyk v tomto stadiu postrádá některé funkce, které jsou u jiných jazyků zcela běžně dostupné. Druhou potenciální nevýhodou je jeho nízká rozšířenost. Ta ovšem velmi rychle roste a hlavně díky platformě iOS a popularity vývoje aplikací pro ni se z jazyka Swift stává velmi populární jazyk pro vývoj. Nyní je také dostupný jako open source, takže lze očekávat, že jeho popularita dále poroste.

3.3 Výsledný výběr jazyka

Po přehodnocení všech vlastností, výhod a nevýhod jednotlivých možností ve výběru zbyly pouze dva jazyky, ve kterých je efektivní aplikaci vytvořit. Vzhledem k jejich optimalizaci pro prostředí OSX, kde bude aplikace používána, a dostupnost všech systémových knihoven, které budou k vývoji potřeba, bylo rozhodováno mezi jazyky Objective-C a Swift. Původní záměr autora bylo využít pro práci striktně jazyk Swift, ale některé systémové funkce byly předepsány v jazyce Objective-C a autor se je rozhodl do jazyka Swift nepřevádět. Vzhledem k tomuto rozhodnutí bude aplikace vytvářena v oficiálním vývojovém prostředí společnosti Apple, v Xcode. Tím bude možné jednoduše využívat všechny systémové knihovny dostupné pro oba jazyky. Níže jsou oba programovací jazyky popsány podrobněji se stručným uvedením jejich historie.

3.3.1 Objective-C

Objective-C vznikl jako nadstavba jazyka C na počátku 80. let 20. století. Z jazyka C si ponechal svou syntax, primitivní datové typy a některé funkce. Na to ovšem přidal funkce

umožňující definovat třídy a metody díky čemu umožnil pracovat s objekty. Tyto vlastnosti si vypůjčil od programovacího jazyka SmallTalk-80, který byl, jak jeho název naznačuje, publikován v roce 1980 a je založen na čistě objektovém přístupu k programování. Využívá pouze dva základní stavební prvky: pojmenování objektu a posílání zpráv mezi jednotlivými objekty.[8]

Jazyk Objective-C nebyl zásadně využíván až do roku 1988, kdy jej licencovala společnost NeXT Computers (založena Stevem Jobsem) a vyvinula pro něj potřebné knihovny. Také vyvinula vývojové prostředí pro práci s jazykem, nazývané NEXTSTEP. Následně byla v roce 1992 podpora jazyka Objective-C doplněna do vývojového prostředí Free Software Foundation GNU. Díky tomu je jazyk i jeho vývojové prostředí dostupné pod licencí GNU General Public License a je tak možné jej bezplatně využívat a upravovat pro své potřeby. Systém NEXTSTEP byl následně v roce 1994 standardizován pod jménem OPENSTEP.

V roce 1996 společnost Apple oznámila akvizici společnosti NeXT spolu se všemi jejími produkty. To zahrnovalo i systém NEXTSTEP. Ten se následně stal základem pro novou verzi operačního systému Mac OSX, kde byl pojmenován Cocoa. V následujících letech bylo díky tomuto spojení možné vytvořit kompletní vývojové prostředí pro vývoj aplikací nejprve pro Mac OSX a následně i pro další operační systémy společnosti Apple (iOS, tvOS a watchOS).

Toto prostředí bylo pojmenováno Xcode a obsahuje vše potřebné pro vývoj aplikací pro produkty Apple. Xcode je k dispozici zdarma pro všechny uživatele počítačů Mac a v rámci tohoto prostředí je také možné stáhnout a prohlížet kompletní dokumentaci k programovacímu jazyku i ke knihovnám použitelným jednotlivými platformami. V roce 2007 Apple uvolnil aktualizaci pro jazyk Objective-C na verzi 2.0 a tato verze je aktuální ke dni odevzdání této práce.[9]

---

[8] MERUNKA, Vojtěch, *Objektové modelování*. Str.48

[9] KOCHAN, Stephen G. *Objective-C 2.0*. Str.13

3.3.2 Swift

V létě roku 2014 Apple na své konferenci WWDC 2014 (World Wide Developer Conference) představil nový programovací jazyk, který nazval Swift. Swift názorově vychází z Objective-C a bere si z něj některé prvky, které byly pro vývoj prospěšné. Jinak se ovšem jedná o zcela nový jazyk s vlastní syntaxí a specifickými vlastnostmi, které by měly vývojářům ulehčit práci a zároveň zvýšit bezpečnost a stabilitu kódu. Tyto vlastnosti budou rozebrány dále v textu.

Swift byl dostupný nejprve jako beta verze a první vydané aplikace bylo možné začít nabízet na podzim téhož roku (Apple má v oblibě svazovat finální vydání nových softwarových produktů do balíků, které poté vydává zároveň s novým hardwarem – nová verze iOS kopíruje vydání nových iPhonů apod.).

Rok po vydání první beta verze jazyka Swift vydal Apple v roce 2015 jeho druhou verzi Swift 2.0. Toto je také poslední aktuální verze tohoto programovacího jazyka ke dni dokončení této práce. Tato práce využívá vlastnosti jazyka Swift 2.0 a bude se odkazovat pouze na tuto jeho aktuální verzi.
V prosinci roku 2015 Apple vydal Swift pod licencí Apache 2.0 license with a Runtime Library Exception. Tím se z jazyka efektivně stal Open Source, jelikož jeho zdrojový kód je dostupný na serveru GitHub. Tam je volně ke stažení a je také možné zpětně na server nahrávat vlastní příspěvky ke kódu. Také díky tomu kolem jazyka vzniká silná komunita a podle Apple je to jeden z nejrychleji přijatých nových programovacích jazyků v historii. V seznamu nejpoužívanějších jazyků si Swift mezi lety 2015 a 2016 polepšil o 9 pozic a předběhl i Objective-C, který se propadl ze 3. na 15. místo. Swift se za pouhé dva roky své existence stal 14. nejpoužívanějším programovacím jazykem na světě (viz níže).

Mar 2016	Mar 2015	Change	Programming Language	Ratings	Change
1	2	↑	Java	20.528%	+4.95%
2	1	↓	C	14.600%	-2.04%
3	4	↑	C++	6.721%	+0.09%
4	5	↑	C#	4.271%	-0.65%
5	8	↑	Python	4.257%	+1.64%
6	6		PHP	2.768%	-1.23%
7	9	↑	Visual Basic .NET	2.561%	+0.24%
8	7	↓	JavaScript	2.333%	-1.30%
9	12	↑	Perl	2.251%	+0.92%
10	18	⇈	Ruby	2.238%	+1.21%
11	13	↑	Delphi/Object Pascal	2.005%	+0.85%
12	28	⇈	Assembly language	1.847%	+1.23%
13	10	↓	Visual Basic	1.674%	-0.28%
14	23	⇈	Swift	1.587%	+0.77%
15	3	⇊	Objective-C	1.461%	-5.23%
16	20	⇈	R	1.285%	+0.33%
17	36	⇈	Groovy	1.193%	+0.78%
18	19	↑	MATLAB	1.193%	+0.19%
19	17	↓	PL/SQL	1.193%	+0.16%
20	31	⇈	D	1.139%	+0.64%

Obr. 2 - Aktuální seznam nejpoužívanějších programovacích jazyků.[10]

3.4 Xcode

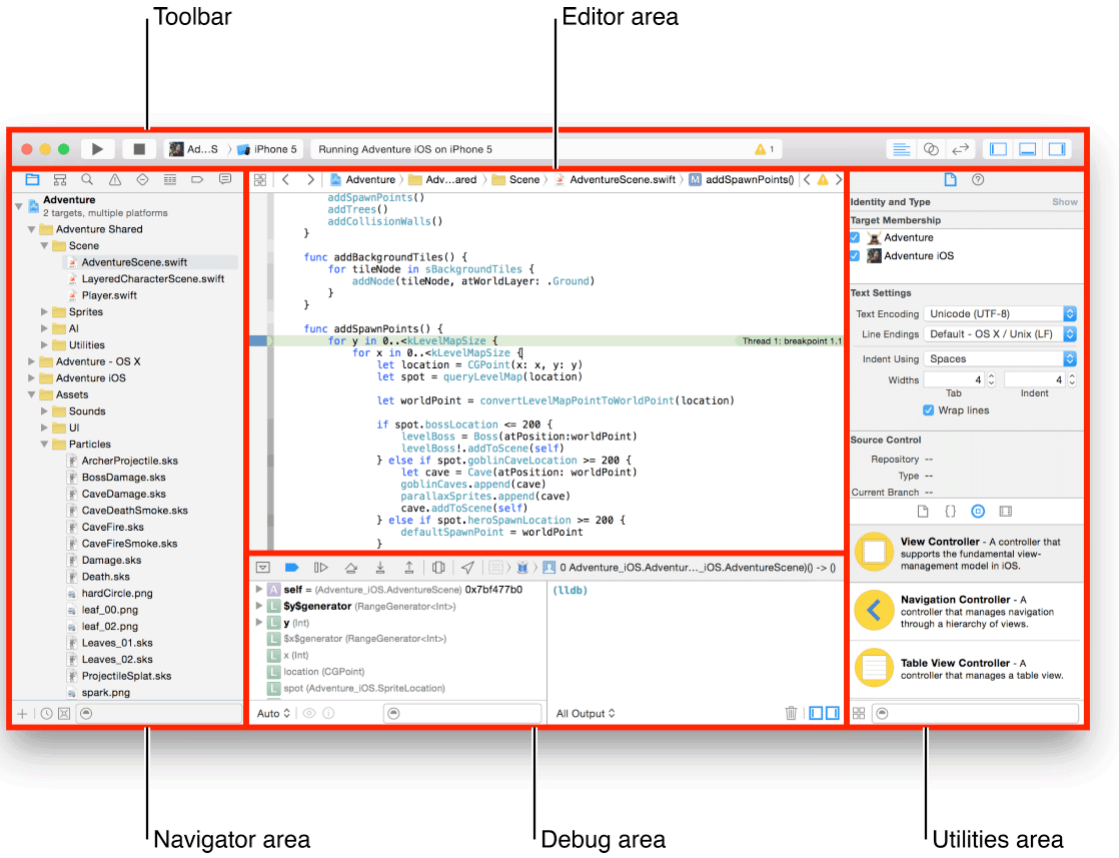
Xcode je kompletní a komplexní vývojové prostředí pro vývoj aplikací pro všechny platformy společnosti Apple. Jeho první verze byla vydána již v roce 2003 a vycházela z předchozí verze vývojového prostředí Project Builder[11]. Každá nová verze vycházela zpravidla na podzim a byla svázána s novou verzí operačního systému pro počítače Mac (později také iOS a další). Aktuální verze prostředí je označována jako Xcode 7 a tato práce se bude odkazovat na tuto aktuální verzi.

[10] Dostupné online: http://www.tiobe.com/tiobe_index?page=index

[11] Vývojové prostředí vyvinuté ještě firmou NeXT pro vývoj jejich operačního systému NeXTSTEP a koupené společností Apple při akvizici NeXT.

Xcode je jedinou oficiálně podporovanou možností, jak efektivně vytvářet nové aplikace pro produkty Apple. To vývojářům znemožňuje výběr mezi jimi preferovanými prostředími. Na druhou stranu nabízí Xcode vše, co je k vývoji potřeba. Obsahuje kompilátor programovacího jazyka, debugger pro detekci a opravu chyb a také simulátor různých operačních systémů a verzí zařízení pro optimalizaci aplikací.

Pro vývoj se Xcode z mnoha důvodů blíží ideálu. Veškerá práce na vývoji aplikaci se omezí na jediné okno. Toto okno je rozděleno na několik oblastí podle jejich obsahu.



Obr. 3 - Rozložení uživatelského rozhraní vývojového prostředí Xcode.[12]

[12] Dostupné online: https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/TheWorkspaceWindow.html#//apple_ref/doc/uid/TP40010215-CH25-SW1

1. Ovládací lišta – jedná se o horní část okna, kde je sdruženo ovládání běhu programu (spustit a zastavit běh aplikace), informační okno a v pravé části prvky pro zobrazení a skrytí ostatních částí okna.
2. Navigační panel – zabírá levou část okna aplikace. Tato část má na starosti většinu navigace v rámci vyvíjeného projektu. Obsahuje kompletní adresářovou strukturu všech souborů v rámci projektu. Sdružuje také všechny externí prvky (například všechny obrázky a grafiky, které budou součástí aplikace). Dále se zde nachází výstupy z testování a debuggeru a je zde k dispozici také seznam varování a chybových hlášení ze zdrojového kódu.
3. Panel nástrojů – nachází se v pravé části okna. Jeho obsah se dynamicky mění podle toho, s jakým prvkem uživatel v té chvíli pracuje v hlavní části okna. V případě, že má uživatel označené například okno aplikace, je zde k dispozici nastavení parametrů tohoto okna. Je zde možné nahlédnout do dokumentace pro každý prvek zdrojového kódu a jsou zde také šablony pro úryvky kódu (například cykly nebo jednoduché funkce).
4. Spodní část okna je vyčleněna Debuggingové[13] oblasti. Je možné zde zobrazit výstup z konzole, případně zobrazit průchod kódem po jednotlivých breakpointech.[14]
5. Hlavní část okna je vždy ve středu prostředí Xcode. V ní se odehrává samotný vývoj aplikace. Zde se upravuje a vytváří samotný kód aplikace a zde se také tvoří její grafické rozhraní. Hlavní část okna je možné rozdělit na dvě části a simultánně mít přístup ke dvěma oblastem vývoje. Klasicky je toto využívá pro zobrazení grafického rozhraní a ve druhé části k úpravě kódu, který bude tuto část rozhraní obsluhovat. Je ovšem možné zde zobrazit libovolné dva soubory z adresářové struktury souborů používaných k vývoji dané aplikace.

Možnost, mít veškerý vývoj soustředěný do jednoho okna, je uživatelsky příjemná. Zvyšuje to ovšem i efektivitu práce při samotném vývoji. Pokud si uživatel dá dostatek

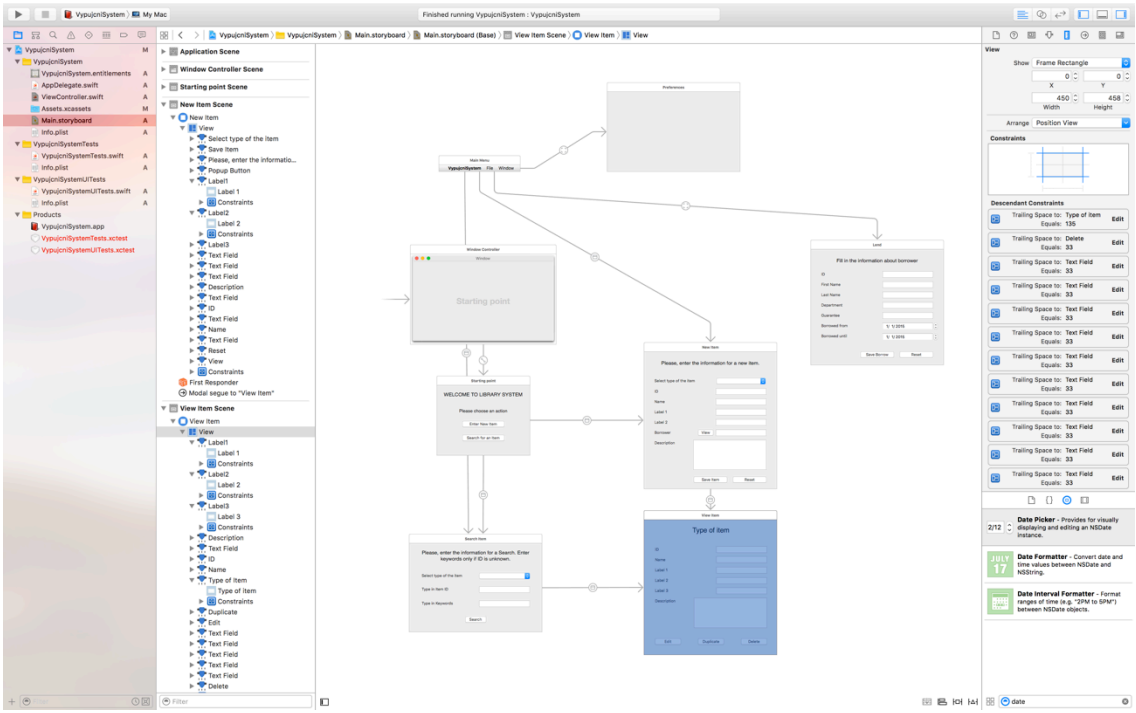
[13] Debugging je výraz používaný pro označení procesu odstraňování chyb objevených při testování aplikace.

[14] Místo označené v kódu, kde se běh programu při testování vždy zastaví a je nutné pokračování v běhu programu potvrdit ručně. Výhodné právě při debuggingu, kdy můžeme kontrolovat průběh aplikací.

času na přípravu prostředí, nemusí jej pak vůbec opustit a veškerá práce je poté mnohem přímočařejší.

Velkou výhodou je také nástroj, který Xcode využívá při tvorbě grafického rozhraní. Celé grafické rozhraní je možné mít situováno do jediného okna. V tomto okně vývojář specifikuje veškerá okna aplikace, všechna pole, která v okně budou a také všechny vazby mezi těmito okny. Panel nástrojů v této chvíli změní svůj obsah tak, aby bylo možné definovat rozměry jednotlivých oken a polí, aby bylo možné upravit výchozí text a výběr pro okna a také, aby bylo možné definovat základní chování všech objektů. Je možné rovněž nadefinovat omezení pro jednotlivé prvky, kdy budou pevně zafixovány k sousedním objektům v okně. Díky tomu je možné kontrolovat chování uživatelského rozhraní během změny velikosti okna, případně tuto změnu zcela zakázat, jako v případě Výpůjčního Systému, kde se autor rozhodl kvůli konzistenci nepovolit změnu velikosti okna aplikace.

Nejen, že je veškeré uživatelské rozhraní v jednom okně, ale je možné ho mít uložené také v jediném souboru. Tento soubor má příponu .storyboard. Jak v něm vypadá rozložení aplikace v průběhu vývoje po definování většiny uživatelského rozhraní, je zobrazeno na následujícím obrázku.



Obr. 4 - Návrh uživatelského rozhraní aplikace v Xcode a souboru .storyboard.[15]

Možnou nevýhodou prostředí Xcode oproti ostatním populárním vývojových prostředím (například Eclipse) je, že Xcode neumožní uživateli rozhraní vývojového prostředí šířeji přizpůsobit. Poloha jednotlivých panelů je pevně dána a neměnná. Apple obsah každého panelu přizpůsobil na míru místu jeho zobrazení a není možné jej měnit. Upravit lze pouze rozměry jednotlivých oblastí prostředí, případně je podle potřeby zobrazovat a skrývat. Pro představu například uživatelské rozhraní zmíněného vývojového prostředí Eclipse (velmi populární pro vývoj v jazyce JavaScript) je možné customizovat[16] prakticky od základu. Toto omezení může vývojáře začínajícího na platformě Apple na začátku omezovat. Je ovšem pravdou, že Xcode je používán již více než 10 let a za tuto dobu Apple s vývojáři spolupracoval i na jeho vzhledu a rozložení. V konečném důsledku je uživatelské rozhraní Xcode pro vývoj pro Apple produkty prakticky dokonalé a umístění jednotlivých prvků zde má svůj důvod.

[15] Screenshot z aplikace je dílem autora.

[16] Výraz customizovat („kastomizovat“) byl převeden z angličtiny a používá se pro vyjádření zakázkové úpravy daného prvku.

4 Vlastní řešení

4.1 Návrh Aplikace

4.1.1 Postup vývoje

Ze zkušenosti autora jasně vyplývá, že je nezbytné mít při vývoji jasnou osnovu a strukturu jednotlivých kroků v celém vývojovém procesu. Od začátku tedy panovala snaha vyvíjet Výpůjční Systém za použití některé z dostupných metodik.

Vzhledem ke snaze o objektový návrh aplikace bylo preferováno, aby vybraná metodika tento způsob vývoje aplikace plně podporovala. Na doporučení vedoucího této práce se autor soustředil na metodu BORM. Tato původně česká metoda vývoje softwaru, vytvořená v roce 1993 a stále průběžně vyvíjená, se orientuje právě na čistě objektově řešené aplikace spolu s objektovými nerelačními databázemi.

Finální uložení dat bude pravděpodobně v nějaké verzi relační databáze. Aplikace samotná ovšem bude s daty pracovat objektově, proto bylo rozhodnuto, že tuto metodu bude možné využít pro účely Výpůjčního Systému. Metoda samotná je ovšem velmi robustní a je schopna pokrýt kompletní životní cyklus vývoje softwaru na podnikové úrovni. Pro potřeby této práce tedy bylo vhodné ji místy zjednodušit. Metoda BORM dělí životní cyklus vývoje softwaru do šesti fází:

1. Strategická analýza – v této fázi dochází k základnímu vymezení problému a jsou rozpoznány základní procesy v rámci systému i v jeho okolí.
2. Úvodní analýza – problém je podrobněji rozpracován a jsou pojmenovány a mapovány konkrétní procesy a objekty, které se na těchto procesech podílejí.
3. Podrobná analýza – podrobně analyzovány jsou jednotlivé objekty a vazby mezi těmito objekty.
4. Úvodní návrh – začíná práce na systému v takové podobě, aby bylo možné jej softwarově implementovat.
5. Podrobný návrh – v tomto momentě je model přeměňován způsobem, který již odpovídá konkrétní implementaci (konkrétní programovací jazyk, databáze apod.).

6. Implementace – závěrečná práce vývoje softwaru, ve které dochází již ke konečnému fyzickému naprogramování aplikace.

Autor práce se rozhodl sloučit první tři body do jednoho a provést celkovou analýzu v rámci jednoho bodu. Toto rozhodnutí je odůvodněno relativně malým rozsahem vývoje aplikace a autor nepředpokládá, že by toto rozhodnutí mohlo způsobit výskyt chyb při vývoji.[17]

4.1.2 Požadavky

Prvním krokem při vývoji byl sběr požadavků a interview s budoucími uživateli aplikace. Výpůjční systém bude jednouživatelská aplikace, to znamená, že v úvodní fázi k ní bude mít přístup pouze jeden uživatel. Ten bude pracovat s daty a spravovat aplikaci jako celek. Tímto uživatelem bude sekretářka katedry. Nebylo tedy nutné řešit správu více uživatelských účtů v rámci aplikace a veškerá data budou přístupná přímo. Aplikace bude instalována pouze na této stanici, není tedy nutné přistupovat k aplikaci nebo k datům externě.

Hlavním požadavkem na aplikaci byla jednoduchost uživatelského rozhraní a rychlost odezvy. Aplikace bude tedy řízena z jednoho okna a zobrazení dalších oken bude řešeno pouze vyskakovacím oknem (pop-up okno). Tato okna budou zobrazena pouze v případě potřeby a po jejich zavření bude opět otevřeno pouze úvodní okno aplikace.

Na pracovní stanici, kde bude systém nainstalován, bude mít přístup pouze jeden uživatel, není tedy potřeba aplikaci zabezpečovat samostatným heslem, bude chráněna přístupovým heslem na stanici samotné. Kvůli tomu bylo také upuštěno od jakéhokoliv šifrování uložených dat. Stanice bude mít šifrovaný celý disk pomocí XTS-AES 128-bitového šifrování[18] poskytnutého utilitou FileVault[19]. Jakékoliv další šifrování dat by tedy pouze

[17] MERUNKA, Vojtěch, *Objektové modelování*. Str.146

[18] Dostupné online: https://support.apple.com/en-us/HT204837

omezilo uživatelský komfort z hlediska rychlosti odezvy, ale větší zabezpečení dat by nepřineslo.

Požadavek na rychlost odezvy znamená klást zvláštní důraz na práci s daty a odladit přístup do databáze. Nebyl vznesen požadavek na žádné konkrétní databázové řešení, které by bylo podmínkou přijetí aplikace. Uživatelka navíc nebude mít k „backendové“ části aplikace vůbec přístup, způsob uložení dat a technologie práce s nimi byla tedy ponechána plně na uvážení autora.

Budoucí uživatelka systému je zkušená v práci s osobním počítačem a nemá v této práci žádná omezení. Nevznesla tedy žádné konkrétní požadavky na ovládání aplikace (nutnost obejít se bez myši, případně zvláštní přizpůsobení pro konkrétní hendikep). Přizpůsobení aplikace pro práci hendikepovaných uživatelů zůstane tedy plně v kompetenci systémové podpory (Accessibility a VoiceOver[20]) a žádná další optimalizace nebude provedena. Toto je v současné době běžný trend u nespecializovaných aplikací. Autor tedy nepředpokládá, že by byl omezen uživatelský komfort nebo možnosti využití samotné aplikace.

4.1.3 Základní návrh jednotlivých komponent aplikace

Z požadavků kladených na aplikaci jasně vyplynuly některé potřeby a nástroje, které bude nutné použít a uspokojit. Zároveň však také vyplynuly možné problémy, které naše aplikace díky jejímu, alespoň z počátku, omezenému rozsahu nebude muset řešit a umožní tím rychlejší zpracování dat a jednodušší správu.

Je evidentní, že aplikace bude potřebovat určitou formu databáze, ve které budou uloženy jednotlivé položky, které bude možné vypůjčit. Také se zdá, že sledovat bude nutné nejen samotné položky, ale i jednotlivé osoby, které si je mohou vypůjčit. Položek k vypůjčení je

---

[19] Proprietární technologie, která po aktivaci zašifruje obsah celého pevného disku počítače. Jedinou možností k přístupu k datům je uživatelské heslo a bezpečnostní klíč, který je vygenerován při spuštění služby. V případě, že uživatel ztratí obě tyto informace, k datům na disku již není možné přistupovat.

[20] VoiceOver je technologie Apple, která umožňuje nevidomým práci s počítačem předčítáním obsahu monitoru.

více druhů, bude tedy potřeba je podle těchto druhů také členit. Unikátní identifikátor jak osob, tak i předmětů, bude jejich evidenční číslo, pod kterým jsou evidovány na katedře. Toto číslo je a priori unikátní, nebude tedy potřeba jej generovat v rámci systému.

Aplikaci bude obsluhovat zkušený uživatel, který ovšem není považován za profesionálního informatika. To klade určité nároky na uživatelské rozhraní. Nemusí zde být vyčerpávající popisky a názorně animovaná navigace aplikací, ale prostředí musí být dostatečně přehledné a přívětivé, aby uživateli co nejvíce ulehčilo práci se systémem.

V počáteční fázi aplikace bude možné se vyhnout dodatečnému zabezpečení, řešení administrace více uživatelských účtů v aplikaci i jakékoliv integraci na jiné systémy katedry a fakulty.

4.1.4 Datový model

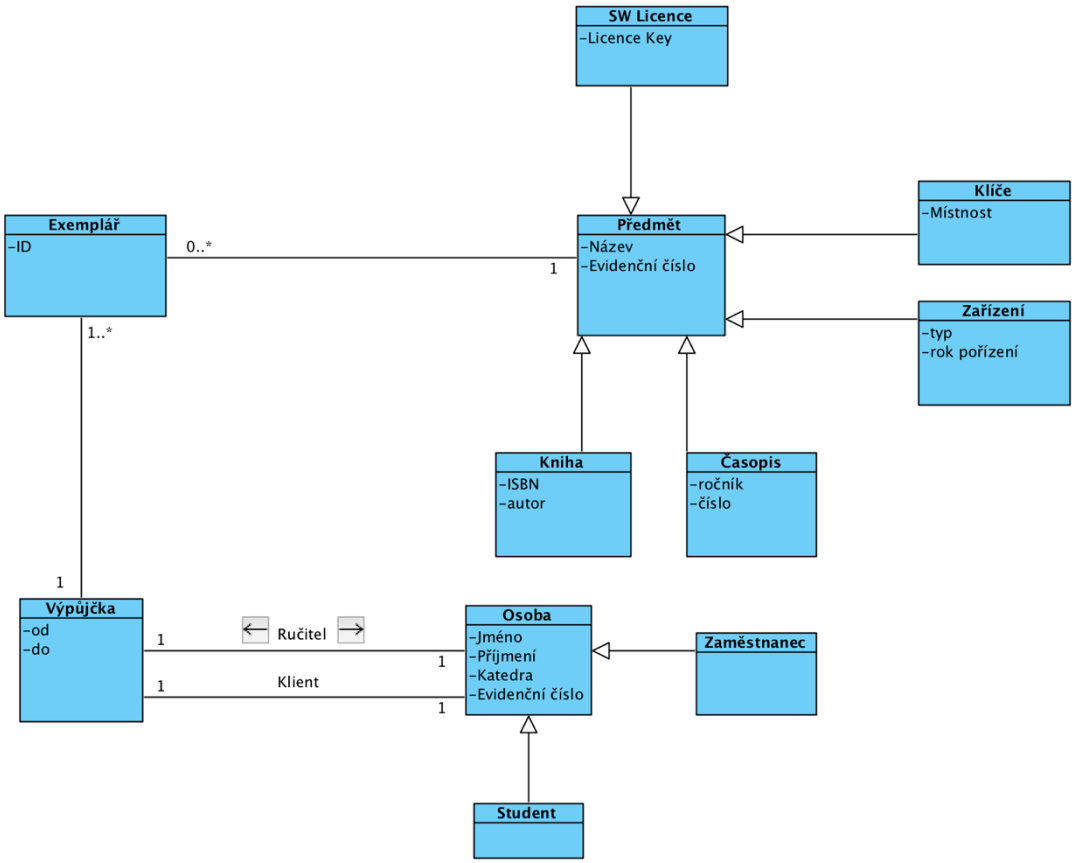
V současné době je veškerá evidence vedená papírovou podobou a záznamy nemají pevnou strukturu. Pro potřeby vývoje aplikace bylo nutné s uživatelkou stanovit podobu zadávaných dat a její konkrétní požadavky na ně.
Vzhledem ke snaze autora stavět aplikaci objektově, byla data stavěna do následujících tříd:

1. Předmět – jedna ze dvou stěžejních tříd pro datový model aplikace. Z této třídy dědí všechny předměty a publikace, které bude databáze využívat. Základní atributy jsou Název a Evidenční číslo. Tyto vlastnosti bude mít každý předmět. K nim se poté pro každou dědící třídu přidají specifické parametry, které mají tyto předměty jedinečné. Děděné třídy jsou:
    a. SW Licence – licence zakoupeného softwaru na katedře. Specifickým atributem bude Licenční klíč.
    b. Klíče – klíče od učeben, charakterizovat je bude číslo místnosti.
    c. Zařízení – elektronické zařízení v majetku katedry, které bude možné si vypůjčit (notebook, projektor apod.).
    d. Časopis – Odborná publikace určena k vypůjčení, charakterizována ročníkem a číslem.

    e. Kniha – podobný typ jako Časopis, ale je definována svým ISBN a jménem autora.
2. Osoba – druhá klíčová třída, která bude použita jako vzor pro dědící třídy pod ní. Definuje všechny osoby na katedře, které budou moci být zadány do systému. Jejími atributy jsou: jméno, příjmení, katedra a evidenční číslo. Třídy, které budou z Osoby dědit jsou:
    a. Student – studenti, kteří budou moci požádat o výpůjčku.
    b. Zaměstnanec – zaměstnanec katedry, který bude moci zažádat o výpůjčku a zároveň může sloužit jako ručitel pro výpůjčku studenta.

    Obě odvozené třídy nepřidávají žádné atributy k nadřazené třídě Osoba. Je však nutné je vytvořit kvůli definování vztahu student-profesor.
3. Exemplář – tato třída představuje kolekci všech předmětů ze systému.
4. Výpůjčka – tato třída definuje konkrétní výpůjčku, kdy propojuje konkrétní exemplář(e) a osoby vypůjčitele a jeho ručitele (pokud je vypůjčitel student).

Všechny položky, které budou evidovány, budou uloženy v databázi položek. Položky budou v databázi uloženy po celou dobu své životnosti na katedře. O osobách v rámci systému toto ovšem neplatí. Aplikace nemá za cíl schraňovat databázi studentů a vyučujících na katedře. Osoba bude uložena pouze jako záznam u konkrétního exempláře, aby bylo možné dohledat údaje o vypůjčiteli, případně ručiteli výpůjčky, kdyby daná položka nebyla vrácena ve stanoveném termínu. Schéma datového modelu aplikace je možné vidět na třídním diagramu níže.



Obr. 5 - Zjednodušený datový model aplikace.[21]

4.1.5 Přístup k uložení dat

V dnešní době je k dispozici obrovské množství možností, jak ukládat uživatelská data pro běh aplikací. Všechny je ovšem možné rozdělit na dva základní typy podle umístění dat: on-site databáze a cloudové databáze. On-site řešení je nainstalováno a uloženo přímo na pracovišti, případně na jeho lokální síti. Databáze je tak „pod jednou střechou“. Cloudové řešení je takové, kdy jsou uživatelská data umístěna externě a pro přístup k datům je nutné využívat sítě internet. Každé toto řešení má své výhody a svá specifika. Níže budou stručně rozepsána jednotlivá řešení podle autorovy zkušenosti.

[21] UML diagram byl vytvořen v balíku aplikací Visual Paradigm a je dílem autora.

4.1.5.1 On-site databáze

Dnes je toto řešení považováno za to „staromódnější“ ze dvou nabízených. Největší výhodou tohoto řešení bývá udávána bezpečnost, možnosti přizpůsobení na míru konkrétní aplikaci a také jeho výkonnost, kdy je rozdíl při velkém počtu dotazů do databáze velmi zřetelný v porovnání s cloudovým řešením.

Bezpečnost je dána skutečností, že databáze nemusí být vůbec přístupná ze sítě internet (a také v drtivé většině případů nebývá). To znamená, že je velmi jednoduché nastavit pravidla přístupu k datům přes použití různých dílčích povolení nebo uživatelských rolí. Je možné sledovat a ukládat do logů jednotlivé přístupy k datům a akce, které uživatel nad těmito daty provádí, a tím dohledat konkrétní případy zneužití nebo poškození dat. Navíc jsou data v databázi zabezpečena heslem a zpravidla šifrována. Díky tomu je prakticky možné vyloučit proniknutí a odcizení citlivých dat jakékoliv společnosti nabouráním do databáze hrubou silou. Jedinou efektivní možnosti je tzv. phishingový útok, kdy útočník využívá zranitelnost samotných uživatelů systému, kteří mu sami poskytnou přístup k datům.

On-site databáze jsou také zpravidla rychlejší v přistupování k datům a práci s nimi. Výhodou je, že k nim uživatel přistupuje většinou po rychlé LAN síti uvnitř společnosti (vynechme nyní přístup k on-site databázi přes internet pomocí zabezpečeného kanálu, zpravidla VPN, kdy jsou databáze umístěny například v jiné pobočce). Vlastník databáze má na provozování vyčleněný vlastní hardware, který je dimenzovaný pro potřeby konkrétního pracoviště a je tedy schopen velmi rychle obsloužit všechny požadavky uživatelů.

Díky tomu, že je vlastník dat zároveň vlastníkem databáze, může si navolit přesnou konfiguraci a konkrétní vlastnosti, které má databáze a její hardware splňovat. Takovéto míry customizace nelze z principu u cloudových databází dosáhnout. Zároveň tyto možnosti přizpůsobení pomáhají zabezpečit jak výkonnost databáze, tak i její bezpečnost.

Z toho ovšem plyne i největší nevýhoda databází umístěných přímo u klientů: jejich vysoká cenová náročnost. K ceně samotného systému databáze je nutné přidat i cenu

hardwaru a cenu za vybavení serverovny, kde budou stanice uloženy. Nemalým nákladem je i mzda zaměstnanců, kteří musí databázi spravovat a v případě nutnosti opravovat. Z tohoto důvodu se klasické on-site databáze omezily pouze na velké podniky, které si dokáží náklady na jejich pořízení a provoz odůvodnit a obhájit.

4.1.5.2 Cloudové databáze

Cloudové databázové systémy jsou dnes považovány za budoucnost ukládání dat ve firmách i u koncových uživatelů. V dnešní době, kdy se klade důraz na možnost přístupu k datům „odkudkoliv“ a kdy jsou mobilní zařízení využívána i pro pracovní činnost jako takovou a ne pouze ke komunikaci, představují cloudové databáze vyhledávaný způsob, jak ukládat a spravovat podniková i osobní uživatelská data. Skutečnost, že je přístup k datům podmíněn pouze internetovým připojením a je tedy možný odkudkoliv ze Země, je také považována za největší devizu cloudových databázových řešení.

Jejich další výhodou jsou relativně nízké náklady. Uživatel vždy platí pouze za ty prostředky, které využívá. To v praxi znamená, že uživatel specifikuje svoje požadavky a očekávání a poskytovatel cloudového databázového řešení mu poskytne dostatek výkonu, aby byly operace prováděné nad daty efektivní. Z toho také vyplývá relativně jednoduchá škálovatelnost těchto databází. V případě, že klient zjistí, že mu dosavadní řešení nedostačuje, nebývá zpravidla problém si za poplatek vyžádat dodatečné zdroje a navýšit výkon téměř okamžitě. Na rozdíl od on-site databází, kdy upgrade hardwaru a jeho implementace může trvat až měsíce, nehledě na nárůst nákladů.

Cloudové ukládání dat se stalo trendem posledních let a díky tomu může těžit z nejmodernějších technologií. Díky tomu jsou tyto databáze velmi dobře připraveny na propojení s aplikací, nabízí kompletní sadu nástrojů pro správu dat a jejich knihovny jsou k dispozici pro mnoho moderních programovacích jazyků.

Donedávna vzbuzovala největší kontroverzi otázka bezpečnosti cloudových řešení. Uživatelé byli skeptičtí z faktu, že jejich data budou uložena v datovém centru mnohdy za hranicemi jejich státu, někdy na jiném kontinentu. Způsob ukládání osobních údajů upravuje zákon č. č. 101/2000 Sb., o ochraně osobních údajů, který v sobě zakotvuje

principy evropské směrnice č. 95/46/ES (ta upravuje nakládání s osobními údaji v rámci EU).[22] V současné době je však bezpečnost cloudových řešení na velmi podobné úrovni, jako u on-site databází. Komunikace je šifrována a chráněna heslem uživatele. Největší slabinou těchto systémů se tedy, podobně jako u lokálních databází, stává sám uživatel.

4.1.5.3 Výběr výsledného řešení

Data uložena v databázi Výpůjčního Systému nejsou důvěrná ani citlivá. To znamená, že otázka bezpečnosti není kritickou pro výběr finálního databázového řešení. Z požadavků vyplývá, že nejdůležitějším kritériem bude rychlost odezvy. Z tohoto důvodu by byl jasným favoritem na výběr řešení nějaký produkt ve verzi on-site. Je však pravdou, že aplikace bude využívána pouze jedním uživatelem a počet požadavků, které je jediný uživatel schopen zaslat do databáze, je tak malý, že se výkonový rozdíl mezi oběma typy databází zcela stírá.

Co však nabízí cloudové řešení oproti lokálnímu, je potenciál aplikaci rozšiřovat a přidávat další funkcionalitu v budoucnu. Kritériem se tedy staly pouze náklady na provozování databáze. Cílem je najít poskytovatele cloudové databáze, kterou by bylo možno využít zdarma (vzhledem k tomu, že by lokální databázi bylo možné provozovat přímo na stanici, kde bude spuštěn Výpůjční systém, náklady na vybudování vlastní databáze jsou nulové). Jako ideální řešení z hledání vyplynul produkt společnosti Parse.

Parse je cloudově řešená databáze, která si klade za cíl poskytnout vývojářům mobilních i desktopových aplikací jednoduchou možnost, jak ukládat svá data a následně k nim odkudkoliv přistupovat. Parse má knihovny a nástroje pro OSX, iOS i Android, takže na všech těchto platformách je možné vyvinout aplikaci, která bude mít k datům přístup.

Pro implementaci Parse je nutné se registrovat na oficiálních stránkách výrobce. Díky této registraci budou poskytnuty přihlašovací údaje, pomocí kterých se následně bude aplikace přihlašovat do databáze. Dalším krokem je pouze stažení a import potřebných knihoven do

[22] Cloud a právo: pozor na umístění vašich dat. In *root.cz*. [online]. KREISL, Marek, PIKAL, Daniel

aplikace Xcode, díky kterým bude možné se přihlásit do databáze a zapisovat, editovat a mazat data v ní. Všechny tyto knihovny jsou již dodány hotové a jediné, co zbývá pro vývojáře, je volat jejich funkce.

4.1.5.4 Naplnění databáze daty

Jak již bylo řečeno v kapitole pojednávající o datovém modelu, veškeré současné záznamy jsou vedeny v papírové podobě ve formě sešitů, do kterých jsou zapsány jednotlivé výpůjčky. Z tohoto důvodu nebyl pro databázi designován ani vyvíjen žádný nástroj, který by umožňoval hromadný import dat. Všechny záznamy o položkách budou muset být přepsány ručně a mohou tedy být vepsány rovnou do databáze pomocí jejích importních nástrojů bez nutnosti tyto nástroje vyvíjet v aplikaci samostatně.

Stejným způsobem bude v budoucnu přistupováno k případné potřebě data masově upravovat, případně nahrávat velké množství dat nových. Data budou z databáze exportována do souborů s příponou .csv, ve kterých bude jednoduché je pomocí příslušných programů (například známý MS Excel, případně jeho OpenOffice ekvivalent, Spreadsheet) editovat. V tomto souboru mohou být data za pomocí vzorců upravena hromadně a relativně jednoduše, a poté budou nahrána přímo do databáze.

Tato činnost zůstane v kompetenci vývojáře, případně zvoleného administrátora, a běžný uživatel k ní nebude mít přístup. Toto rozhodnutí padlo zejména z důvodu zachování konzistence dat v databázi, kdy musí být nejprve provedena kontrola na duplicity, než dojde k samotnému nahrání dat. Aplikace totiž může zůstat aktivní i během úpravy dat a je tedy nutné zajistit, aby některé nové položky nebyly již dříve přidány ručně.

4.1.5.5 Ukončení činnosti PARSE

28.1.2016 vydal jeden ze zakladatelů PARSE prohlášení, ve kterém oznámil, že služba již nebude dále poskytována. Z pozdějších informací vyplynulo, že PARSE byla odkoupena firmou Facebook, která plánuje její nástroje integrovat do svého ekosystému. Společnost

PARSE dala uživatelům na výběr dvě možné cesty a čas na úpravu svého řešení. Datum úplného ukončení činnosti je stanoveno na 28.1.2017.[23]

První variantou je poskytnutí Open Source zdrojového kódu k jejich Parse Serveru. (https://github.com/ParsePlatform/parse-server) To znamená, že by bylo možné vytvořit jejich server lokálně na stanici a aplikace by mohla běžet dále téměř nedotčena (musely by se upravit přihlašovací údaje a adresy). Toto řešení je ovšem nepřijatelné z toho důvodu, že by aplikace přišla o všechny možnosti budoucího rozšíření, kvůli kterým se na počátku volil cloudový systém.
Druhou možností, kterou PARSE poskytl svým uživatelům, je migrační nástroj pro databázi MongoDB. Díky tomu by bylo jednoduše možné přenést veškerá data z PARSE do MongoDB. To je také poskytovatel cloudové databáze. Data však ukládá jiným způsobem, není to relační databáze, ale dokumentová databáze. Data jsou uložena v kolekcích sestávajících z jednotlivých dokumentů. V tomto případě data sestávají ze zápisu ve formátu JSON bez hlubšího systému nebo řádu. Problémem je také fakt, že MongoDB je zpoplatněné řešení. Toto je v současné době nepřijatelné a není tedy možné využít náhradu za PARSE nabízenou oficiálně přímo společností.

Bylo tedy rozhodnuto, že bude nalezen jiný cloudový poskytovatel databázového řešení, který bude sloužit jako náhrada za PARSE. Jako prvotní zdroj bude sloužit komunitní stránka na serveru GitHub, kde vývojáři z celého světa přidávají své návrhy na potenciální náhradu za PARSE.[24]

Práce byla však vypracována a designována pro službu PARSE a s touto službou bude také ve zbytku této práce počítáno. Všude, kde se bude autor odkazovat na databázi, bude tím myšleno cloudové databázové řešení PARSE.

---

[23] Oznámení je dostupné online na stránkách poskytovatele: http://blog.parse.com/announcements/moving-on/

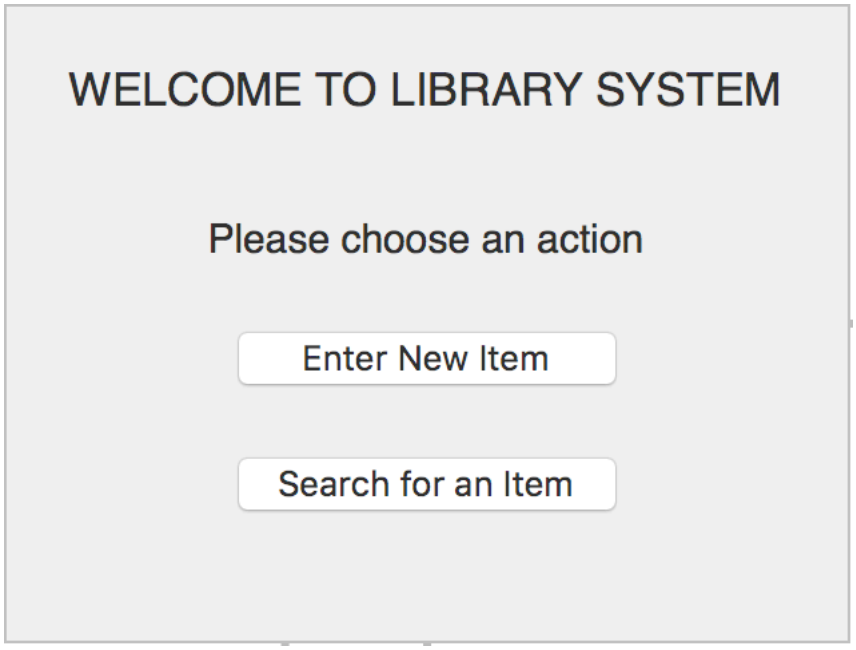
[24] Náhrady jsou průběžně doplňovány online: https://github.com/relatedcode/ParseAlternatives

4.2 Design uživatelského rozhraní

Uživatelské rozhraní aplikace je kontaktní plocha mezi softwarem a uživatelem. Jsou na něj tedy kladeny nemalé nároky. Rychlost uživatelského rozhraní aplikace byla také jedním z hlavních požadavků při jejím zadání. Dalším požadavkem byla jeho jednoduchost a snadná obsluha. Bylo tedy rozhodnuto použít aplikaci soustředěnou do jednoho okna. V případě, že bude nutné zobrazit dodatečné informace, z hlavního okna vyroluje druhé okno. Toto bude v té chvíli jediné aktivní okno celé aplikace. Díky tomu se nestane, že bude uživatel v průběhu používání aplikace zahlcen mnoha okny. Bude mít vždy pouze jedno aktivní okno pro práci.

Vzhledem ke snaze o co největší univerzálnost byl zvolen jako jazyk uživatelského rozhraní angličtina. Takže i v případě, že by se v průběhu životnosti systému stal uživatelem cizinec, nebude nutné upravovat uživatelské rozhraní. Vytvoření více jazykových mutací s možností výběru na stránce Nastavení bylo v současné fázi zamítnuto, ale aplikace je pro tuto možnost připravena. Pokud by se v budoucnu provozovatel Výpůjčního systému rozhodl mít uživatelské rozhraní ve více jazycích, je možné relativně jednoduše doplnit překlady polí a do stránky nastavení přidat možnost volby zobrazovaného jazyka.

4.2.1 Úvodní obrazovka

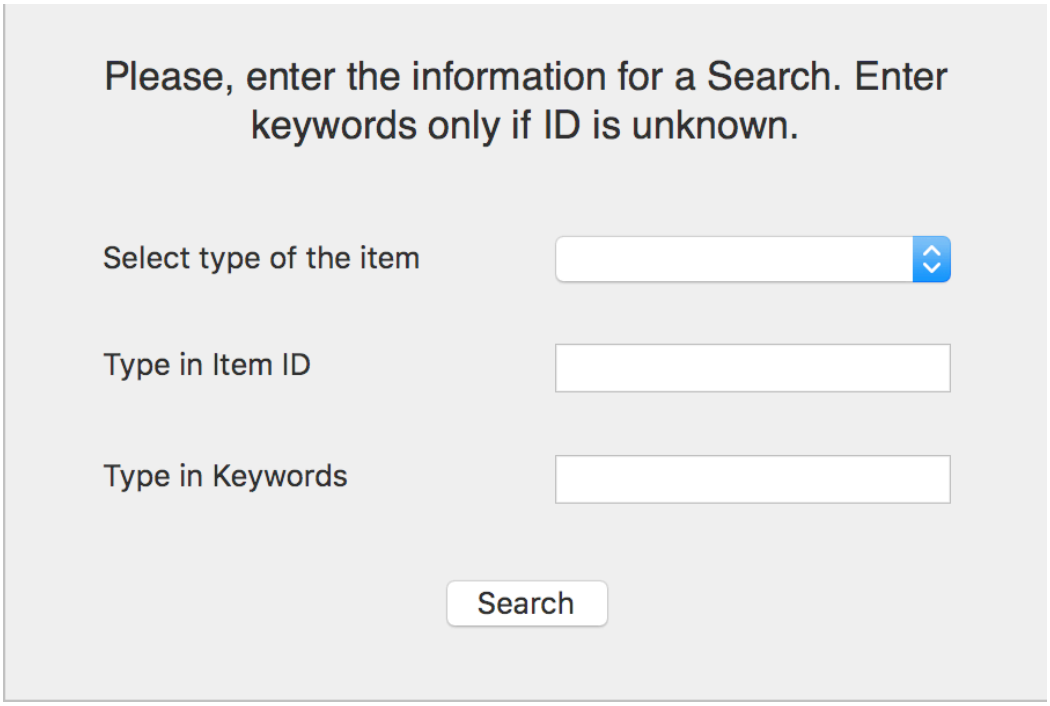


Obr. 6 - Úvodní okno aplikace Výpůjční systém.[25]

Úvodní okno bylo omezeno na dva základní úkony. Vložení nové položky nebo vyhledání stávající. Ve výsledku jsou to jediné dvě možnosti, které lze na začátku provést s každým exemplářem. Po zvolení příslušné možnosti bude zobrazeno druhé okno, které dovolí provést příslušnou operaci.

[25] Screenshot je dílem autora.

4.2.2 Vyhledání exempláře



Obr. 7 - Okno, které se zobrazí při zvolení možnosti vyhledat exemplář.[26]

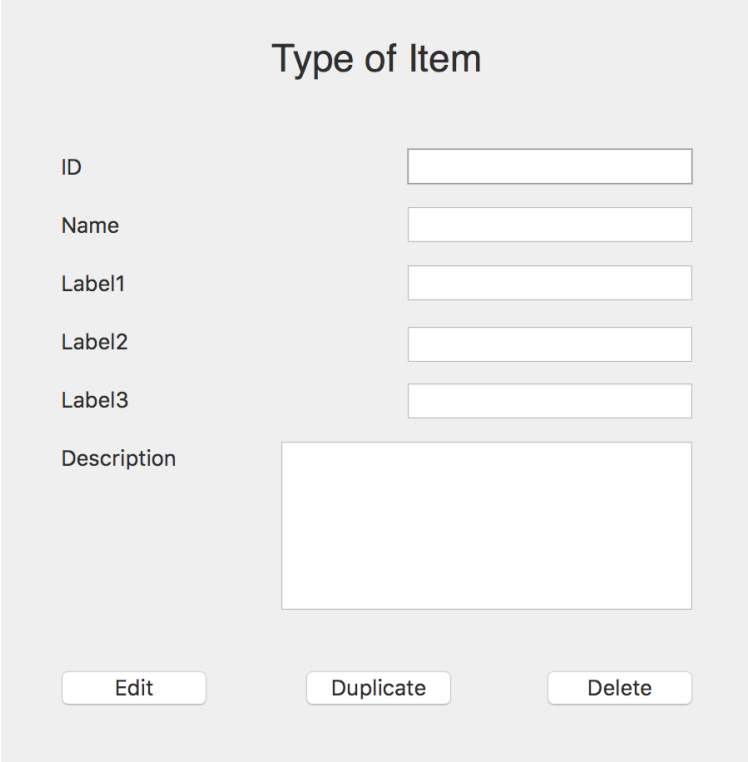
Po zvolení možnosti vyhledat položku bude zobrazeno druhé okno, které se automaticky stane aktivním. Toto okno bude pevně svázáno s původním oknem, takže bude vždy jasné, že patří právě k dané aplikaci a nebude mást uživatele.

Vyhledávat položku lze podle tří atributů: typ položky (klíče, zařízení, kniha apod.), evidenční číslo položky nebo klíčové slovo. Preferovanou možností je hledání podle evidenčního čísla, které vždy vrátí pouze jeden výsledek: konkrétní předmět. Hledání podle typu je doporučeno pro zúžení výsledků vyhledávání a je doporučeno využívat spolu s vyhledáváním podle klíčového slova. Klíčové slovo je řetězec, který vyhledává napříč názvy, ISBN, ročníkem atd.

[26] Screenshot je dílem autora.

Vyhledávání má také svou hierarchii. Prioritu má hledání podle evidenčního čísla. Pokud systém nalezne v databázi shodu, vrátí tento jediný záznam a ignoruje ostatní parametry hledání. Pokud podle evidenčního čísla nebude žádný předmět nalezen, aplikace vrátí chybu s chybným evidenčním číslem. Pokud není evidenční číslo vyplněno, aplikace nejdříve omezí vyhledávání podle typu položky. Pokud nejsou zadána žádná klíčová slova, aplikace zobrazí všechny položky daného typu. Těch může být velké množství, je tedy kladen důraz na co největší omezení vyhledávání. Každá položka má ovšem na sobě natištěné evidenční číslo. Je tedy nepravděpodobné, že by uživatel často vyhledával položky jinak, než právě podle evidenčního čísla.

V případě vyhledávání podle klíčových slov nebo podle typu položky bude zobrazen jednoduchý seznam s výsledky, kterým bude možné listovat a kde budou vyplněny ostatní atributy položky kromě jejího popisu (toto textové pole může obsahovat velký počet znaků a deformovalo by tabulku s výsledky vyhledávání). Z něj poté uživatel vybere položku, kterou hledal a ta se následně zobrazí v samostatném okně se všemi informacemi a popisem (viz níže).



Obr. 8 - Okno, které uživateli ukáže výsledek vyhledávání podle evidenčního čísla.[27]

Ve chvíli, kdy systém najde konkrétní položku podle shodného evidenčního čísla, zobrazí ji ve druhém okně. Toto okno je nyní jediné aktivní okno aplikace. Nadpisem okna bude typ zobrazené položky, aby bylo ihned jasné, co okno zobrazuje. Pod ním budou vypsány jednotlivé atributy dané položky. První bude vždy její název, ten je pro všechny položky vyplněn. Další pole jsou při vývoji designu nadepsána pouze očíslovanými štítky. Toto označení bylo zvoleno proto, že tato pole budou zobrazována a plněna dynamicky podle toho, o jakou položku se bude jednat. To znamená, že v případě, že půjde o klíče, bude zobrazeno pouze jedno dodatečné pole s označením Číslo místnosti. Posledním polem je Popis. Toto pole bylo přidáno kvůli možnosti zaznamenat například stav půjčovaného předmětu. Typicky u zařízení je vhodné sem zapsat případná poškození nebo vady, aby bylo možné jednoduše zjistit, zda bylo zařízení poškozeno. Toto pole je přiřazeno každému exempláři, nicméně nebude možné v něm vyhledávat.

[27] Screenshot je dílem autora.

Pod jednotlivými poli okna budou zobrazena tlačítka, s pomocí kterých bude možné nad jednotlivými exempláři vykonávat požadované akce.

1. Edit – tlačítko, které odemkne jednotlivá pole okna a údaje pro exemplář bude možné upravit podle potřeb. Zobrazená pole jsou v základním režimu zamčená a jsou určena tedy pouze pro čtení. Tento přístup byl zvolen kvůli univerzálnosti rozhraní – kdyby se provozovatel systému rozhodl v budoucnosti systém zpřístupnit více uživatelům, pomocí tohoto přístupu bude mnohem jednodušší zajistit bezpečnost dat, protože tlačítko se zobrazí pouze uživateli, který k tomu bude mít oprávnění. Pokud bude okno v režimu úpravy, ostatní tlačítka zmizí a místo tlačítka Edit se zobrazí tlačítko Save (Uložit). Tím budou úpravy zapsány do databáze a provede se commit[28].
2. Duplicate – toto tlačítko přepne okno do režimu zadávání nové položky (viz níže), s tím rozdílem, že pole kromě evidenčního čísla budou již předvyplněná. Tato možnost bude využívána při zadávání nových položek, pokud se bude jednat o jiný kus stejného zařízení nebo například jiný výtisk stejné knihy. Uživatel pouze zadá unikátní evidenční číslo a nebude muset vyplňovat ostatní údaje, které jsou již v systému zadány.
3. Delete – posledním tlačítkem je tlačíko Delete (Smazat). Po stisknutí tohoto tlačítka bude zobrazeno potvrzující okno, zda si uživatel přeje danou položku opravdu smazat a v případě potvrzení, bude položka odstraněna z databáze. S možností rollback[29] není v současné verzi aplikace počítáno, smazání položky tedy bude nevratné. V případě více uživatelů v budoucnosti bude toto tlačítko opět vázané na uživatelskou roli a bude přístupné pouze oprávněným uživatelům.

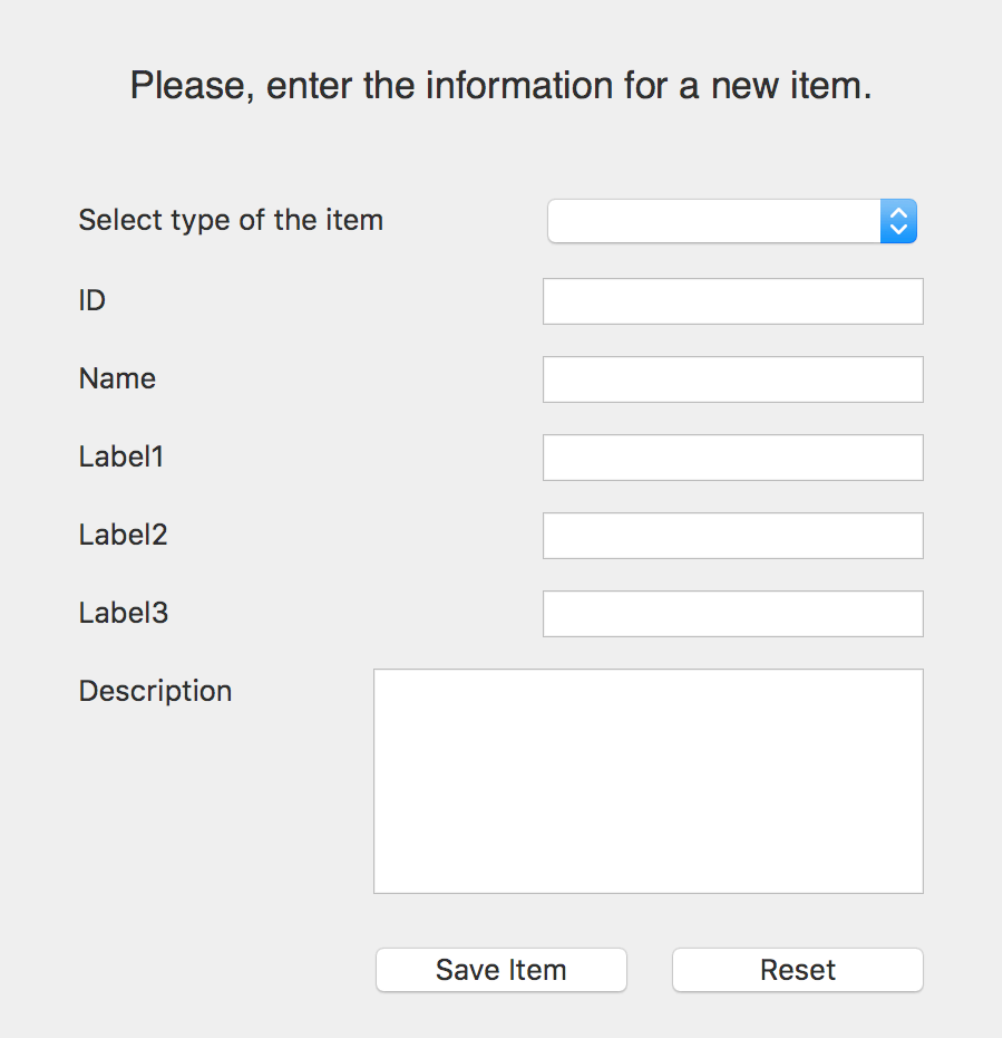
4.2.2.1 Zadání nové položky

Po vybrání možnosti Zadat novou položku v úvodním okně aplikace bude uživatel přesměrován do druhého okna, ve kterém bude mít k dispozici všechna pole, která budou

[28] Výraz commint se používá při práci s daty v rámci databáze. Jedná se o příkaz, kterým uživatel potvrdí změnu dat a tato změna je následně zapsána do databáze.

[29] Výraz rollback se používá při práci s daty v rámci databáze. Označuje akci, kdy uživatel vrací data zpět do původního stavu, ve kterém byla před změnou.

pro zadání položky potřeba vyplnit. Design okna byl volen co možná nejblíže oknu vyhledávání, aby aplikace v každém režimu působila povědomě a ovládání bylo co nejjednodušší. Druhé zobrazené okno bude opět jediné aktivní okno aplikace.



Obr. 9 - Okno, které uživateli dovolí zadat novou položku.[30]

Okno vložení nové položky sestává z řady polí, z nichž některá jsou stálá a neměnná. Těmito poli jsou:

[30] Screenshot je dílem autora.

1. Výběrové pole typu položky – zde uživatel zvolí, o jaký typ položky se jedná (zařízení, kniha, software apod.). Toto pole bude po jeho zvolení použito pro zobrazení dynamických polí (viz níže).
2. Evidenční číslo – pole, do kterého uživatel vepíše unikátní evidenční číslo natištěné na konkrétní položce.
3. Název – název položky podle jmenné konvence zvolené uživatelem.

Ostatní pole zobrazená na stránce jsou dynamická a budou zvolena a zobrazena ve chvíli, kdy uživatel vybere typ položky. Zobrazení okna se v té chvíli obnoví a zobrazí i dynamická pole. Jejich popisky se generují také dynamicky, proto jsou v návrhu designu okna uživatelského rozhraní označena pouze číslovanými štítky.

Posledním polem na stránce je Popis. Toto pole není povinné vyplnit, je tedy možné ho v případě nové položky nechat prázdné a případná poškození nebo jiné záznamy přidávat později.

V okně jsou také přítomna dvě tlačítka. První z nich, Uložit položku (Save Item), zapíše položku pod daným evidenčním číslem do databáze a provede commit. Zároveň toto tlačítko přepne okno do režimu zobrazení položky (stejný režim, jako po vyhledání položky pomocí jejího evidenčního čísla) a uživatel tak může okamžitě zkontrolovat vytvořenou položku. Toto přepnutí zároveň podle evidenčního čísla opět položku vyhledá a nahraje z databáze, takže uživatel může zkontrolovat, že všechna data jsou do databáze vepsána správně. V případě potřeby jsou k dispozici opět tlačítka Edit, Duplicate a Delete. Druhé tlačítko v okně zadávání nové položky je Reset. Tato volba vynuluje všechny záznamy, kromě typu položky, v okně a umožní uživateli začít zadávat informace o položce od začátku.

V případě zadání nové položky bude před provedením akce commit v databázi provedena kontrola na duplicity. Tato kontrola vezme zadané evidenční číslo a pokusí se ho vyhledat v databázi exemplářů. Pokud toto hledání dopadne neúspěšně, aplikace poté provede samotný commit a nová položka bude uložena do databáze. V případě, že systém v databázi najde položku se stejným evidenčním číslem, bude uživateli vrácena chyba

s vysvětlením a položku nebude možné pod daným evidenčním číslem uložit, aby nedošlo k duplikaci dat.

4.2.2.2 Nastavení aplikace

Posledním oknem aplikace je okno nastavení. Toto okno se zobrazuje ze systémové lišty přes navigaci: VypujcniSystem/Soubor/Nastavení[31]. Toto okno bylo přidáno pro budoucí využití, jako sdružené místo pro kontrolu veškerého uživatelského nastavení aplikace. V současné chvíli není žádná možnost nastavení definována, proto okno zůstává prázdné. V budoucnu však již nebude nutné upravovat grafické rozložení aplikace, ale pouze se přidá položka do nastavení. Vložení tohoto okna do aplikace již v tomto stadiu bylo tedy žádoucí a vzhledem k tomu, že s ním uživatel za normálních okolností nemusí přijít vůbec do styku, nebude nijak komplikovat práci s aplikací ani mást uživatele.

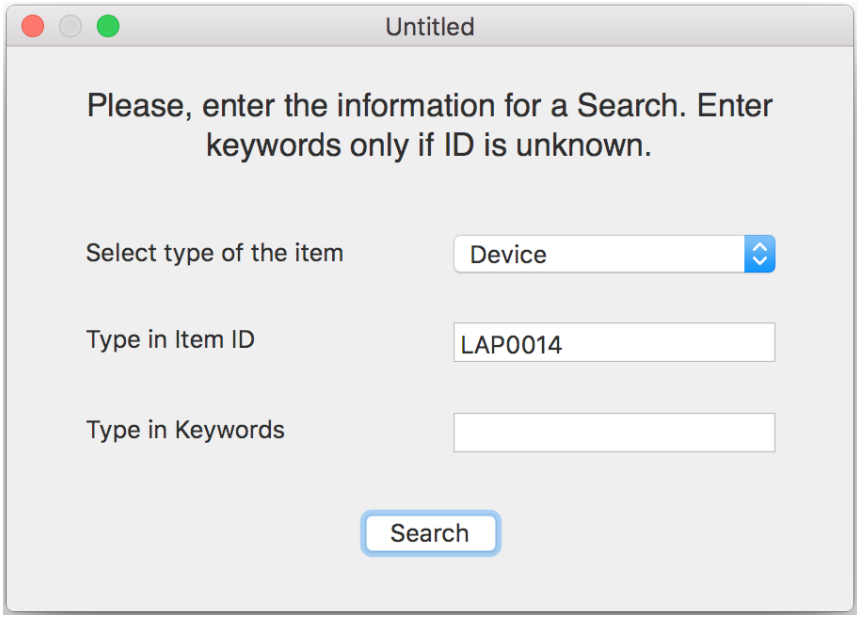
4.2.3 Popis vzorové výpůjčky

Pro názornost bude nyní popsána vzorová výpůjčka. Databáze je naplněná položkami a je tedy možné provést jejich výpůjčku.

Prvním krokem bude nalézt konkrétní položku. V našem případě se bude jednat o notebook Lenovo SL420 s evidenčním číslem LAP0014. Můžeme tedy vyhledávat přímo podle evidenčního čísla, což bude také hlavní use case[32] pro celou aplikaci. Proto byl tento případ vybrán jako vzorový.

[31] Nabídka je zobrazena v nástrojové liště u horní hrany obrazovky v systému Mac OSX.

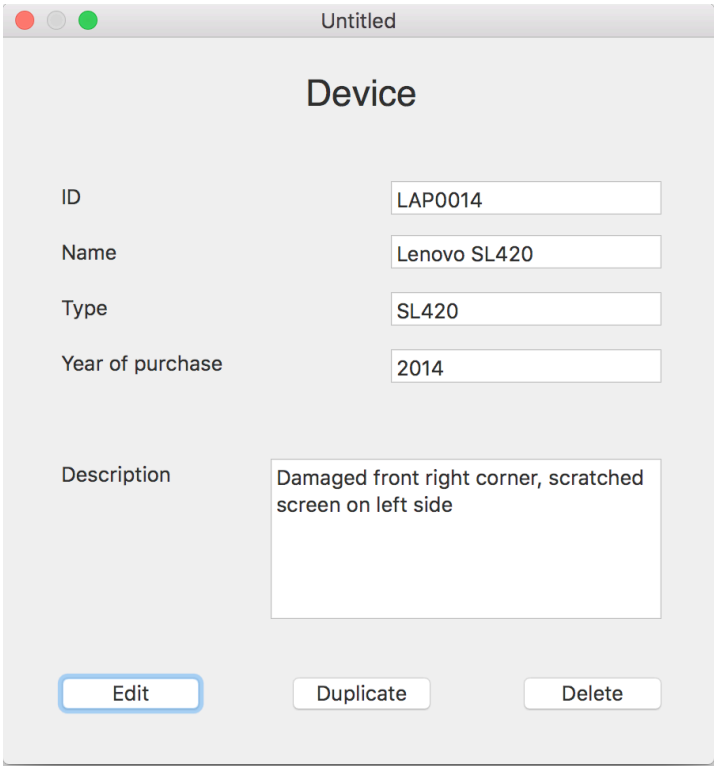
[32] Výraz use case označuje určitý způsob užití dané aplikace.



Obr. 10 - Vyhledávání v aplikaci podle evidenčního čísla.[33]

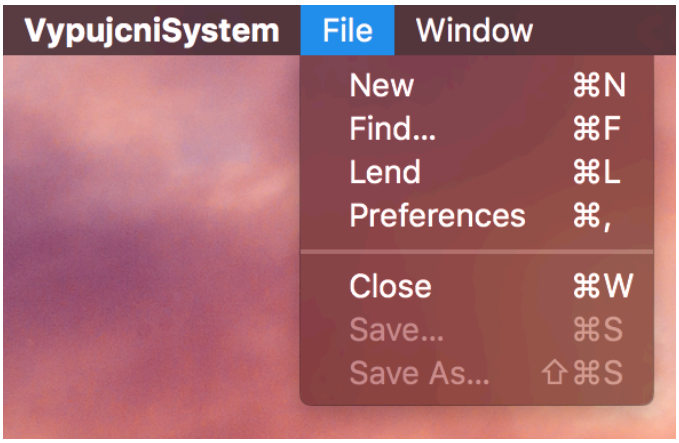
Jelikož je zařízení v databázi již uloženo, po stisknutí tlačítka Search (Vyhledat) se v okně zobrazí konkrétní položka spolu s informacemi k ní.

[33] Screenshot je dílem autora.



Obr. 11 - Výsledek vyhledávání notebooku Lenovo SL420.[34]

V případě, že položka odpovídá hledanému exempláři, uživatel přes systémovou lištu, případně přes klávesovou zkratku Cmd+L, může půjčit žadateli danou položku.

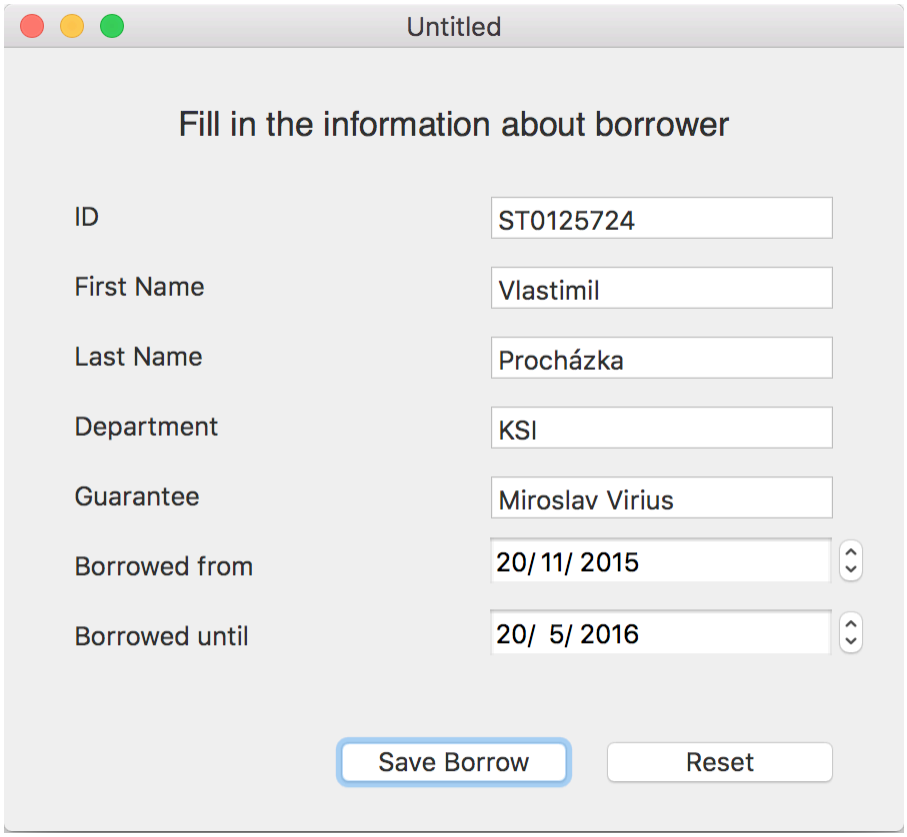


Obr. 12 - Systémová lišta aplikace s možností vypůjčit danou položku.[35]

[34] Screenshot je dílem autora.

[35] Screenshot je dílem autora.

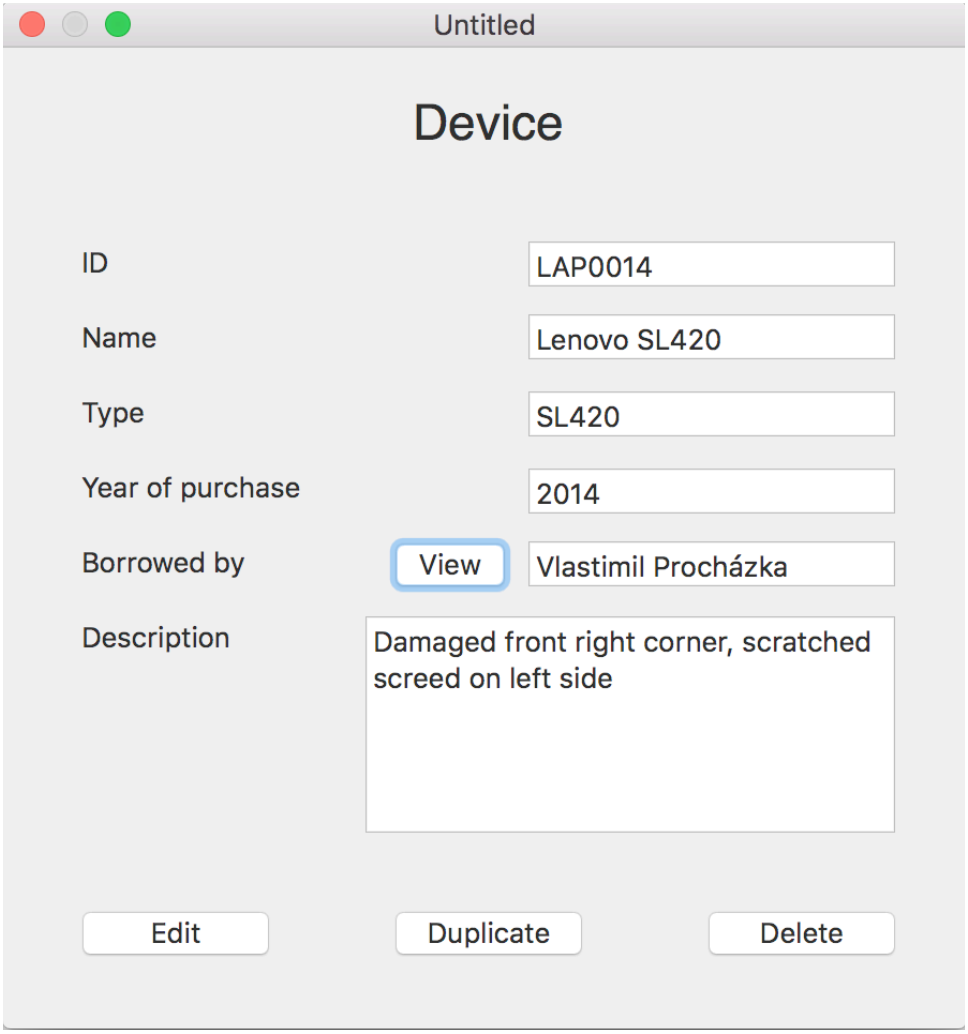
Po vybrání možnosti Lend se zobrazí okno, kde bude možné vyplnit informace o dané výpůjčce. Vyplněno bude evidenční číslo studenta, jeho plné jméno, katedra, na které studuje, ručitel dané výpůjčky (zaměstnanec, který bude odpovídat za vypůjčenou položku) a datum od kdy a do kdy bude položka vypůjčena (jména použitá v tomto názorném příkladu neodpovídají skutečnosti a jsou zmíněna pouze pro názornost).



Obr. 13 - Zobrazení vyplněných údajů o konkrétní výpůjčce.[36]

Po vyplnění údajů o výpůjčce budou tyto informace uloženy v databázi výpůjček, kde budou provázány přes evidenční číslo s konkrétním exemplářem. V okně položky bude zobrazeno navíc pole, které vypíše jméno vypůjčitele. Na této položce také bude deaktivována možnost vypůjčit.

[36] Screenshot je dílem autora.



Obr. 14 - Položka s vyplněnou výpůjčkou a zobrazeným vypůjčitelem.[37]

Vedle jména vypůjčitele se zobrazí tlačítko View (Prohlédnout). Po jeho stisknutí bude uživatel opět převeden na okno konkrétní výpůjčky. Toto okno zůstane aktivní pro editaci v každém momentu výpůjčky. V případě, že vypůjčitel položku vrátí, stačí na tomto okně stisknout tlačítko Reset a znovu Save. Všechny údaje o výpůjčce se smažou a tím se přeruší i vazba na konkrétní položku. Díky tomu bude položka opět k dispozici pro vypůjčení přes systémovou lištu i klávesovou zkratku.

[37] Screenshot je dílem autora.

4.3 Možnosti budoucího rozšíření aplikace

Z požadavků i designu aplikace jasně vyplývá, že nebylo cílem zadavatele ani autora vytvořit aplikaci na podnikové úrovni co se týče vzhledu, výkonnosti ani například zabezpečení. Díky vhodnému výběru nástrojů a technologií má aplikace velmi výrazný potenciál k budoucímu rozšiřování. Toto rozšíření je možné z pohledu funkcí, počtu uživatelů, velikosti databáze a počtu položek, které zvládne aplikace obsloužit i například z pohledu zabezpečení. Níže budou rozepsány možné směry, kterými by se v budoucnu mohl výpůjční systém vydat, pokud by si jeho provozovatel přál rozšířit jeho možnosti. Jako určitá prerekvizita těchto rozšíření by bylo velmi vhodné integrovat aplikaci na univerzitní Active Directory server.

4.3.1 Integrace na Active Directory

Z pohledu autora největším a také evidentním přínosem by byla integrace aplikace s databází Active Directory na příslušné katedře. Jak již bylo řečeno, Systém bude spouštěn na Mac OSX Server, který autor implementoval a konfiguroval na katedře v minulosti. Serverová verze operačního systému v sobě obsahuje službu Open Directory, která je chápána jako variace na známý Active Directory. Tato služba může být relativně jednoduše integrována do služby Active Directory, kde jsou uloženy informace o všech studentech a zaměstnancích dané univerzity. Pokud by byla tato integrace schválena a provedena, systém by tím získal všechna data potřebná pro rozšíření funkcí i zabezpečení.

Tato integrace v sobě nese potenciální riziko z pohledu zabezpečení. V Active Directory jsou všechny informace, to znamená i citlivá data o studentech a zaměstnancích včetně jejich adres bydliště, rodných čísel a soukromých telefonních čísel a e-mailů. Toto riziko by však bylo možné eliminovat za použití speciální role na serveru, která by omezovala data, ke kterým by měl Mac OSX Server přístup a díky tomu by bylo možné vyhnout se citlivým datům uživatelů.

Po integraci serveru na databázi Active Directory již pro Výpůjční Systém nebude problém přistupovat pomocí systémových knihoven k těmto informacím uloženým na serveru. A

jelikož bude bezpečnost řešena přímo na straně Active Directory, nebude nutné implementovat žádná složitá opatření přímo do aplikace.[38]

4.3.2 E-mailové notifikace

Jedním ze směrů, kterým by bylo možné aplikaci rozšířit, je implementace e-mailových notifikací uživateli aplikace i vypůjčitelům ohledně jednotlivých výpůjček. Bylo by nutné pouze obdržet e-mail všech zúčastněných stran. Ten by byl součástí dat získaných integrací na Active Directory (univerzitní e-mail není považován za citlivou informaci, je tedy bezpečné ho stáhnout do aplikace).

Na straně aplikace by bylo potřeba vytvořit kód, který při každé změně výpůjčky zašle uživateli, který je na výpůjčce uveden a ručiteli dané výpůjčky potvrzovací e-mail, který v sobě bude obsahovat informace o výpůjčce včetně data, do kterého je nutné výpůjčku vrátit.

Xcode a Swift jsou pro tuto možnost vybaveny a je nutné implementovat pouze třídu MFMailComposeViewController. Tato třída v sobě obsahuje všechny metody, které jsou potřeba pro sestavení e-mailu a naplnění ho daty.

V první fázi nebude zasílání e-mailu automatizováno, ale bude vždy využito lokálního e-mailového klienta na stanici, ve kterém bude automaticky zpráva sestavena a připravena k odeslání. Odeslat ji však bude muset uživatel ručně. Tento přístup bude zvolen z důvodu možné potřeby zprávu před odesláním upravit.

V případě, že by se počet uživatelů razantně zvýšil a manuální práce s e-mailovou zprávou by přestala být komfortní, je možné na serveru definovat vlastní e-mailový server a ten poté využít k odeslání těchto zpráv. Takový počet požadavků, kdy by byla tato úprava potřeba, se nicméně v počátku užívání systému nepředpokládá. Níže je možné nahlédnout,

[38] DUBNIČKA, Zdeno. *Využití Mac OSX Server: Bakalářská práce*. Str.34.

jak snadno by tato funkcionalita byla přidána za použití jazyka Swift. Ukázka je vyňata přímo z dokumentace k jazyku.

```
let composeVC = MFMailComposeViewController()
composeVC.mailComposeDelegate = self

// Configure the fields of the interface.
composeVC.setToRecipients(["address@example.com"])
composeVC.setSubject("Hello!")
composeVC.setMessageBody("Hello from California!", isHTML: false)

// Present the view controller modally.
self.presentViewController(composeVC, animated: true, completion: nil)
```

Obr. 15 - Ukázka implementace třídy obsluhující e-mailovou notifikaci.[39]

Z ukázky je zřejmé, že implementace funkcionality a vytvoření e-mailové zprávy jsou velmi jednoduché. Je možné jednoduše poslat zprávu s údaji vypůjčitele, exempláře a ručitele a zmíněná to na základě definovaného rozložení sama zformuluje do e-mailu a připraví k odeslání.

4.3.3 Mobilní přístup

Největším přínosem pro systém z hlediska uživatelů by byla jistě možnost přistupovat do něj pomocí mobilních aplikací, případně přes webové rozhraní. S tímto rozšířením v mysli byl také původně vybrán cloudový databázový poskytovatel. Díky čemu nebude nutné otevírat porty přímo na serveru na fakultě pro přístup zvenčí, ale k datům se bude přistupovat pomocí zabezpečeného protokolu https. Autor věří, že díky tomu by bylo také možné zabránit průnikům do vnitřní sítě univerzity v případě, že by v kódu aplikace bylo nedopatřením zanecháno zranitelné místo.

[39] Ukázka kódu je dostupná online: https://developer.apple.com/library/ios/documentation/MessageUI/Reference/MFMailComposeViewController_class/

Cloudoví poskytovatelé nabízí knihovny pro všechny významné platformy. Díky tomu by bylo možné vytvořit mobilní aplikace pro obě nejpoužívanější platformy: Android i iOS. Díky tomu, že bude desktopová aplikace již funkční, bude možné použít principy, funkce a knihovny z ní na vytvoření mobilní aplikace pro iOS. Jazyk Swift je shodný a bylo by pouze potřeba vytvořit mobilní uživatelské rozhraní a přizpůsobit pro něj kód. Tento zásah není v ekosystému Apple tak drastický, jak by se na první pohled mohlo zdát. Přístup z webové aplikace je také možný a snadno implementovatelný.

Toto rozšíření by dovolovalo zpracovat výpůjčky i ostatním zaměstnancům univerzity a veškerá agenda by se rozložila na více pracovníků. Nicméně se zde stále nepředpokládá přístup studentů do aplikace v současném stavu.

4.3.4 Multiuživatelský přístup

Poslední rozšíření aplikace již představuje její významné přepracování. Nicméně znamená také výrazný krok kupředu. Tímto krokem je vytvoření plného a komplexního knihovního systému, v němž by si sami studenti mohli vyhledávat, rezervovat a půjčovat zařízení.

V této fázi aplikace by musela být plně integrována uživatelská databáze a zároveň vyřešený mobilní přístup k Výpůjčnímu Systému. Dalším požadavkem by bylo vytvoření interaktivního katalogu spolu s vyhledáváním podle více mnohem podrobnějších parametrů (u knih například rok vydání, vydavatelství, obor atd.), aby uživatelé mohli efektivněji vyhledat požadovaný exemplář.

Posledním, neméně významným, zásahem do aplikace by bylo vytvoření další vrstvy, která by obsluhovala studenty a za kterou by studenti neměli přístup. Tato je určena právě pro rezervování výpůjček a pro žádání o výpůjčky. V případě zařízení by každá výpůjčka například musela podstoupit schvalování v rámci katedry a až poté by bylo možné jí v systému rezervovat, případně přímo provést.

Toto schvalování by zároveň vyžadovalo vytvořit pevnou hierarchii v rámci zaměstnanců, aby měl systém jasně definováno, kdo může schválit výpůjčku jakého zařízení a jakému

vypůjčiteli. To zároveň vyžaduje mít připravený a implementovaný systém uživatelských rolí a omezení, podle kterých se budou definovat pravomoci v rámci systému.

Toto rozšíření by ovšem mělo význam ve chvíli, kdy by bylo uvažováno nad tím, nasadit tento Výpůjční Systém v rámci celé fakulty. Autor se nedomnívá, že by v případě jedné katedry byly časové nároky na vývoj opodstatněné, vzhledem k tomu, že aplikace by v tomto případě musela být velmi výrazně přepracována.

5 Závěr

Cílem práce bylo navrhnout a vyvinout Výpůjční Systém pro konkrétní katedru na FJFI ČVUT, který by mohl být provozován na Mac OSX serveru, který byl na pracovišti implementován a konfigurován autorem již dříve. Všechny prvky této aplikace tedy musely být voleny s ohledem na operační systém, na kterém poběží.

Tato specifika byla rozebrána v první části práce, která se zaměřila na charakteristiky vývoje aplikací pro platformy společnosti Apple. Byly probrány možnosti platformy a ekosystému, ze kterého mohou aplikace těžit a také možnosti, které může vývojář pro produkty Apple využít. Následně byly vyjmenovány a popsány způsoby, jakými je možné aplikaci sdílet a instalovat spolu se specifiky každé možnosti.

Poté proběhlo vyjmenování programovacích jazyku, pomocí kterých je možné požadovanou aplikaci vytvořit. Každý jmenovaný jazyk byl vybrán z určitého důvodu a tyto důvody jsou také spolu s výhodami a nevýhodami každého jazyka v textu rozebrány. Jazyky, nad kterými autor uvažoval byly: Java, Smalltalk, Objective-C a Swift. V závěru této části byly vybrány konkrétní jazyky použité pro vývoj, kterými jsou hlavně oficiální programovací jazyk Swift a také Objective-C, ze kterého však byly použity pouze dílčí knihovny. Každý z těchto jazyků je následně podrobněji popsán a charakterizován.

Z výběru jazyka bylo jednoznačné, že pro vývoj bude použito oficiální vývojové prostředí společnosti Apple, Xcode. Díky tomu se dostane aplikaci plné podpory systémových knihoven a také je toto prostředí pro vývoj aplikací pro Mac OSX plně optimalizováno. Navíc poskytuje komfort vývoje kompletní aplikace pouze v jediném okně, kdy jsou všechny prvky (zdrojový kód, design uživatelského rozhraní, výstupy z testování i nastavení samotné distribuce) přístupny na jednom místě.

Když bylo jasné, jaké prostředky pro vývoj budou využity, bylo přistoupeno k samotnému vývoji aplikace. Nejprve byly shrnuty požadavky na její obsah, výkon, chování i vzhled. Z těchto požadavků vyplynul postup vývoje a zdroje, které budou pro běh aplikace potřeba. Bylo zvoleno cloudové databázové řešení PARSE, které neomezí výkon aplikace, ale poskytne jí potenciál růst v budoucnu z pohledu dalších služeb, které může poskytnout.

Také je poskytováno bezplatně, pokud počet požadavků nepřekročí 30 dotazů za vteřinu, což při softwaru pro jednoho uživatele nemůže nastat.

Následně bylo navrženo uživatelské rozhraní aplikace a bylo vytvořeno pomocí prostředí Xcode spolu s veškerým chováním oken a jejich polí. Toto uživatelské rozhraní bylo následně propojeno se zdrojovým kódem, který obsluhuje pozadí aplikace, tedy zápis do a čtení z databáze.

Na konci práce bylo nastíněno několik možností, kam by se Výpůjční Systém mohl ubírat v budoucnu. Jsou mezi nimi například: integrace na fakultní Active Directory server, možnost vývoje mobilní a webové verze aplikace anebo například také její zpřístupnění zbytku fakulty, pokud by o to byl zájem a bylo by svoleno k částečnému přepracování systému, aby toho byl schopen.

6 Seznam použitých zdrojů

6.1 Knižní publikace

MERUNKA, Vojtěch. *Objektové modelování.* 1. Vyd. Praha: Alfa Nakladatelství, 2008. 184 s. ISBN 978-80-87197-04-2

KOCHAN, Stephen G. *Objective-C 2.0, Výukový kurz porgramování pro Mac OSX a iPhone*. Dotisk 1.Vyd. Brno: Computer Press, 2011. 550 s. ISBN 978-80-251-2654-7

ČADA, Ondřej. *Cocoa: úvod do programování počítačů Apple*. 1. vyd. Praha: Grada, 2009. 199 s. ISBN 978-80-247-2778-3

ČADA, Ondřej. *Objektové programování: naučte se pravidla objektového myšlení*. 1. vyd. Praha: Grada, 2009. 200 s. ISBN 978-80-247-2745-5

HART-DAVIS, Guy. *AppleScript: průvodce skriptováním v Mac OS X*. Vyd. 1. Brno: Computer Press, 2011. 382 s. ISBN 978-80-251-3195-4

MYER, Thomas, NEGUS Christopher, CAEN, Francois. *Mac OS X UNIX toolbox: 1000+ commands for Mac OS X power users*. Indianapolis, IN: Wiley Pub., Inc., 2009. 260 s. ISBN 978-0-470-47836-3

DUBNIČKA, Zdeno. *Využití Mac OSX Server: Bakalářská práce*. Praha: Česká Zemědělská Univerzita – Provozně ekonomická fakulta, 2012

6.2 Internetové zdroje

Cloud a právo: pozor na umístění vašich dat. In *root.cz*. [online], vytvořeno 25.1.2016 [citováno 25.3.2016]. KREISL, Marek, PIKAL, Daniel. Dostupné na WWW: http://www.root.cz/clanky/cloud-a-pravo-pozor-na-umisteni-vasich-dat/

Cocoa and Objective-C cookbook: move beyond basic Cocoa development using over 70 simple and effective recipes for Mac OS X development [online]. Birmingham, U.K.: Packt Pub., 2011 [cit. 2016-03-30]. HAWKINS, Jeff. Dostupné na WWW: http://site.ebrary.com/lib/natl/Doc?id=10477262

The Swift Programming Language, Swift 2 Edition [online]. Cupertino, California USA: Apple Inc., 2016. Kolektiv autorů. Dostupné na WWW: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/index.html#//apple_ref/doc/uid/TP40014097-CH3-ID0

Using Swift with Cocoa and Objective-C [online]. Cupertino, California USA: Apple Inc., 2016. Kolektiv autorů. Dostupné na WWW: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/BuildingCocoaApps/#//apple_ref/doc/uid/TP40014216-CH2-ID0

TIOBE Index for March 2016 [online]. Eindhoven, Netherlands: TIOBE Software, 2016. Dostupné na WWW: http://www.tiobe.com/tiobe_index?page=index

Workspace Window Overview [online]. Cupertino, California USA: Apple Inc., 2016. Dostupné na WWW: https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/TheWorkspaceWindow.html#//apple_ref/doc/uid/TP40010215-CH25-SW1

MFMailComposeViewController [online]. Cupertino, California USA: Apple Inc., 2016. Dostupné na WWW: https://developer.apple.com/library/ios/documentation/MessageUI/Reference/MFMailComposeViewController_class/