Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Java aplikace - teorie a praxe

Aneta Doubková

© 2017 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Aneta Doubková

Informatika

Název práce

Java aplikace – teorie a praxe

Název anglicky

Java applications – theory and practice

Cíle práce

Diplomová práce je tematicky zaměřena na problematiku vývoje aplikací v jazyce Java. Hlavním cílem práce je analyzovat současný stav v programování aplikací v jazyce Java.

Dílčí cíle diplomové práce jsou:

- analýza stávající konkrétní aplikace pro OS Windows a návrh a implementace její podoby pro Mac OS a nejpoužívanější distribuce OS Linux

- v rámci návrhu by mělo dojít k optimalizaci a zjednodušení oproti původní aplikaci a také ke zlepšení uživatelského rozhraní

- první verze aplikace, tedy verze, kterou se bude zabývat tato práce, bude obsahovat jen základní funkčnost.

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Dále také na vlastních zkušenostech získaných při tvorbě aplikace. Na základě poznatků teoretické části a výsledků praktické části budou formulovány závěry a doporučení pro další budoucí vývoj (např. rozšíření o funkčnost chybějící oproti původní aplikaci napsané pro OS MS Windows)

Doporučený rozsah práce

60 -80 stran

Klíčová slova

Java aplikace, IDE, MyQ, porovnání aplikace v ostatních OS

Doporučené zdroje informací

ČADA, O. Objektové programování – naučte se pravidla objektového myšlení. Praha: Grada. 2009. ISBN 978-80-247-2745-5.

DARWIN, I., F. Java – Kuchařka programátora. Brno: Computer Press. 2006. 800 str. ISBN 80-251-0944-5.

ECKEL, B. Myslíme v jazyku Java: knihovna zkušeného programátora. Praha: Grada. 2001. ISBN 80-247-0027-1.

HEROUT, P. Java : grafické uživatelské prostředí a čeština. 1. vyd. České Budějovice : KOPP, 2001. 316 str. ISBN 80-7232-150-1.

HEROUT, P. Učebnice Jazyka Java. 2. vyd. České Budějovice: KOPP. 2006. 349 str. ISBN 80-7232-115-3.

KEOGH, J. Java bez předchozích znalostí. Brno: Computer Press. 2012. ISBN 978-80-251-0839-0.

KISZKA, B. 1001 tipů a triků pro jazyk Java. Brno: Computer Press. 2009. ISBN 978-80-251-2467-3.

SCHILDT, H. Java 7 – Výukový kurz. Brno: Computer Press. 2012. ISBN 97880-251-3748-2.

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Čestmír Halbich, CSc.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 18. 10. 2016

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 10. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 24. 03. 2017

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Java aplikace - teorie a praxe“ jsem vypracovala 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 autorka uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušila autorská práva třetích osob.

V Praze dne 29. března 2017

Aneta Doubková

Poděkování

Ráda bych touto cestou poděkovala Ing. Čestmíru Halbichovi, CSc. za odborné vedení a poskytnuté rady v průběhu psaní diplomové práce. Dále bych ráda poděkovala zástupcům společnosti MyQ za poskytnutí tématu a kolegům z QA oddělení za otestování vytvořené aplikace. Velké díky také patří rodině a všem, kteří mě podporovali během tvorby této práce.

Java aplikace - teorie a praxe

Souhrn

Diplomová práce je zaměřena na vývoj aplikací v programovacím jazyce Java. Nejdříve byla provedena analýza aktuálně používaných Java technologií následovaná analýzou konkrétní aplikace pro OS Windows. Na základě provedených analýz byly vybrány vhodné technologie pro řešení praktické části této práce. Vybrané technologie byly následně použity k realizaci aplikace primárně určené pro Mac OS.

Klíčová slova: Java, JavaFX, Intellij IDEA, Ant, Maven, UDP, HTTP, síťový tisk

Java applications - theory and practice

Summary

The thesis is focused on application development in programming language Java. Firstly was done the analysis of the current Java technologies, followed by the analysis of concrete application for OS Windows. According the performed analysis were chosen suitable technologies for practical solution of this thesis. The chosen technologies were used for realisation of the application primarily designed for Mac OS.

Keywords: Java, JavaFX, Intellij IDEA, Ant, Maven, UDP, HTTP, network printing

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Java je univerzální, platformně nezávislý programovací jazyk, vyvíjený společností Sun Microsystems. Java je zdarma dostupná pro různé operační systémy a je oblíbená zejména pro dlouhodobou kompatibilitu, která zaručuje fungování aplikací vyvinutých na jejích starších verzích.

Ačkoliv se Java v dnešní době nerozšiřuje tak rychle, jako například javascriptový framework NodeJS, je stále nejrozšířenějším a jedním z nejpopulárnějších programovacích jazyků.

To vše jsou důvody toho, proč je v žebříčku pracovních míst jedním z nejpoptávanějších programovacích jazyků. Podle průzkumu New Relic uvedeného na webu `blog.newrelic.com` za prvních pět měsíců roku 2016 jasně vedla nad ostatními poptávanými programovacími jazyky.

Zejména rozšířenost, dlouhodobá kompatibilita a nezávislost na operačním systému způsobují, že se výborně hodí pro implementaci nové aplikace Easy Job Manager společnosti MyQ.

Easy Job Manager je nástroj pro správu tisku vyvinutý pro OS Windows. Jedná se o aplikaci, která je součástí tiskového systému MyQ - jednoduchého systému určeného pro sledování a účtování tiskových úloh.

Kapitola 2

Cíl práce a metodika

2.1 Cíl práce

Diplomová práce je tematicky zaměřena na problematiku vývoje aplikací v jazyce Java. Hlavním cílem práce je analyzovat současný stav v programování aplikací s využitím tohoto programovacího jazyka a na základě této analýzy vybrat vhodné technologie pro řešení praktické části této práce, jež se bude zabývat implementací kontrétní aplikace.

Dílčími cíli diplomové práce jsou analýza stávající konkrétní aplikace pro OS Windows, dále návrh a implementace její podoby pro Mac OS a nejpoužívanější distribuce OS Linux. V rámci návrhu by mělo dojít k optimalizaci a zjednodušení oproti původní aplikaci a také ke zlepšení uživatelského rozhraní. V rámci této práce bude implementovaná první verze aplikace. Tato první verze bude obsahovat základní funkčnost.

2.2 Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Dále také na vlastních zkušenostech získaných při tvorbě aplikace.

Na základě poznatků teoretické části a výsledků praktické části budou formulovány

závěry a doporučení pro další budoucí vývoj (např. rozšíření o funkčnost chybějící oproti původní aplikaci napsané pro OS MS Windows).

Kapitola 3

Přehled řešené problematiky

V první části této kapitoly jsou vysvětleny pojmy a teorie nutné pro pochopení řešení zadaného projektu. Dále je popsána provedená analýza stávající aplikace pro OS Windows, jež je vzorem pro tvorbu nové aplikace. Následuje přehled požadavků získaných na základě analýzy. Požadavky kladené na vyvíjenou aplikaci jsou zároveň kritérii pro výběr technologií.

Druhá část kapitoly se zabývá hlavním cílem diplomové práce, analýze současného stavu v programování aplikací v jazyce Java.

3.1 Obecné pojmy

3.1.1 Síťový tisk

Jak již název napovídá, jedná se o tisk přes počítačovou síť, do níž jsou zapojeny tzv. síťové tiskárny (anglicky network printers). Připojení tiskárny do sítě umožňuje[1] sdílení jedné tiskárny mezi více uživateli, respektive mezi více pracovními stanicemi. Tiskárny a pracovní stanice jsou přes počítačovou síť propojeny zařízením (či aplikací) nazývaným tiskový server (anglicky print server).[22][4]

[1]Na rozdíl od lokálního tisku, kdy je tiskárna připojena pouze k jedné pracovní stanici.

3.1.2 Komunikace klient - server

Způsob komunikace klient - server spočívá v tom, že server poslouchá na určitém portu a klient na něj odesílá data. Jedná se o přenos dat mezi dvěma koncovými uzly (tzv. end-to-end komunikaci). Budeme-li brát referenční model ISO/OSI[2], stará se o přenos dat mezi dvěma koncovými uzly transportní vrstva. V prostředí sítí na bázi protokolů TCP/IP je přenos možné realizovat spojově orientovaným protokolem TCP nebo nespojově orientovaným protokolem UDP.[13]

Transmission Control Protocol[3]

Transmission Control Protocol, dále TCP, zajišťuje přenos dat se zaručeným doručením v pořadí, v jakém byla data odeslána a bez duplicit. Spolehlivého doručování lze dosáhnout díky vytvoření spojení (někdy nazýváném tunel nebo roura), v němž lze přenášet data oboustranně.

Data (pakety) jsou rozdělena do tzv. TCP segmentů a následně posílána na server až do doby jejich úspěšného doručení. Tím je zaručeno doručení všech paketů.

Pořadí doručení zaručuje mechanismus potvrzování paketů na straně serveru. Potvrzování funguje na principu, kdy pakety, které následují po odeslaném paketu, nemohou být doručeny dříve, než je doručen předchozí paket.

User Datagram Protocol[4]

User Datagram Protocol, dále UDP, neposkytuje záruky zaručované protokolem TCP. Nezaručuje doručení všech odeslaných dat, nezaručuje ani pořadí, v jakém budou data doručena a data mohou být doručena duplicitně.

Jednotlivá data (zprávy) jsou odesílána nezávisle na sobě. Tento protokol je vhodný pro aplikace, kdy by zpoždění (anglicky delay) způsobené čekáním na doručení všech

[2]Referenční model otevřené komunikace (OSI – Open Systems Interconnection) vypracovaný mezinárodní organizací ISO za účelem standardizace počítačových sítí.[7]

[3]Specifikace protokolu dostupná na `https://tools.ietf.org/html/rfc793`

[4]Specifikace protokolu dostupná na `https://www.ietf.org/rfc/rfc768.txt`

dat v pořadí, v jakém byla data odeslána, činilo problémy.

Porovnání TCP a UDP

Rozdíly mezi protokoly TCP a UDP jsou shrnuty v tabulce 3.1.

TCP	**UDP**
- Transmission Control Protocol	- User Datagram Protocol
- se zárukami (doručeno vše, pořadí doručení, žádné duplicity)	- bez záruk
- komplexní a složitý protokol	- jednoduchý protokol
- vytvořeno spojení → umožňuje oboustranný přenos	- data odesílána nezávisle na sobě
- vhodný pro aplikace, které vyžadují spolehlivý přenos	- vhodný pro aplikace, kde je rychlost nad spolehlivostí
- jednotka přenosu: TCP segment	- UDP datagram
- používán protokoly: HTTP, HTTPs, FTP, SMTP, Telnet	- používán protokoly: DNS, DHCP, TFTP, SNMP, RIP, VOIP

Tabulka 3.1: Porovnání protokolů TCP a UDP[14][21]

3.1.3 Tiskový server

Tiskový server má na starosti řízení tisku. Úlohy odeslané k vytisknutí nejsou posílány přímo na tiskárnu, ale jsou zařazovány to tiskové fronty (anglicky print queue), kde jsou dočasně uloženy před samotným tiskem.[23]

Zařazování tiskových úloh do fronty se říká **job spooling** a umožňuje tisk tiskových úloh až po tom, co jsou přeneseny[5], dále rozložení tisku mezi více tiskáren, účtování tisku, apod.

[5]Na rozdíl od tisku je přenos rychlá operace a pracovní stanice tak není zatěžována během tisku.

3.2 MyQ Easy Job Manager

MyQ Easy Job Manager, dále EJM, je, spolu s dalšími aplikacemi a komponentami, součástí komplexního tiskového systému MyQ, proto nejprve pár slov o tomto systému.

3.2.1 MyQ

MyQ je systém pro správu tiskových úloh, který řeší:

- ochranu a zabezpečení dat prostřednictvím terminálů s podporou identifikace uživatelů,
- úsporu prostředků na provoz tiskových zařízení tím, že poskytuje přehledné reporty o tom CO, KDO, KDY a KDE[6] tisknul či kopíroval a umožňuje tím nastavit pro zaměstnance pravidla tisku nebo rovnou stanovení limitů,
- úsporu času zaměstnanců poskytnutím funkcí zjednodušujících proces tisku.[1]

3.2.2 Stávající aplikace EJM

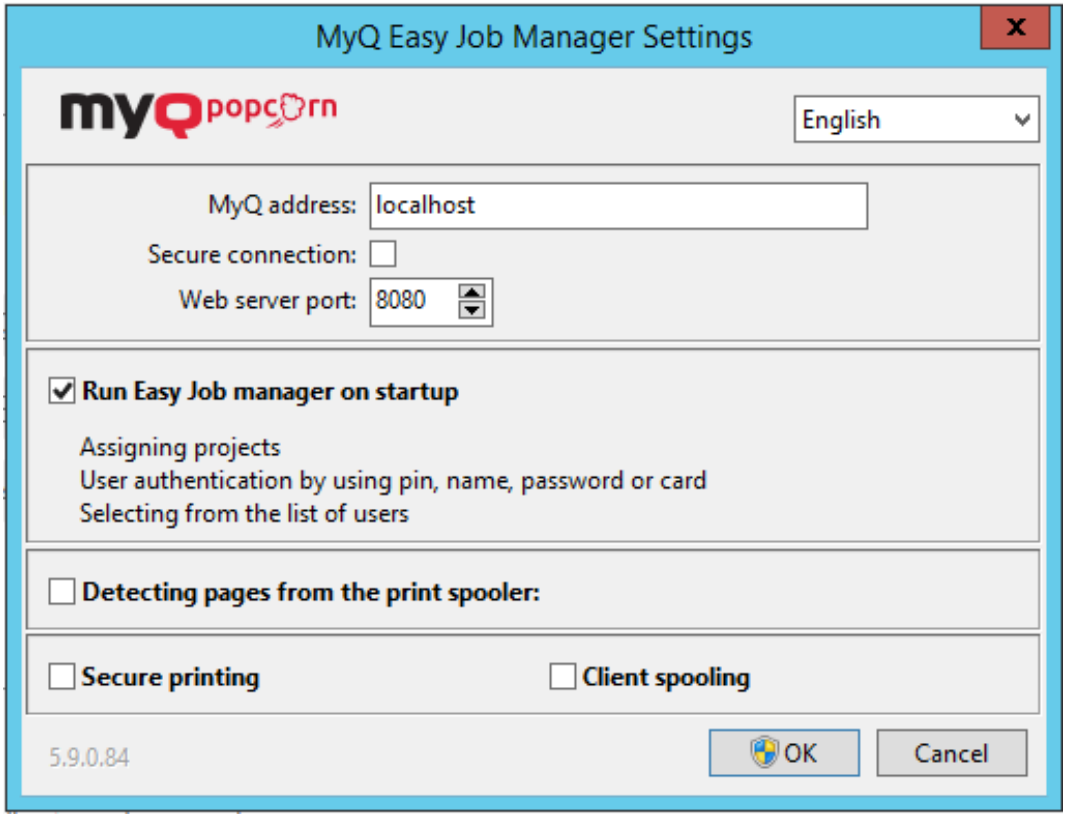
EJM je nástroj pro správu tisku vyvinutý pro OS Windows. Instalátor aplikace je dodáván spolu s MyQ serverem a je dostupný ve složce MyQ. Po instalaci je spuštěna aplikace Easy Job Manager, která běží jako tzv. TrayApp a má ikonu v notifikačním panelu. Ikona pod sebou skrývá menu pro rychlý přístup k hlavnímu oknu aplikace, nastavení a ukončení aplikace.

Nastavení

K nastavení má přístup pouze uživatel s oprávněním, tedy Administrátor. Je přístupné výběrem `Settings Easy Job Manager` z hlavní nabídky Windows nebo zvolením volby `Settings` umístěné v nabídce pod ikonou aplikace[7].

[6] Na jakém tiskovém zařízení.

[7] Nabídka se zobrazí na pravé kliknutí.



Obrázek 3.1: Easy Job Manager - Nastavení

Dialog *Nastavení* je rozdělen do šesti částí, jak je patrné z obrázku 3.1.

1. Záhlaví obsahuje logo a výběr jazyka. Tento výběr jazyka slouží pouze pro dialog *Nastavení*, nikoliv pro celou aplikaci. Pro další části aplikace může uživatel nastavit jazyk v hlavním okně aplikace.

2. Nastavení spojení s MyQ, tedy adresa serveru, volba bezpečného spojení a nastavení.

3. Zvolením volby `Run Easy Job manager on startup` administrátor zařídí, že bude aplikace vždy spuštěna po startu systému.

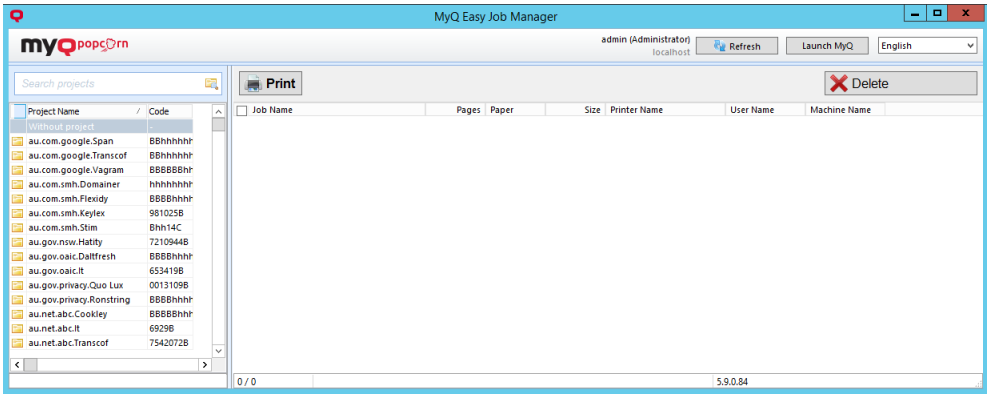
4. Další část se týká nastavení detekce stránek tiskových úloh na uživatelem specifikovaných typech portů.

5. V této části má uživatel na výběr mezi dvěma možnostmi: `Secure Printing` a `Client Spooling`. Zvolením možnosti `Secure Printing` uživatel nastaví, že MyQ Easy Job Manager přijme tiskové úlohy na síťové adrese *localhost:515* a pošle je do MyQ tiskového serveru. Vybrání možnosti `Client Spooling` naopak způsobí, že budou tiskové úlohy po přijetí poslány přímo do tiskového zařízení [8] nebo po autentizaci na terminálu[9].

6. Poslední částí je zápatí obsahující číslo verze aplikace a tlačítka pro potvrzení a zrušení změn.

Hlavní okno aplikace

Snímek hlavního okna aplikace je zobrazen na obrázku 3.2. Nejdůležitější částí hlavního okna aplikace je seznam tiskových úloh připravených k tisku. Pokud jsou zapnuté projekty, zobrazuje se v levé části hlavního okna seznam dostupných projektů včetně volby *Without project*[10] a filtru pro snažší vyhledávání v seznamu projektů.



Obrázek 3.2: Easy Job Manager - Hlavní okno

Okno nelze minimalizovat na panel spuštěných aplikací, je vždy minimalizováno do

[8] Je-li nastavena tisková fronta typu *direct queue*.

[9] Je-li nastavena tisková fronta typu *follow me*

[10] Volba umožňující tisk bez přiřazeného projektu, je-li povolena.

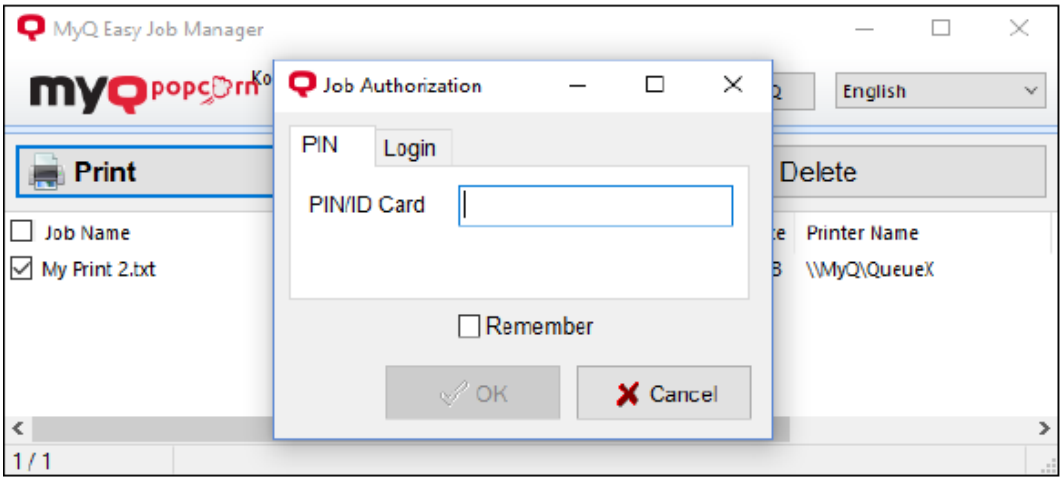
ikony v notifikačním panelu. V případě změny seznamu tiskových úloh je okno automaticky zobrazeno s volbou *always on top*, tedy vždy na vrchu. Součástí seznamu tiskových úloh jsou dvě akční tlačítka, jedno pro tisk, druhé pro smazání vybraných tiskových úloh.

Autorizace tiskové úlohy

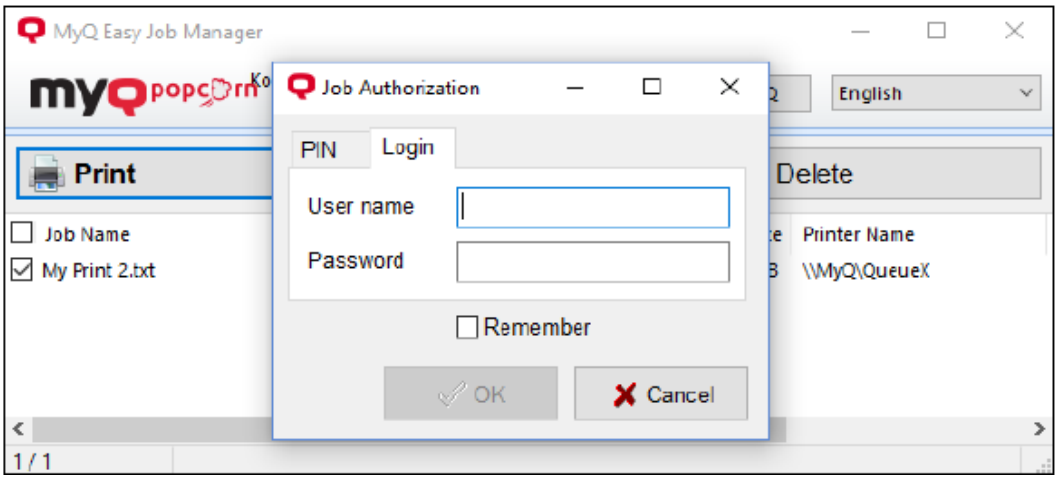
Autorizace vybraných tiskových úloh může být provedena zadáním přihlašovacího jména a hesla nebo zadáním PINu.

Autorizační dialog je stejný pro oba způsoby autorizace. Pro každý ze způsobů autorizace obsahuje jednu záložku. Autorizační dialog s aktivní záložkou pro zadání PINu je zobrazen na obrázku 3.3, autorizační dialog s aktivní záložkou pro zadání přihlašovacího jména a hesla je zobrazen na obrázku 3.4.

Uživatel volí způsob autorizace přepínáním mezi těmito dvěma záložkami. Obě tyto záložky obsahují volbu `Remember`. Jejím zvolením a potvrzením dialogu uživatel docílí, že pro každý další tisk už nebudou vyžadovány přihlašovací údaje.



Obrázek 3.3: Easy Job Manager - Autorizace pomocí PINu



Obrázek 3.4: Easy Job Manager - Autorizace pomocí hesla

3.2.3 Používané protokoly

Nová aplikace bude fungovat na podobných principech jako ta stávající[11] a bude využívat stejné Server API, proto si zde popíšeme základní princip komunikace se serverem. Technickým detailům se budeme věnovat až v praktické části této práce.

Komunikace se serverem je realizovaná částečně prostřednictvím HTTP požadavků, a částečně protokolem UDP.

HTTP komunikace je používána pro požadavky, u nichž je vyžadovaná odpověď a je inicializovaná ze strany aplikace, jakožto klienta. Příkladem jejího použití může být situace, kdy aplikace potřebuje získat nastavení uložené na serveru, seznam projektů, které je možné přiřadit k tiskovým úlohám nebo pro zahájení tisku.

UDP komunikace slouží zejména serveru, aby mohl informovat aplikaci o změně stavu. Například o nových tiskových úlohách nebo o nepovedené autentizaci uživatele.

3.2.4 Požadavky na novou aplikaci EJM

Jak již bylo řečeno v úvodu této diplomové práce, cílem projektu je vytvořit nástroj pro správu tisku, který poběží na Mac OS X a nejpoužívanějších distribucích OS Linux.

[11] Verze EJM pro OS Windows.

To, že se aplikace vyvíjí znovu, využijeme k přehodnocení té stávající a je možné, že dojde během vývoje k přidání, změně či odebrání některých vlastností.

První verze aplikace bude podporovat autorizaci tiskových úloh a přiřazení projektů. Podporovaný OS bude primárně Mac OS X, takže specifika jako jsou klávesové zkratky pro usnadnění ovládání aplikace, budou řešena v první řadě pro něj, design bude co nebližší nativním aplikacím tohoto operačního systému, apod. Po celou dobu bude kladen důraz na zjednodušení workflow správy tiskových úloh a minimalistické GUI[12].

Uživatelé - aktéři

- Administrátor - uživatel, který má přistup k nastavení aplikace a může jej měnit.
- Uživatel pro komunikaci - uživatel, pod jehož přihlašovacím jménem aplikace komunikuje se serverovou částí[13].
- Uživatel pro autorizaci - uživatel, pod jehož přihlašovacími údaji jsou autorizovány tiskové úlohy.

Katalog funkčních uživatelských požadavků

Následuje seznam funkčních požadavků. Každá položka ze seznamu obsahuje název funkčního požadavku, jeho stručný popis a přiřazenou prioritu. Stupnice priorit má tři stupně a je uvedená v tabulce 3.2.

Priorita	**Kategorie**
1	must have
2	nice to have
3	future improvement

Tabulka 3.2: Stupnice priorit uživatelských požadavků

[12]GUI - Graphical User Interface

[13]Uživatel pro komunikaci je uživatelem, pod kterým je spuštěna aplikace.

Z uvedené tabulky je patrné, že v rámci této diplomové práce je nutné splnit funkční požadavky s prioritou 1. Pokud zbyde čas, může být funkčnost aplikace rozšířena o funkční požadavky s prioritou 2 a s implementací funkčních požadavků s prioritou 3 je počítáno jako s rozšířením či vylepšením do budoucna.

1. **Přihlášení uživatele OS**

   Popis: Aplikace bude umožňovat automatické přihlášení uživatele, který je v daném okamžiku přihlášen na desktopu, a pod kterým je aplikace spuštěna. Pod přihlašovacím jménem tohoto uživatele bude aplikace komunikovat s MyQ (resp. se serverem).

   Priorita: 1

2. **Přehled nevytisknutých tiskových úloh**

   Popis: Aplikace bude uživateli zobrazovat seznam nevytisknutých úloh se základními údaji o těchto úlohách.

   Priorita: 1

3. **Správa tiskových úloh**

   (a) **Řazení nevytisknutých tiskových úloh**

   Popis: Aplikace bude uživateli umožňovat řadit seznam tiskových úloh na základě uvedených jednotlivých informací o těchto úlohách.

   Priorita: 2

   (b) **Multiselekce**

   Popis: Aplikace bude uživateli umožňovat práci s více úlohami najednou.

   Priorita: 2

   (c) **Tisk nevytisknutých tiskových úloh**

   Popis: Aplikace bude uživateli umožnovat tisk tiskových úloh ze seznamu.

   Priorita: 1

   (d) **Mazání nevytisknutých tiskových úloh**

   Popis: Aplikace bude uživateli umožnovat mazání tiskových úloh ze se-

znamu.

Priorita: 1

4. **Autorizace tiskové úlohy**

   Popis: Během tisku bude aplikace umožňovat přihlášení uživatelů na základě uživatelských údajů (uživatelského jména a hesla nebo PINu), a tímto přihlášením autorizovat tisk vybraných tiskových úloh.

   Priorita: 1

5. **Přiřazování projektů**

   Popis: Aplikace bude uživateli umožňovat každé tiskové úloze vybrané k tisku přiřadit projekt ze seznamu dostupných projektů.

   Priorita: 1

6. **Vyhledávání projektů**

   Popis: Aplikace bude umožňovat během přiřazování projektů rychlé vyhledávání projektů pomocí textového filtru.

   Priorita: 2

7. **Zapamatování uživatele**

   Popis: Aplikace bude umožňovat zapamatování piřhlašovacích údajů uživatele, pod nímž byla autorizována alespoň jedna tisková úloha.

   Priorita: 2

8. **Odhlášení uživatele**

   Popis: Aplikace bude umožňovat zapomenutí piřhlašovacích údajů uživatele, pod nímž byla autorizována alespoň jedna tisková úloha. (navazuje na funkční požadavek č. 3)

   Priorita: 2

9. **Lokalizace**

   Popis: Aplikace bude vícejazyčná, bude podporovat 29 jazyků. Požadované do-

končené překlady pro potřeby této diplomové práce jsou angličtina a čeština.

Priorita: 1

10. **TrayApp**

    Popis: Bude-li to podporováno OS, aplikace poběží jako ikona v notifikačním panelu OS.

    Priorita: 1

11. **Podpora Mac OS X**

    Popis: Aplikace bude podporovaná na Mac OS X.

    Priorita: 1

12. **Podpora Linux**

    Popis: Aplikace bude podporovaná na OS Linux.

    Priorita: 3

13. **O aplikaci**

    Popis: Aplikace bude obsahovat základní informace o své verzi a odkaz na webové stránky společnosti, která aplikaci dodává.

    Priorita: 2

14. **14. Spustit MyQ portál**

    Popis: Aplikace bude umožňovat jednoduše spustit MyQ portál v defaultním webovém prohlížeči.

    Priorita: 1

Nefunkční uživatelské požadavky

- user-friedly interface (snadná použitelnost)
- možnost rozšíření do budoucna
- bezpečnost

- desktopová aplikace
- přizpůsobení designu prostředí operačního systému pro Mac OS X

Slovníček pojmů

Tisková úloha

Tisková úloha je položka určená k tisku.

Projekt

Jednou z podporovaných funkcí tiskového systému MyQ je umožnit účtování tisku jednotlivých tiskových úloh na projekty. Tuto funkci podporuje i aplikace EJM.

Administrátor

Pro účely tohoto dokumentu je pojmem Administrátor myšlen uživatel s oprávněním. Tento uživatel má právo měnit nastavení aplikace EJM.

Uživatel

Uživatelem je myšlen každý uživatel, který pracuje s aplikací EJM a není Administrátor[14].

Tiskový server

Tiskový server řídí tiskové úlohy do jednotlivých síťových tiskáren v rámci sítě.

Autorizační dialog

Autorizační dialog v kontextu aplikace EJM slouží k autorizaci vybraných tiskových úloh. Uživatel je vyzván k zadání přihlašovacích údajů, které jsou odeslány na server za účelem jeho identifikace. V případě úspěšné autentizace následuje autorizace neboli ověření, že je uživatel oprávněn k tisku.

[14]Může se tedy jednat o uživatele pro komunikaci nebo o uživatele pro autorizaci, jak je popsáno výše v této kapitole.

3.3 Programovací jazyk Java

"Java je v dnešní době fenomén, který je skloňován ve všech pádech. Za pět let od svého vzniku prošla neuvěřitelným rozvojem a stejně tak neuvěřitelně začíná ovlivňovat i dění v počítačovém světě."[8]

Toto napsal o programovacích jazyce Java Pavel Herout ve své knize již v roce 2000. V tom roce bylo očekáváno vydání JDK[15] verze 1.3.0. Od té doby se mnohé změnilo. První výraznější změny přišly s verzí JDK 1.5. Jednalo se nejen o změny jazyka, ale také o změny knihovní třídy Java Core API[16].[9]

Tato kapitola se zabývá programovacím jazykem Java a technologiemi s ním souvisejícími. Nejprve je popsána historie a nejdůležitější vlastnosti tohoto jazyka. Následně jsou analyzovány konkrétní technologie související s programováním aplikací v jazyce Java.

3.3.1 Historie

Počátkem vývoje programovacího jazyka Java je rok 1991, kdy firma Sun Microsystems začala s vývojem programovacího jazyka založeném na principech C a C++. Tento jazyk byl původně pojmenován Oak (česky dub) podle stromu, který stál před oknem vedoucího týmu, pana Goslinga. Protože však už jeden jazyk s tímto názvem existoval, došlo k přejmenování. Novým názvem se stal americký slangový výraz pro kávu, tedy Java. Oficiálně byla Java představena firmou Sun na konferenci v květnu 1995.[8]

Kořeny v C a C++

„Java je spřízněna s jazykem C++, který je zase přímým potomkem jazyka C."[24, strana 39]

[15]Zkratka pro **Java Development Kit**, soubor nástrojů potřebných pro vývoj programů v jazyce Java.

[16]Zkratka pro **Application Programming Interface**, tisíce knihovních tříd, které obsahuje každé prostředí, kde se používá Java.[9]

Tak zní jedna z prvních vět o původu programovacího jazyka Java v knize mistrovství Java, kompletní průvodce vývojáře. Z citované věty je pochopitelné, že mezi těmito programovacími jazyky existují patrné podobnosti. Mají společné základní rysy a velmi podobnou syntaxi.

Vznik jazyka C byl, a dodnes je, považován za velký převrat v oblasti programování. Vznikl jednak v době, kdy požadavky na sofware přerostly možnosti programátorů tyto požadavky zrealizovat:
"Vznik jazyka C je přímým důsledkem potřeby strukturovaného, efektivního a vysokoúrovňového jazyka, který by byl schopen při vytváření systémových programů nahradit jazyk symbolických instrukcí (assembler)."[24]
Zároveň vnikl také v době, kdy se hardware stával běžně dostupným pro spoustu lidí a vznikl tak prostor pro různé experimenty. Jedná se vůbec o první programovací jazyk, který pro sebe vytvořili sami programátoři.

Požadavky na software ale postupem času stále rostly, spolu s nimi také složitost algoritmů, a proto bylo nutné začít hledat způsob, jakým se s touto složitostí vypořádat. Výsledkem tohoto hledání se stal v roce 1979 v laboratořích firmy Bell programovací jazyk C++. Jedná se o rozšíření jazyka C o objektově orientované prvky.

Od strojových instrukcí k OOP[17]

Jak již bylo řečeno výše, požadavky na software se od dob vzniku počítačů postupně zvyšovaly, a tím narůstala i složitost programů, které tyto požadavky splňovaly. Stupňování této složitosti může být vyjádřeno postupným vývojem, jakým procházely programovací jazyky. Vývojem, který umožnil vždy o kousek posunout hranice, které stanovují, kdy se stává složitost řešení daného problému neúnosnou.

Následuje přehled vývoje programovacích jazyků tak, jak šel v historii za sebou:

strojové instrukce → *assambler* → *vysokoúrovňové programovací jazyky* (FORTRAN) → *strukturované programování* → **OOP**

[17]OOP – Objektově Orientované programování

Než se dostaneme zpět ke vzniku pogramovacího jazyka Java, pozastavme se na chvíli nad krátkou definicí a vypíchnutím nejdůležitějších pojmů pro poslední dva uvedené "vývojové stupně":

- **Strukturované programování** – problém je řešen pomocí algoritmu, který se skládá z několika dílčích úloh (nazývaných funkce či procedury). Tento způsob programování využívá pouze tři základní řídící struktury: sekvence, výběr, opakování.

- **Objektově orientované programování** – na řešený problém je nahlíženo jako na problém reálného světa. Klíčovými pojmy zde jsou: objekt, dědičnost, abstrakce, zapouzdření a polymorfizmus.

Ani vznik, pro mnohé dokonalého, programovacího jazyka C++ neznamenal konec zvyšování nároků a složitosti, a revoluce v programování tedy nemohla být ukončena. Následně totiž došlo k masivnímu rozšíření Internetu a vzniku webu, a reakce na sebe nenechala dlouho čekat v podobě vzniku Javy. Jak je uvedeno výše, práce na vývoji začaly v roce 1991.

Ačkoliv původním impulsem pro vznik Javy byla potřeba **platformně nezávislého** jazyka[18], aby ten který program nemusel být vždy kompilován pro konkrétní typ procesoru. Nakonec se v rámci jejího vývoje staly důležitějším faktorem již zmiňované masivní rozšíření Internetu a vznik webu. Pro web bylo potřeba psát přenosné (neboli platformně nezávislé) programy víc, než kdy předtím. Internet je místem, ke kterému je připojeno mnoho různých platforem. Cílem bylo, spouštět na všech těchto platformách stejné programy. Pozornost byla přesunuta k "programování pro Internet".

Jazyk Java se stal dalším programovacím jazykem, který navrhli **programátoři pro programátory**. Její vývoj je opřen o jejich zkušenosti a ovlivněn jejich potřebami.

Tvůrci jazyka využili toho, že již existují kvalitní programovací jazyky, jejichž syntaxe a principy jsou dobře známy. Předpokládali, že tato vlastnost by mohla zaujmout

[18] Univerzálního jazyka nezávislého na architektuře počítače.

spoustu programátorů, pro něž bude naučit se programovacímu jazyku, který je inspirován již existujícími programovacími jazyky snadné. Jazyk Java, jazyk C a jazyk C++ mají proto společné vlastnosti, avšak nejsou mezi sebou kompatibilní.

Bezpečnost - Přenositelnost - Bajtkód

Výstupem kompilátoru Javy je sada instrukcí spustitelná v JVM[19] nazývaná bajtkód. Nejedná se o přímo spustitelný kód, což umožňuje snadněji spustit program v rámci různých prostředí. Existují různé virtuální stroje Javy pro různá prostředí, a tím je odstíněn typ procesoru od konkrétního bajtkódu.

Kromě zajištění přenositelnosti programů tento způsob překladu zdrojových kódů zajišťuje zároveň také jejich bezpečnost. O řízení běhu programu se totiž stará virtuální stroj Javy a neumožní žádné akce mimo.

Je obecně známo, že nepřímá kompilace způsobuje pomalejší běh programu. V tomto případě se však jedná o „vysoce optimalizovaný“ bajtkód, který lze spouštět velmi rychle. Navíc Java, v případě potřeby, neodepírá možnost kompilace bajtkódu do nativního kódu. Řešením je tzv. Just-In-Time, zkráceně JIT, kompilátor bajtkódu.

Just-In-Time

JIT kompilátor zajišťuje, že jsou vybrané části bajtkódu jedna po druhé kompilovány v reálném čase do spustitelného kódu. Děje se tak na základě okamžité potřeby. Je důležité si uvědomit, že do spustitelného kódu není kompilován celý javový program, ale pouze jeho část. Důvodem je, že určité části kódu je možné kompilovat až v době vykonávání programu.[24]

Kompilace programu

Program je kompilován pomocí kompilátoru, který může být spuštěn z příkazového řádku tak, jak je to uvedené v následující ukázce:

[19] Zkratka pro **Java Virtual Machine**, běhové prostředí jazyka Java, interpretér bajtkódu.

```
$ javac Main.java
```

Kompilace, nebo-li překlad, probíhá do pseudojazyka, který je česky nazýván bajtkód (anglicky byte-code). Přeložený program je následně uložen do souboru s příponou `.class`.[8]

Výstupem uvedeného příkazu bude soubor `Main.class`, který spustíme následujícím příkazem:

```
$ java Main
```

Objektově orientované programování

Objektově orientované programování, zkráceně OOP. Na rozdíl od procesního modelu, který spočívá v lineárním zpracování kódu a bývá často popisován také jako kód, který pracuje s daty, přistupuje objektově orientované programování k řešení problému opačným způsobem, a to tak, že se jedná o data, která řídí přístup k programu.

U procedurálního přístupu se s rostoucí velikostí a složitostí programů[20] vyskytuje řada problémů, které se postupně stávají neúsnosnými. Složitost je v objektově orientovaném přístupu řízena prostřednictvím abstrakce.

Abstrakce je velmi důležitou součástí OOP, je jeho základním stavebním kamenem. Díky abstrakci uvažuje programátor o reálném světě jako o objektech, z nichž každý má své jedinečné chování. Tímto způsobem uvažování je odstíněn od spousty zbytečných detailů, které byl nucen řešit v procesním modelu.[24]

Jednotlivé objekty mohou sestávat z více jednotlivých podobjektů, které společně tvoří jeden celek. Toto je v objektově orietovaném přístupu vyřešeno tzv. hierarchií. Pojem hierarchie bude podrobněji vysvětlen dále, v principech OOP.

OOP má tři obecně známé základní principy:

1. **Zapouzdření**

   Základem objektově orientovaného programování je třída. Každá třída definuje

[20] Příkladem procedurálního programovacího jazyka je například již zmiňovaný jazyk C.

vlastnosti (prostřednictvím proměnných) a chování (prostřednictvím metod). Úkolem zapouzdření je zabezpečit přístup k vlastnostem a chování třídy proti jejich nesprávnému použití odjinud, než z dané třídy. Tohoto zabezpečení je dosaženo skrze definované rozhraní, které odstiňuje jeho uživatele (programátory) od znalosti zbytečných deailů. Tím je zajištěn tzv. kontrolovaný přístup.
„Síla zapouzdřeného kódu spočívá v tom, že každý ví, jak k němu přistupovat, atedy jak jej používat bez ohledu na konkrétní podrobnosti implementace – což ovšem znamená bez obav z neočekávaných vedlejších efektů.“ [24, strana 55]

2. **Dědičnost**
   Dědičnost umožňuje předávat v rámci hierarchické klasifikace vlastnosti jednoho objektu objektům níže postaveným pod tímto objektem. Klasifikace postupuje směrem shora dolů. Zjednodušuje se tím definice společných charakteristik (vlastností a chování) objektů. Tyto společné charakteristiky jsou definovány pouze jednou, v tzv. předkovi. Všechny jeho potomci tyto vlastnosti dědí a dále je možné každému z nich přidat vlastnosti, které jsou specifické pouze pro tento objekt, respektive pro tuto třídu. Nejen, že nedochází k duplikaci definic a složitost daného programu roste lineárně (narozdíl od procesního přístupu, kde roste geometrickou řadou), další výhodou je, že předem známe minimální charakteristiky, kterými bude potomek disponovat.

3. **Polymorfismus**
   Polymorfismus neboli mnohotvarost znamená použití definované obecné třídy akcí v rámci více tříd. Zmíněná třída akcí je definována v tzv. rozhraní. Rozhraní je předpisem akcí, které by měli být v rámci třídy implemenovány.

Multithreading

Další vlastností Javy je, že podporuje tzv. multithreading, nebo-li psaní vícevláknových aplikací. Za vícevláknovou aplikaci je považovaná aplikace, která se skládá ze dvou a více

částí, které mohou běžet současně. Jednotlivé části aplikace jsou nazývané jako vlákna (anglicky thread).[24]

Každá javová aplikace spouští alespoň jedno vlákno. Toto vlákno je vytvořeno a spuštěno automaticky hned po startu aplikace a nazývá se *hlavní vlákno*. Hlavní proto, že se jedná o první spuštěné vlákno, za běhu aplikace se od něj oddělují další vlákna a zároveň je posledním vláknem, jehož běh bude ukončen.

Vlákno je v programovacím jazyce Java vytvořeno, když je vytvořena instance objektu typu `Thread`. To je možné provést dvěma způsoby.

1. Vytvořením instance třídy implementující rozhraní `Runnable`. Tato třída musí implementovat metodu `run`. Ke spuštění vlákna dojde voláním metody `start`, která provede volání metody `run`.

2. Vytvořením instance třídy, která rozšiřuje třídu `Thread`. Tato třída musí překrývat metodu `run` a volat metodu `start` pro start nového vlákna.

3.3.2 Grafické uživatelské rozhraní

Tato kapitola rozebírá možnosti řešení grafického uživatelského rozhraní, zkráceně GUI[21], v programovacím jazyce Java. Existují dvě standardní známé knihovny pro tvorbu grafického uživatelského rozhraní. První z nich je Java AWT (Abstract Window Toolkit), druhá, novější Java Swing (JFC – Java Foundation Classes).

Java AWT

Jedná se o první javovou (standartní, heavyweight) knihovnu určenou pro tvorbu grafického rozhraní. O vykreslování grafických komponent se v případě jejího použití stará operační systém, komponenty jsou tedy přizpůsobeny danému operačnímu systému, a to jak chováním, tak jejich vzhledem.

[21] Anglicky **Graphical User Interface**

Java Swing

Tato grafická knihovna v podobě dědění přebírá některé vlastnosti knihovny awt, avšak o vykreslování komponent se již stará sama (lightweight). Výhodou tohoto přístupu je především přenositelnost. Daný program se na každém operačním systému chová a vypadá stejně, můžeme jeho chování předpokládat, dále se můžeme spolehnout, že funkce, které jsme použili, budou fungovat. Nevýhodou je potom vyšší režie na vykreslování grafického rozhraní, a tím zpomalení celé aplikace. Při dodržování základních pravidel je však toto zpomalení unesitelné a lze s danou aplikací bez problémů pracovat.

Ve Swingu existuje privilegované vlákno, Event Dispatching Thread, které se stará o obsluhu všech událostí. Jeho další vlastností pak je, že je to jediné vlákno, které smí modifikovat uživatelské rozhraní.

JavaFX

Existuje ještě jeden, už ne tolik známý, framework pro tvorbu okenních aplikací. Je jím JavaFX. Platforma určená pro tvorbu bohatých okenních a bohatých internetových aplikací, o níž Oracle oznámil, že do budoucna nahradí své předchůdce. Proto jsme se rozhodli aplikaci naprogramovat rovnou s jejím využitím a bude jí věnována celá následující kapitola.

3.3.3 Platforma JavaFX

Historie

JavaFX, původně projekt s označením F3[22], je dílo Chrise Olivera z dílny Sun Microsystems[23]. První verze byla oficiálně představena na konferenci JavaOne v roce 2007.[15]

Od verze 2.0 z roku 2011 je používána jako nativní knihovna Javy, a tím končí podpora JavaFX Scriptu, deklarativního skriptovacího jazyku určeného pro programování na platformě JavaFX. Aktuální verze jsou obsaženy v Java 8 a Java 9.

[22]Zkratka pro Form Follows Function.

[23]V roce 2010 koupený společností Oracle.

Vlastnosti

Platformu JavaFX lze využít pro vývoj jak desktopových aplikací (jako v našem případě), tak pro vývoj webových a mobilních aplikací. Jedná se o technologii založenou na programovacím jazyce Java, která je plně integrovaná s JRE[24]. Je jednoduchá na použití a obsahuje sadu nástrojů a technologií, které usnadní spolupráci vývojářů a designérů. Klíčovou vlastností je podpora různých platforem. Jinak řečeno mohou aplikace běžet na různých operačních systémech a zařízeních bez potřeby složitého přepisování kódu individuálně pro každou obrazovku.[15]

Princip

Existují dva způsoby, jak vyvíjet aplikace na platformě JavaFX . První z nich funguje na stejném principu, jako je tomu ve výše zmiňovaném Swingu. Tedy vytvářením instancí jednotlivých formulářových prvků, jež jsou následně vkládány do tzv. layoutů. Druhým způsobem je použití jazyka FXML, jazyka pro tvorbu formulářů. Jak již název napovídá, jedná se o jazyk odvozený z XML.[6]

Ať zvolíme první či druhý způsob, vstupním bodem pro všechny JavaFX aplikace je třída `Main`, která rozšiřuje abstraktní třídu[25] `javafx.application.Application`. Metodu `start` této třídy je nutné překrýt[26]. Metoda `start` je volána poté, co je systém připraven k běhu aplikace. Přijímá parametr typu `Stage`. `Stage` představuje v JavaFX kontejner nejvyšší úrovně.[18]

Vizuální prvky uživatelského rozhraní reprezentuje tzv. Scene Graph. Je to hierarchický strom uzlů s jedním kořenovým uzlem (anglicky root node). Kořenový uzel je první uzel grafu, který nemá žádného rodiče. Každý jednotlivý prvek uživatelského rozhraní představuje v tomto grafu prvek nazývaný uzel (anglicky node). Uzly existují

[24]Zkratka pro Java Runtime Environment - https://techterms.com/definition/jre

[25]Abstraktní třídou se nazývá třída, která obsahuje deklarace abstraktních, nebo-li neimplementovaných metod.[20]

[26]Překrýt, jinými slovy definovat novou verzi dané metody rodičovské třídy. Překrývají se metody, které třída zdědila, ale byly pro ní nevyhovující.[20]

dvou typů. První typ, nazývaný branch node, může obsahovat další uzly. Druhý typ, nazývaný leaf node, neobsahuje žádné další uzly.[19]

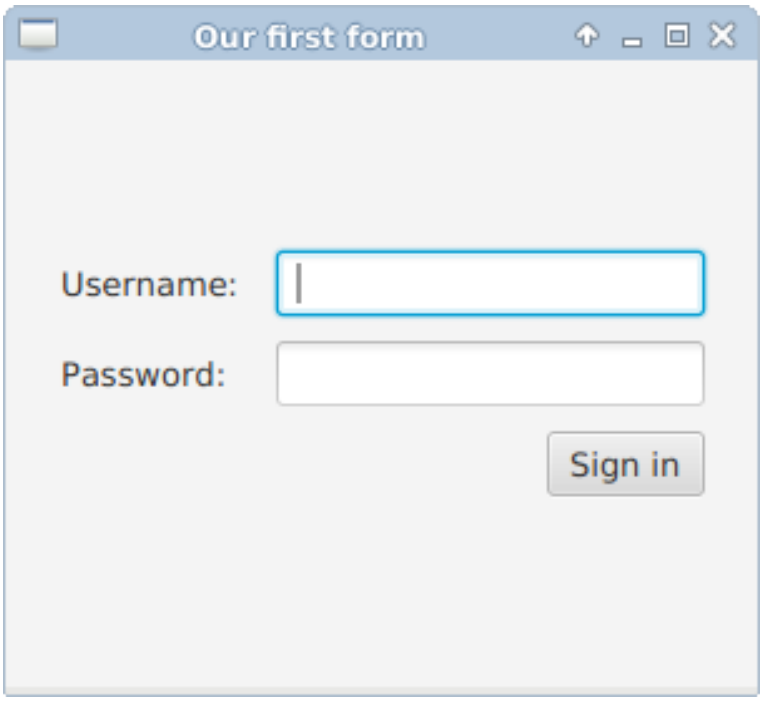
Pro lepší pochopení je na následující ukázce zdrojového kódu zachycena základní struktura JavaFX aplikace.

```
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        final GridPane root = new GridPane();
        final TextField username = new TextField();
        final PasswordField password = new PasswordField();
        final Label usernameLabel = new Label("Username: ");
        final Label passwordLabel = new Label("Password: ");
        final HBox buttonBox = new HBox();
        buttonBox.setAlignment(Pos.CENTER_RIGHT);
        buttonBox.getChildren().add(new Button("Sign in"));
        GridPane.setRowIndex(usernameLabel, 0);
        GridPane.setColumnIndex(usernameLabel, 0);
        GridPane.setRowIndex(username, 0);
        GridPane.setColumnIndex(username, 1);
        GridPane.setRowIndex(passwordLabel, 1);
        GridPane.setColumnIndex(passwordLabel, 0);
        GridPane.setRowIndex(password, 1);
        GridPane.setColumnIndex(password, 1);
        GridPane.setRowIndex(buttonBox, 2);
        GridPane.setColumnIndex(buttonBox, 0);
        GridPane.setColumnSpan(buttonBox, 2);

        root.setAlignment(Pos.CENTER);
        root.setHgap(10);
        root.setVgap(10);
        root.getChildren().addAll(usernameLabel, username,
            passwordLabel, password, buttonBox);
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("Our first form");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Ttřída `Main` z ukázky obsahuje metodu `main` a metodu `start`. Metoda `main` umožňuje spouštět JAR soubory bez JavaFX Launcheru. V případě, že je JAR soubor aplikace vytvořen pomocí nástroje JavaFX Package Tool[27], není tato metoda vyžadovaná.

Metoda `start` přijímá jako jediný parametr GUI kontejner typu `Stage`. Je v ní vytvořen kořenový uzel, v tomto případě se jedná o `GridPane` umožňující přidávat jednotlivé prvky na základě mřížkového layoutu (tedy čísel řádků a sloupců). Do `GridPane` jsou následně přidány potomci reprezentující jednotlivá pole formuláře a poté je předán konstruktoru objektu typu `Scene` [28] s požadovanými rozměry.

Výstupem předchozí ukázky je dialog s jednoduchým přihlašovacím formulářem, jak je patrné z obrázku 3.5.



Obrázek 3.5: Výstup ukázkového kódu - přihlašovací formulář

Následuje ukázka vytvoření toho samého formuláře za použití jazyka FXML. Pomocí

[27]Vkládá do JARu JavaFX Launcher - link

[28]Základní kontejner pro všechen obsah ve scene graph.

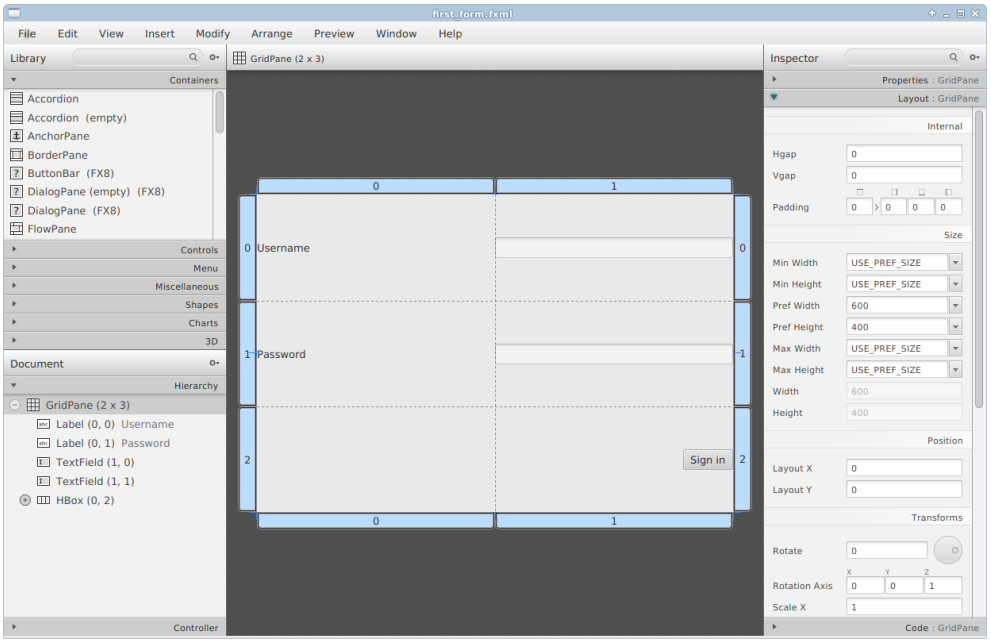
aplikace JavaFX Scene Builder[29], jejíž podoba je zobrazena na obrázku 3.6, vytvoříme FXML šablonu přihlašovacího formuláře, a tu následně načteme v metodě `start`.

```
    @Override
    public void start(Stage primaryStage) throws Exception {
        GridPane root = FXMLLoader.load(Main.class.getClassLoader()
            .getResource("templates/form.fxml"));
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("Our first form");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.RowConstraints?>

<GridPane alignment="CENTER" hgap="10.0" maxHeight="-Infinity"
    maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
    vgap="10.0" xmlns="http://javafx.com/javafx/8.0.65" xmlns:fx="
    http://javafx.com/fxml/1">
     <children>
         <Label text="Username" GridPane.columnIndex="0" GridPane.
             rowIndex="0" />
         <Label text="Password" GridPane.columnIndex="0" GridPane.
             rowIndex="1" />
         <TextField GridPane.columnIndex="1" GridPane.rowIndex="0" /
             >
         <TextField GridPane.columnIndex="1" GridPane.rowIndex="1" /
             >
         <HBox alignment="CENTER_RIGHT" GridPane.columnSpan="2"
             GridPane.rowIndex="2" GridPane.valignment="CENTER">
              <children>
                  <Button mnemonicParsing="false" text="Sign in" />
              </children>
```

[29]Grafický návrhář, který je dostupný ke stažení na adrese `http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html`

```
            </HBox>
        </children>
        <columnConstraints>
            <ColumnConstraints />
            <ColumnConstraints />
        </columnConstraints>
        <rowConstraints>
            <RowConstraints />
            <RowConstraints />
            <RowConstraints />
        </rowConstraints>
        <padding>
            <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
        </padding>
</GridPane>
```



Obrázek 3.6: JavaFX Scene Builder

Balíček javafx.concurrent

Spuštěním JavaFX aplikace dojde zároveň ke spuštění hlavního vlákna uživatelského rozhraní, vlákna nazývaného *JavaFX Application Thread*. Toto vlákno zpracovává

všechny události týkající se uživatelského rozhraní. Není vhodné v něm provádět dlouhotrvající volání. Taková volání mohou způsobit tzv. zamrznutí aplikace. Proto pro JavaFX aplikace existuje podpora takovýchto volání v podobě balíčku `javafx.concurrent`.

Balíček `javafx.concurrent` obsahuje rozhraní `Worker` a tři abstraktní třídy `Task`, `Service` a `ScheduledService`, které toto rozhraní implementují.[29]

Balíček `javafx.concurrent` umožňuje volání složitějších, déle trvajících operací na pozadí. Pro krátké, jednoduché operace poskytuje třída `Platform` metodu `runLater`, která způsobí běh `Runnable` na JavaFX Application Thread.

AquaFX

Jedním z požadavků, kladených na výsledný program, je přizpůsobení designu prostředí operačního systému pro Mac OS X. Tento požadavek bude jednoduše vyřešen využitím jedním z dostupných JavaFX Themes. Je jím AquaFX. Umožní nastylovat celou aplikaci bez složitého ohýbání CSS.

3.3.4 Balíček java.net

Důvodů, proč zvolit Javu jakožto programovací jazyk pro realizaci síťové komunikace je hned několik. Jak už bylo zmíněno výše, umožňuje programátorům generovat bezpečný, platformně nezávislý kód. Také poskytuje podporu síťové komunikace prostřednictvím balíčku `java.net`.[24]

Princip síťové komunikace, a s ním související protokoly, byly popsány hned v úvodu teoretické části této práce. Tato kapitola bude věnována prostředkům, které balíček `java.net` obsahuje pro jejich podporu. Balíček `java.net` poskytuje třídy a rozhraní pro tvorbu síťových aplikací. Třídy, které obsahuje, můžeme rozdělit do dvou skupin podle způsobu, jakým umožňují přístup k datům a zdrojům v rámci sítě.

První skupina se nazývá *Low Level API* a poskytuje podporu pro práci se sokety, síťovými rozhraními a adresami. Patří sem třídy `InetAddress`, `Socket`, `ServerSocket`, `DatagramSocket`, `MulticastSocket` a `NetworkInterface`.

V rámci tzv. *High Level API* poskytuje třídy podporující přístup k datům prostřednictvím URL[30]. Zahrnuje třídy `URI`, `URL`, `URLConnection` a `HttpURLConnection`.[17]

Podpora HTTP

Http komunikace je v Javě podporovaná prostřednictvím třídy `HttpURLConnection`. Třída `HttpURLConnection` je *„podtřída třídy URLConnection zajišťující podporu pro připojení prostřednictvím protokolu HTTP.“*[24, strana 752].

Její instanci lze vytvořit zavoláním metody `openConnection` na objekt typu URL, jak je ukázáno v ukázce na straně 31.

Pro vytvoření instance typu URL je použit konstruktor přijímající čtyři parametry: `String nazevProtokolu`, `String hostname`, `int cisloPortu`, `String cesta`.

Následně je vytvořeno spojení a jsou vypsány informace jako je použitá metoda požadavku, kód odpovědi a zpráva odpovědi. Pod ukázkou zdrojového kódu je zobrazen výstup ukázkového programu.

```
class HttpConnectionSample {
    public static void main(String args[]) throws Exception {
        URL url = new URL("http", "docs.oracle.com", 80, "/javase
            /8/docs/api/java/net/HttpURLConnection.html");
        HttpURLConnection httpURLConnection = (HttpURLConnection)
            url.openConnection();
        System.out.println("Request method: " + httpURLConnection.
            getRequestMethod());
        System.out.println("Response code: " + httpURLConnection.
            getResponseCode());
        System.out.println("Response message: " + httpURLConnection
            .getResponseMessage());
    }
}
```

```
Request method: GET
Response code: 200
Response message: OK

Process finished with exit code 0
```

[30]URL - Uniform Resource Locator

Podpora UDP

„Java implementuje datagramy nad protokolem UDP pomocí dvou tříd: instance typu DatagramPacket je kontejnerem pro data, zatímco instance typu DatagramSocket je mechanizmus používaný k odesílání či přijímání dat (instancí typu DatagramPacket).“[24, strana 755]

Třída `DatagramSocket` slouží jak pro odesílání paketů, tak k jejich přijímání. Pro odesílání paketů slouží metoda `send`. Pro přijímání paketů slouží metoda `receive`. Obě metody přijímají jako parametr objekt typu `DatagramPacket`.

Třída `DatagramPacket` představuje UDP paket a poskytuje metody pro přístup k paketu a jeho datům. Jedná se například o metodu, která vrací adresu zdroje (v případě přijatého paketu) nebo metodu, která umožňuje nastavit adresu, na kterou má být paket odeslán. Dále je možné získat data uložená v datagramu (v případě odesílání je do něj načíst), zjistit délku dat v paketu, nastavit či získat číslo portu, apod.

Vytvoření soketu je možné jedním ze čtyřech poskytovaných konstruktorů. Vznikne-li při vytváření soketu chyba, mohou všechny z nich vyvolat výjimku `SocketException`. Přehled konstruktorů obsahuje tabulka 3.3.

Konstuktor	**Popis**
DatagramSocket()	Vytvoří instanci typu DatagramSocket a váže ji s libovolným portem na lokálním počítači.
DatagramSocket(int port)	Vytvoří instanci typu DatagramSocket a váže ji se specifickým portem na lokálním počítači.
DatagramSocket(int port, InetAddress laddr)	Vytvoří instanci typu DatagramSocket a váže ji se specifickou adresou.
DatagramSocket(SocketAddress bindaddr)	Vytvoří instanci typu DatagramSocket a váže ji se specifickou soketovou adresou.

Tabulka 3.3: Přehled konstruktorů třídy DatagramSocket[24][16]

3.3.5 Řízení překladu

Jednou z fází vývoje aplikace je její setavení, zjednodušeně řečeno převod zdrojových kódů na spustitelný soubor. Tato sekce se zabývá třemi nejznámějšími nástroji, jejichž cílem je usnadnění tohoto procesu.

Utilita make

„make je nástroj pro překlad a sestavení programu, který se používá na systémech Unix a při vývoji C/C++.“[5, strana 42] Představuje nadstavbu příkazového interpreteru a není přenositelný mezi platformami, protože využívá platformně závislé příkazy operačního systému.[10]

Ant

Ant, zkratka pro Another Neat Tool, je označení pro platformně nezávislý nástroj pro sestavování aplikací.[11]

Na rozdíl od utility `make` nevyužívá nativní příkazy operačního systému, ale poskytuje tzv. úlohy, což jsou nejpoužívanější příkazy operačních systémů naimplementované v jazyce Java. Programátor tak není při sestavování konfiguračního souboru limitován jedním typem operačního systému.

build.xml

Sestavování aplikace je řízené xml souborem, obvykle pojmenovaným build.xml. Kořenovým elementem souboru build.xml je element *project*, který obsahuje jeden povinný atribut *default*. Hodnota atributu default určuje jméno implicitního cíle (anglicky target), který bude vykonán v případě spuštění aplikace Ant bez parametrů. Cíle jsou v souboru definovány pomocí elementu *target* s povinným atributem *name*, který slouží pro jejich identifikaci.

Pro účely rozšíření množiny elementů o úlohy, které nejsou standardní součástí aplikace Ant, je určen element *taskdef*. V tomto elementu je nutné specifikovat jméno

nového elementu a jméno kvalifikované třídy obstarávající implementaci dané úlohy, o kterou bude množina elementů rozšířena.[12]

Java Application Bundler

Příkladem takového rozšíření může být nástroj Java Application Bundler dostupný ke stažení ná stránkách projektu java.net: `https://java.net/projects/appbundler/downloads`. Instalace tohoto nástroje spočívá ve vytvoření adresáře `lib` na nejvyšší úrovni projektu a v přidání staženého souboru appbundler-1.0.jar do tohoto adresáře. Poté je nutné rozšířit množinu elementů o úlohu bundleapp prostřednictívm elementu taskdef následovně:

```
<taskdef name="bundleapp"
  classname="com.oracle.appbundler.AppBundlerTask"
  classpath="lib/appbundler-1.0.jar" />
```

Nově definovaný element se používá stejně, jako standardní elemety obsažené v aplikaci Ant. Příklad použití je uveden na následující ukázce:

```
<target name="bundle-sample">
  <bundleapp outputdirectory="dist"
      name="SampleBundler"
      displayname="Sample - Bundler"
      identifier="cz.sample.SampleBundler"
      mainclassname="cz.sample.SampleBundler">
      <classpath file="dist/SampleBundler.jar" />
  </bundleapp>
</target>
```

Maven

Maven je novější nástroj pro sestavování aplikací, než je výše popisovaný nástroj Ant. Podnětem k jeho vzniku byly práce na projektu Jakarta Turbine, který se skládal

z několika projektů a při používání sestavovacích souborů aplikace Ant docházelo k duplikaci konfigurací a jejich nepřehlednosti.[12]

Proces sestavování aplikace je řízen xml souborem pojmenovaným pom.xml[31]. Tento soubor obsahuje základní informace o projektu, konfigurační detaily a závislosti projektu na externích knihovnách.

3.3.6 Java IDE

IDE (Integrated Development Environment), zkratka pro vývojové prostředí. Vývojové prostředí je aplikace, která slouží pro usnadnění vývoje software. Poskytuje funkce jako klasický textový editor obohacený o podporu programování,refaktoringu, kontroly syntaxe, ladících nástrojů, apod. V současné době se jednotlivá vývojová prostředí liší především kvalitou dokumentace, množstvím nabízených pluginů (které umožňují vývojové prostředí rozšířit například o podporu jazyků běžně používaných s javovými aplikacemi) a snadností ovládání.[2]

Tři nejznámější vývojová prostředí s podporou vývoje Java aplikací jsou:

1. Eclipse – open source vývojové prostředí se zaměřením na možnost rozšíření pomocí pluginů.

2. NetBeans – open source vývojové prostředí od firmy Oracle. Oproti ostatním nabízí možnost stáhnout verzi předkonfigurovanou přesně pro potřeby programátora bez zbytečných doplňků navíc.

3. JetBrains IntelliJ IDEA – komerční produkt dostupný jak v open source verzi Community Edition, tak v placené verzi Ultimate Editon. Nabízí integraci pro Ant a Maven a uživatelé na ní ocení podporu na detekci špatného kódu (například hledání duplicit, apod.)

Autorka této práce měla možnost vyzkoušet všechna tato tři vývojová prostředí. Na základě zkušeností dospěla k názoru, že, alespoň co se Javy týká, je nabídka jednotlivých

[31]POM - Project Object Model

produktů v současné době tak vyrovnaná, že záleží hodně na osobních preferencích toho kterého vývojáře.

Projekt bude realizován JetBrains IntelliJ IDEA, verzi Ultimate Edition.

3.4 Systém pro správu verzí

Při vývoji každé aplikace je užitečné používat nějaký ze systémů pro správu verzí zdrojového kódu, a to z následujících důvodů:

- Vždy se může stát nějaká nepředvídaná událost, která způsobí ztrátu či poškození dat na lokálním počítači, repozitář potom v tomto případě funguje jako jistá forma zálohy.
- Programátor může udělat chybu a potřebuje se vrátit ve svém řešení zpět.
- Jedná-li se o týmový projekt, upravuje kód více programátorů, a je nutné mít pod kontrolou jejich úpravy, aby si například kód vzájemně nepřepisovali,
- navíc jim umožňuje vyvíjet aplikaci v tzv. větvích.
- Pomocí tohoto nástroje lze sdílet zdrojové kódy.
- Dává nám možnost porovnání verzí projektu, čímž získáváme přehled o průběhu vývoje.

3.4.1 Centralizované systémy

Systémy pro správu verzí ukládají změny provedené v projektu do úložišť nazývaných repository, neboli repozitáře. Do těchto úložišť jsou data pouze přidávána, nikoliv přepisována, proto se lze případně kdykoliv vrátit k jakékoliv verzi projektu. Je-li systém založen na ukládání dat na jediný server a pro využití je nutná komunikace s tímto serverem, protože na straně klienta je známa pouze nejnovější verze, jedná se

o centralizovaný systém. Zastupují ho například CSV či novější Subversion, zkráceně SVN.

3.4.2 Distribuované systémy

Dalším typem verzovacích systémů je tzv. distribuovaný systém pro správu verzí, kde je celá historie změn ukládána jak na společný server, tak na počítačích vývojářů. Hlavními výhodami jsou tedy možnost pracovat offline, dále je kód zálohovaný na více místech, čili v případě ztráty dat na jednom úložišti nejsou data ztracena. Mezi tyto systémy patří například Git a Mercurial.

Git

Pro projekt byl jako verzovací systém vybrát Git. Repozitář a změny v něm lze prohlížet přes webové rozhraní a pohodlně je s ním možné pracovat přes příkazovou řádku. Mezi nejpoužívanější příkazy patří clone, status, commit, pull, push a checkout.

Stažení projektu Nejprve je potřeba získat existující projekt k sobě, na lokální počítač. To se provede tzv. klonováním repozitáře, pro které slouží příkaz git clone [url]. Tento příkaz zařídí vytvoření kopie všech dat ze serveru. Budeme mít tedy i k dispozici informace o jednotlivých historických verzích zdrojového kódu. Výhodou je, že každý zúčastněný má informace o všem. Pokud by došlo k nějaké havárii a data na serveru byla poškozena nebo úplně ztracena, je možné provést obnovu z nějakého lokálního zdroje, kde existuje kopie dat.

Nahrávání změn Nyní už máme k dispozici gitový repozitář, do kterého můžeme přispívat a pracovní kopii souborů projektu (*checkout*). Každý z těchto souborů může být buď ve stavu **tracked** (sledován) nebo **untracked** (nesledován).

Se soubory tracked se již pracovalo a jsou součástí git repozitáře. Mohou nabývat následujících stavů:

1. modified (změněn)
2. unmodified (nezměněn)
3. staged (připraven k zápisu)

Stavy souborů můžeme kontrolovat pomocí příkazu *git status*.

Abychom mohli provedené změny v souborech nahrát, musí být ve stavu **staged**. Poté můžeme použít příkaz commit, který provede zápis změn. Příkaz commit přijímá parametr -m, za který můžeme připojit zprávu rovnou z příkazové řádky.

```
$ git commit -m \Popis změn"
```

Oprava změn V případě, že jsme provedli commit a po něm jsme ještě provedli změny s ním související, je možné pomocí příkazu git commit –amend připojit změny provedené později k původnímu commitu a připojit novou zprávu či upravit tu původní.

Odeslání změn na server Po provedení commitu jsou provedené změny pouze na lokálním počítači v pracovní verzi souborů gitového repozitáře. Aby se změny projevily i na serveru, je nutné je tam odeslat příkazem git push.

3.5 Průběžná integrace

Průběžná integrace neboli Continuous Integration slouží k urychlení vývoje software. Jedná se o souhrn nástrojů a definovaných metod, které mají vést ze zlepšení spolupráce a urychlení jak vývoje, tak nacházení chyb.

Mnoho chyb ve zdrojových kódech a časová náročnost jejich nalezení byly jedním z hlavním podnětů ke vzniku průběžné integrace. Dále to byly například:

- chaos v rámci verzování jednotlivých buildů
- složitý proces vytváření buildů a jejich údržby
- nepřehlednost v repozitářích se zdrojovými kódy

Průběžná proto, že každý, kdo se podílí na vývoji by měl přispívat průběžně, v ideálním případě každý den. Vše výše zmíněné poskytují tzv. integrační servery, které dodržují základní principy CI:

- sdílený repozitář zdrojových kódů
- automatické vydávání buildů
- automatické testy
- kontrola kvality kódu
- přístupná nejaktuálnější verze

Jedním z takových integrační serverů je i TeamCity, který bude používán v rámci tohoto projektu.

3.5.1 JetBrains TeamCity

TeamCity je server určený pro průběžnou integraci od společnosti JetBrains [32].

Jeho tvůrci si zakládají na uživatelsky přívětivém (anglicky user-friendly) prostředí a snadném nastavení. Jedná se o webovou aplikaci, v níž je možné vytvářet a testovat verze aplikací. Dále je možné sledovat různé statistiky a přehledy spojené s testy či vytvářením verzí.[28]

Probíhající proces vytváření verze aplikace se nazývá *build*. Stejný výraz se používá pro již vytvořenou verzi.

O vytváření jednotlivých verzí se stará tzv. build system. Toto slovní spojení lze přeložit jako sestavovací systém, v praxi se však výraz sestavovací moc nepoužívá. Čeští programátoři si tento výraz často přizpůsobují[33] na buildovací systém. Build system se skládá z TeamCity serveru a build agentů. [27]

[32]Společnost zabývající se vývojem programového vybavení, založená roku 2000. Oficiální stránky dostupné na: `https://www.jetbrains.com`

[33]Pro potřeby této práce se budeme držet anglického slovního spojení build system.

Build agent je část software, která vykonává build proces. Je nainstalovaná a nakonfigurovaná nezávisle na TeamCity serveru. To znamená, že může být nainstalovaná na úplně jiném počítači, než běží TeamCity server[34] a může spouštět jiný operační systém, než server[35]. [26]

TeamCity Server se stará o monitoring všech build agentů, rozdělování build procesů (respektive build front) mezi ně a reportování výsledků. [27]

Do aplikace mají přístup tři typy uživatelů. Výše zmínění build agenti, dále administrátoři (mají na starosti například přidělování práv) a v neposlední řadě také vývojáři.

Pro potřeby našeho projektu budeme do TeamCity přistupovat jako vývojáři. Je důležité vědět, že v TeamCity lze vytvářet projekty a ke každému z nich lze vytvořit konfiguraci podle níž bude proveden build. Více o tomto procesu uvedeme v praktické části této práce, kde si ukážeme konkrétní konfiguraci.

3.6 Aplikace pro Mac OS

Aplikace pro Mac OS jsou obvykle šířeny buď jako `.app`[36] nebo jako `.dmg` soubor. Pokud stáhneme aplikaci jako `.app` soubor, provedeme instalaci zkopírováním souboru na místo, odkud chceme aplikaci spouštět a spustíme jí kliknutím na spustitelný soubor `.app`[37]. Příkladem aplikace, kterou lze stáhnout jako soubor `.app` je hudební aplikace Spotify.

Soubor `.dmg` obvykle obsahuje instalační balíček nebo výše popsaný spustitelný soubor `.app`. Jeho obsah lze zobrazit kliknutím na soubor `.dmg` v aplikaci `Finder`. Po dokončení instalace jeho obsahu je doporučováno soubor `.dmg` smazat. Příkladem aplikace, kterou lze stáhnout jako soubor `.dmg` je vývojové prostředí IntelliJ IDEA.

[34]Tento postup je doporučován z hlediska výkonu serveru, na který je kladen velký důraz.

[35]Podpora paralelních buildů, které běží zároveň na různých platformách a v různých prostředích.

[36]APP - Application Bundle

[37]Jedná se o velmi podobný princip jako se spustitelnými soubory .exe.

Kapitola 4

Praktická část

Na základě poznatků získaných v teoretické části této práce byla vytvořena aplikace. Jejímu popisu a popisu nejdůležitějších aspektů procesu vytváření je věnována tato kapitola. Kapitola je rozdělena do čtyř částí.

První z nich je část s názvem *Návrh*. Obsahuje popis změn, tvorbu prototypu aplikace, následnou změnu workflow [1] a návrh uživatelského rozhraní.

Jako druhá následuje *Realizace*. V této části jsou popsány praktické aspekty od nastavení IDE, přes vytvoření projektu, až po popis nejzajímavějších částí implementace včetně ukázek zdrojových kódů.

Závěrem této kapitoly je popsán systém vydávání verzí aplikace prostřednictvím serveru TeamCity.

4.1 Návrh

Jedním z dílčích cílů této diplomové práce je zjednodušit v rámci návrhu stávající aplikaci. Prostřednictvím zjednodušení dojde ke zlepšení uživatelského rozhraní. Zjednodušení bude dosaženo odstraněním známých problémů.

[1] Pro účely této práce výraz představuje synonymum pro ppis procesů probíhajících v souvislosti s používáním aplikace.

- Výběr projektů v levé části hlavního okna aplikace umožňuje uživateli zvolit tisk bez vybraného projektu. Je-li výběr projektu povinný, a uživatel přesto zvolí tisk, aniž by nejprve vybral projekt, bude zobrazeno okno s upozorněním, že tisk není možné provést dokud nebude vybrán projekt. Uživatel je nucen v tomto případě nejprve potvrdit upozorňující dialog, následně vybrat projekt a teprve poté opět zvolit tisk vybraných úloh. Protože uživatel dopředu neví, že je nutné vybrat projekt, může akce tisk vyžadovat až pět kliknutí myši (výběr tiskových úloh, volba tisk, potvrzení dialogu, výběr projektu, volba tisk). Tento proces je velice zdlouhavý, a proto je nutné jej optimalizovat.

- V případě, kdy je vyžadována autentizace uživatele a zároveň výběr projektu, je uživatel nejprve vyzván k výběru projektu a teprve poté k zadání přihlašovacích údajů. Pokud není autentizace úspěšná, postrádá výběr projektu smysl, proto bude pořadí těchto dvou akcí změněno.

- Do hlavního okna aplikace bude přidán menubar a akce, které přímo nesouvisejí s tiskem, budou přesunuty do jeho nabídky. Jedná se o akci pro otevření webového rozhraní serveru a o informaci týkající se verze aplikace. Verze aplikace bude obsažena v About dialogu.

- V novém návrhu musí být zřejmé jaké akce přímo souvisejí se seznamem tiskových úloh. Jedná se o tisk, mazání, refresh.

- Do nastavení bude nově přístup nejen z nabídky dostupné pravým kliknutím myši na ikonku (tray icon) aplikace, ale i z nabídky (v menubaru) hlavního okna.

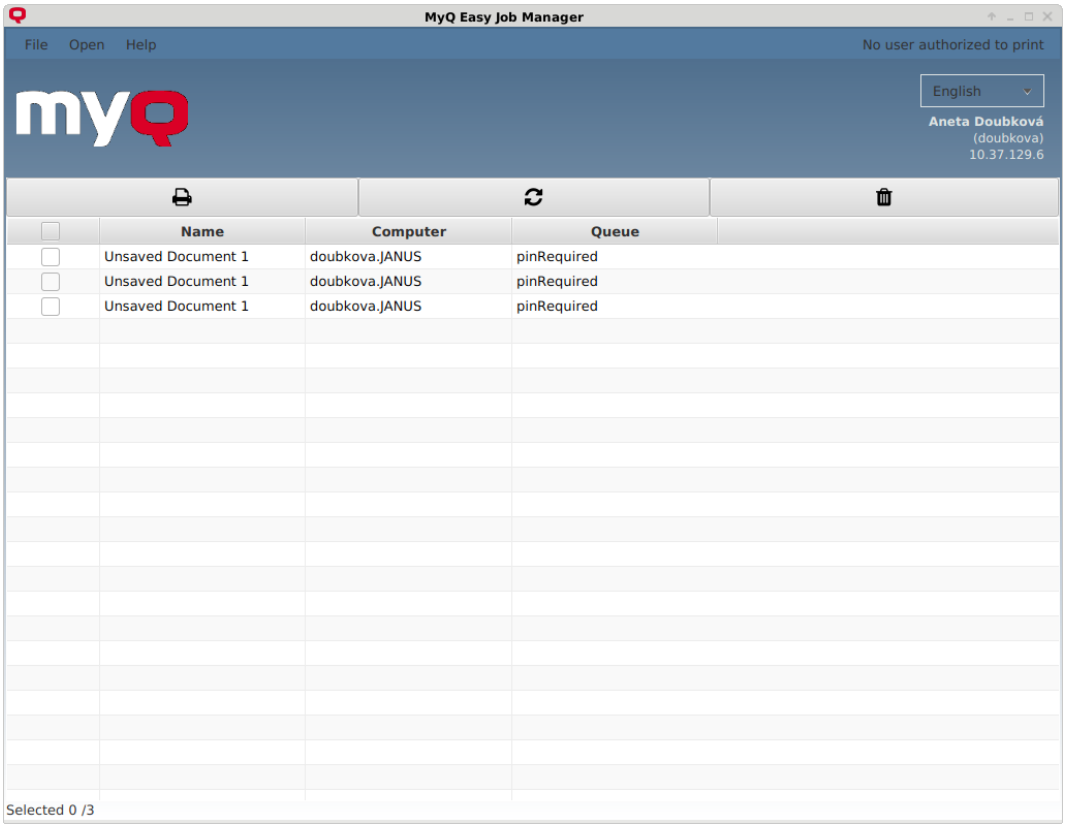
- Protože je počítáno s modelem, kdy (na jednom počítači) pod jedním uživatelským účtem tiskne více uživatelů, bude pro přístup k nastavení vyžadovaná autentizace uživatele s oprávněním[2]. Tím bude zajištěno, že nastavení bude měnit pouze oprávněná osoba.

[2]Uživatelem s oprávněním je myšlena obdoba uživatele Administrátor v OS Windows, tedy sudo uživatel a root pro Linux a Mac.

- Volba Remember (Zapamatuj) bude rozšířena o možnost výběru jak dlouho se mají zadané údaje pamatovat. Uživatel bude po tuto dobu považován za přihlášeného a nebude od něj při tisku vyžadováno zadání autorizačních údajů. Uživatel bude mít možnost odhlásit se. Odhlášením budou zapamatované údaje „zapomenuty“.

4.1.1 Prototyp

Nejprve vznikla ukázková aplikace, jejímž cílem bylo ověřit, že navržené uživatelské rozhraní a workflow jsou vyhovující. Tato aplikace, dále označovaná jako prototyp, neobsahovala reálnou funkčnost, posloužila především pro znázornění workflow a ověření uživatelského rozhraní.



Obrázek 4.1: Prototyp

Na obrázku 4.1 je snímek obrazovky s prototypem nové aplikace. Prototyp byl vytvořen pomocí nástroje Scene Builder[3]. Konktrétnímu postupu se věnuje sekce 4.2 na straně 50.

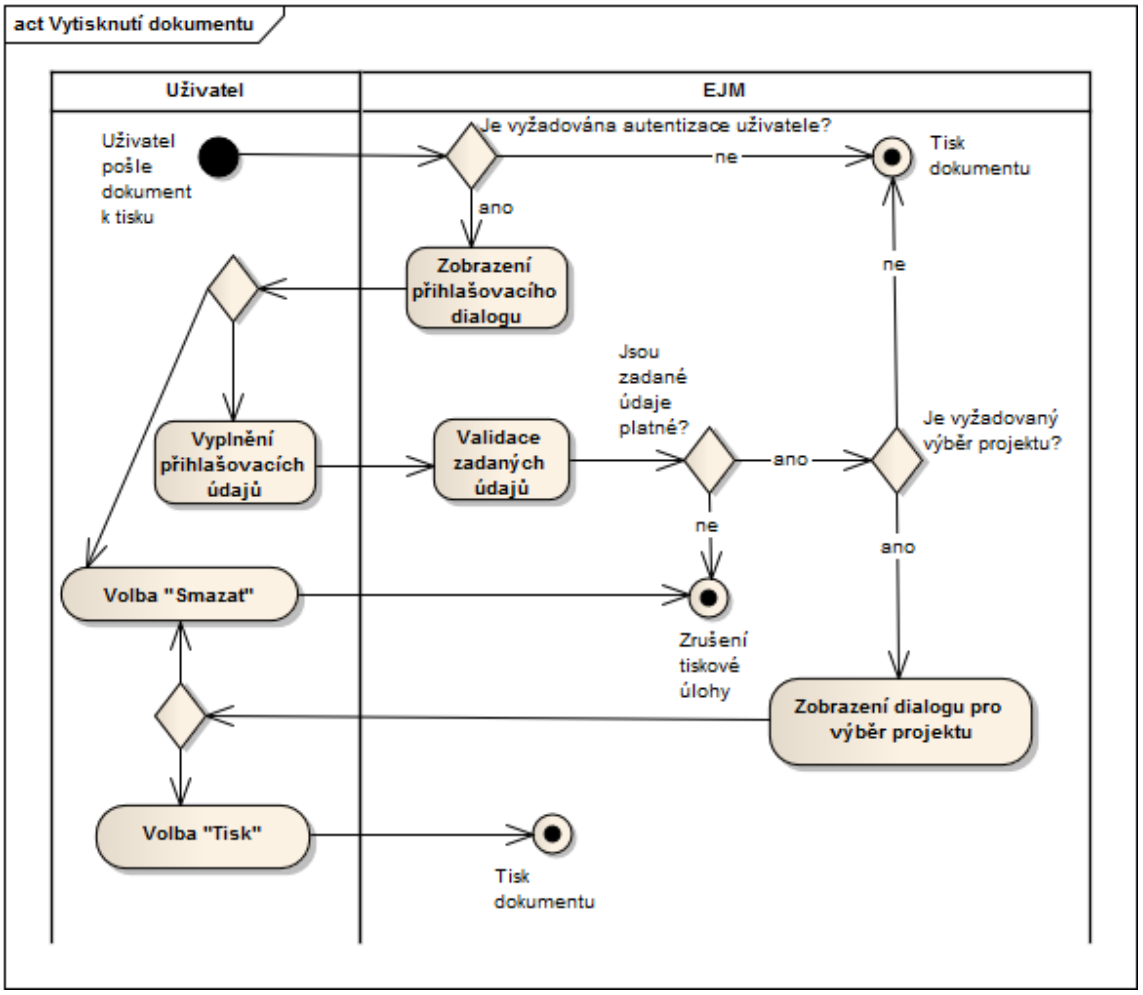
Na základě testování prototypu bylo navrženo úplné odstranění seznamu tiskových úloh. Důvodem bylo zjednodušení oproti původnímu workflow. Zjednodušení spočívá odstraněním nutnosti volby tisku v aplikaci. Jakmile uživatel jednou pošle úlohu k tisku, bude tato akce bezodkladně provedena. Interakci bude aplikace vyžadovat pouze v případě nutnosti výběru projektu nebo autentizace uživatele. Během této interakce bude mít uživatel možnost zrušit tisk výběrem volby delete. Podrobněji bude tato problematika rozebrána v sekci 4.1.2. Se změnou workflow úzce souvisí i změna uživatelského rozhraní, které se věnuje sekce 4.1.3.

4.1.2 Změna workflow

Nové workflow tisku je znázorněno na obrázku 4.2 diagramem aktivit vytvořeném v programu Enterprise Architect[4].

Diagramy aktivit jsou vývojové diagramy, s jejichž pomocí lze modelovat procesy jako aktivitu. Aktivita se skládá z kolekce uzlů, které jsou spojeny hranami. Existují tři typy uzlů: akční, řídící a objektové. Akční uzly znázorňují samostatné jednotky, které nelze v rámci aktivity dále dělit. Řídící uzly slouží k řízení cesty uvnitř aktivity. Objektové uzly představují objekty použité v rámci dané aktivity.[3]

Tento typ diagramů se mimo jiné používá k modelování obchodních procesů a pracovních postupů, proto je vhodný ke znázornění nového workflow aplikace.



Obrázek 4.2: Print workflow

Diagram aktivit

Pro oddělení akcí, které nepatří do stejné oblasti působnosti slouží grafický element swimlanes, označovaný také jako partition. Prvek je modelován svislými nebo vodorovnými čárami.[25] Čáry dělí diagram do jednotlivých oddílů aktivit, v našem případě se jedná o oddíly *Uživatel* a *EJM*. Názvy jednotlivých oddílů jsou samopopisující a je z nich zřejmé, do jaké oblasti působnosti patří. Z diagramu na obrázku 4.2 lze vyčíst,

[3]Nástroj je blíže popsán v teoretické části této práce.

[4]Nástroj od společnosti Sparx Systems, dostupný na https://www.sparxsystems.eu/enterprisearchitect

že probíhá interakce mezi uživatelem a aplikací.

Z diagramu na obrázku je dále patrné, že tisk může skončit dvěma stavy. Tiskem dokumentu, je-li proces úspěšný nebo zrušením tiskové úlohy v případě neúspěšné autorizace uživatele či v případě, že se uživatel rozhodne danou tiskovou úlohu sám smazat.

Podnětem k vyvolání procesu je, že uživatel pošle dokument k tisku. Nejprve dojde k ověření, zda tato akce vyžaduje autentizaci uživatele či nikoliv. V případě, že ne, je dokument rovnou odeslán k tisku. V opačném případě je uživatel formou dialogu požádán o zadání přihlašovacích údajů.

V tomto okamžiku má uživatel na výběr. Může vybrat volbu "Smazat", a tím ukončit proces zrušením tiskové úlohy. Nebo vyplní přihlašovací údaje, které jsou následně ověřeny.

Jsou-li zadány neplatné přihlašovací údaje, proces je opět u konce zrušením tiskové úlohy. Pokud jsou zadané přihlašovací údaje platné, proces pokračuje ověřením, zda daná tisková úloha vyžaduje výběr projektu.

Pokud ne, skončí proces tiskem dokumentu. Pokud ano, je uživatel požádán formou dialogu o výběr projektu. V tomto okamžiku má uživatel na výběr. Může vybrat volbu Smazat, a tím ukončit proces zrušením tiskové úlohy nebo vybere projekt, a tím ukončit proces tiskem dokumentu.

Na základě výše popsaného workflow bylo navrženo uživatelské rozhraní. Jeho popisu se věnuje další sekce.

4.1.3 Uživatelské rozhraní

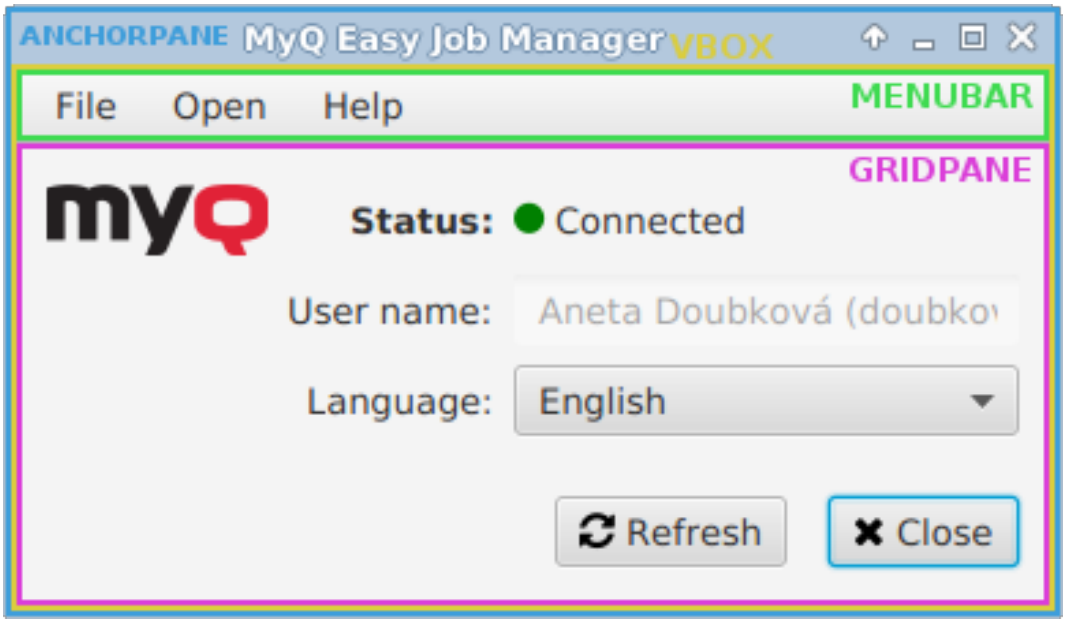
Uživatelské rozhraní bylo navrženo jako hlavní okno aplikace doplněné o další dialogová okna, která budou zobrazována v případě potřeby (např. pro přihlášení uživatele, výběr projektu či pro zobrazení chybové hlášky).

Hlavní okno je tvořeno logem, formulářem, který je rozdělen do dvou částí. První část obsahuje informace týkající se spojení se serverem, o přihlášeném uživateli a nastaveném jazyku. Druhá část obsahuje akce, které lze realizovat pomocí tlačítek umístěných

ve spodní části formuláře.

Další dialogy jsou koncipované velmi podobě jako hlavní okno aplikace. Obsahují logo, výzvu k akci, kterou má uživatel provést (je-li vyžadovaná) a formulář sestávající z polí s informacemi a tlačítek.

Na obrázku 4.3 je zobrazeno hlavní okno aplikace z pohledu použitých grafických komponent.



Obrázek 4.3: Uživatelské rozhraní - komponenty

Celá aplikace je obalená komponentou `AnchorPane`[5] představující hlavní okno aplikace. Komponenta `AnchorPane` obsahuje element `VBox`.

`VBox` je kontejner, v němž jsou komponenty, které obsahuje, řazeny pod sebe (tedy vertikálně). Tento kontejner je tvořený dalšími dvěma elementy. Jsou to elementy `MenuBar` a `GridPane`. `MenuBar` je vrchní menu aplikace. `GridPane` představuje obsah hlavního okna, kterým je formulář.

Vrchní menu obsahuje tři hlavní nabídky: File, Open, Help. Pod těmito nabídkami

[5] `http://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/AnchorPane.html`

se nachází následující možnosti:

- nahrát konfigurační soubor
- aktualizovat spojení se serverem
- zavřít hlavní okno aplikace
- ukončit aplikaci
- otevřít nastavení serveru ve webovém rozhraní
- otevřít nastavení
- otevřít dialog O aplikaci

`GridPane` je kontejner, který řadí komponenty do sloupců a řádků podle zadaných indexů. Princip byl vysvětlen v teoretické části této práce.

Následuje ukázka zdrojového kódu šablony hlavního okna aplikace a rozbor použitých komponent.

```
<AnchorPane
    xmlns="http://javafx.com/javafx/8.0.65" xmlns:fx="http://javafx
        .com/fxml/1"
    fx:controller="cz.myq.ejm.controllers.ManagerController">
  <children>
    <VBox AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0"
        AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
      <MenuBar id="menu-bar" fx:id="menuBar" onKeyPressed="#
          handleKeyInput">
        <menus>
          <Menu text="%FILE">
            <items>
              <MenuItem onAction="#uploadConfigAction" text="%
                  UPLOAD_CONFIG" styleClass="admin-action" />
              <MenuItem onAction="#updateAction" text="%REFRESH" />
              <SeparatorMenuItem />
              <MenuItem fx:id="closeItem" onAction="#closeAction"
                  text="%CLOSE" />
```

```
                <MenuItem fx:id="exitItem" onAction="#exitAction"
                    text="%EXIT" />
                        </items>
            </Menu>
            <Menu text="%OPEN">
              <items>
                <MenuItem onAction="#runAction" fx:id="runOnWeb" />
                <SeparatorMenuItem />
                <MenuItem fx:id="settingsItem" onAction="#
                    settingsAction" text="%SETTINGS" styleClass="admin
                    -action" />
              </items>
             </Menu>
            <Menu text="%HELP">
              <items>
                <MenuItem onAction="#aboutAction" text="%
                    ABOUT_APPLICATION" />
              </items>
            </Menu>
          </menus>
        </MenuBar>
        <fx:include fx:id="top" source="top.fxml" AnchorPane.
            rightAnchor="0" AnchorPane.leftAnchor="0" />
    </VBox>
  </children>
</AnchorPane>
```

Aby nebyla šablona hlavního okna příliš dlouhá a nepřehledná, je formulář umístěn ve vlastním samostatném souboru, který je vložen do šablony hlavního okna pomocí tagu `<fx:include>`.

Komponenta `VBox` obsahuje jako parametry tzv. kotvy (anglicky anchors): `bottomAnchor`, `topAnchor`, `leftAnchor`, `rightAnchor`. Kotvy určují způsob přichycení komponenty `VBox` k okrajům[6] rodičovské komponenty `AnchorPane`. Všechny tyto parametry jsou mastavené na hodnotu 0.0. To znamená, že bude komponenta na svého rodiče přilepená.

Další použitou komponentou je `MenuBar`. Tato komponenta představuje vrchní panel menu se třemi, výše zmiňovanými, hlavními nabídkami. Hlavní nabídky jsou reprezen-

[6]bottom - ke spodnímu okraji, top - k vrchnímu okraji, left - k levému okraji, right - k pravému okraji

továny komponentami `Menu` a položky v nich komponentami `MenuItem`.

Na každou položku je prostřednictvím parametru `onAction` navázána akce. Její obsluha se odehrává ve třídě `ManagerController`.

4.2 Realizace

Během návrhu bylo navrženo uživatelské rozhraní aplikace a nové workflow. V dalším kroku bude provedena implementace rozhraní pro komunikaci se serverem a implementace nového uživatelské rozhraní spolu s funkcionalitou aplikace. Tato kapitola se věnuje popisu kontkrétní implementace. Zaměřuje se na řešení nejzajímavějších problematik.

4.2.1 Nastavení IDE

Pro vývoj bylo vybráno vývojové prostředí IntelliJ IDEA. Abychom mohli začít s implementací, je nutné jej nejprve stáhnout, nainstalovat a nastavit. Kroky jsou následující:

1. Jednotlivé verze vývojového prostředí Intelij IDEA jsou dostupné na oficiálních stránkách Jetbrains: `https://www.jetbrains.com/idea/download`.

2. Protože instalaci provádíme na OS Linux, provedeme jí rozbalením staženého archivu `.tar.gz` do vybraného adresáře:

   ```
   $ sudo tar xfz idea-IU-162.2228.15.tar.gz /home/doubkova/opts
   ```

3. Spustíme soubor `idea.sh` v podadresáři bin.

4. Dále je nutné stáhnout a nainstalovat JDK 8[7].

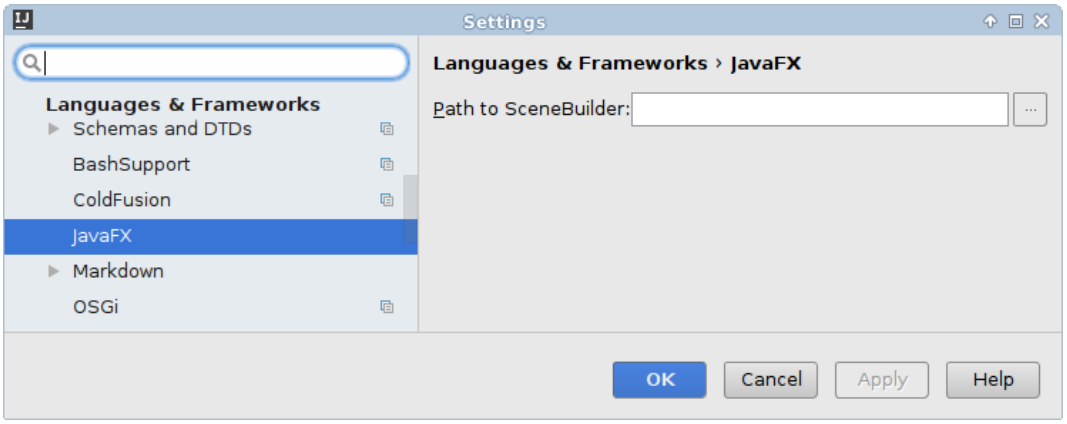
5. Stáhnout JavaFX Scene Builder pro tvorbu UI[8].

[7]JavaFX SDK je obsažena již ve verzi JDK 7, nejnovější verze JDK je dostupná ke stažení na http://www.oracle.com/technetwork/java/javase/downloads/index.html.

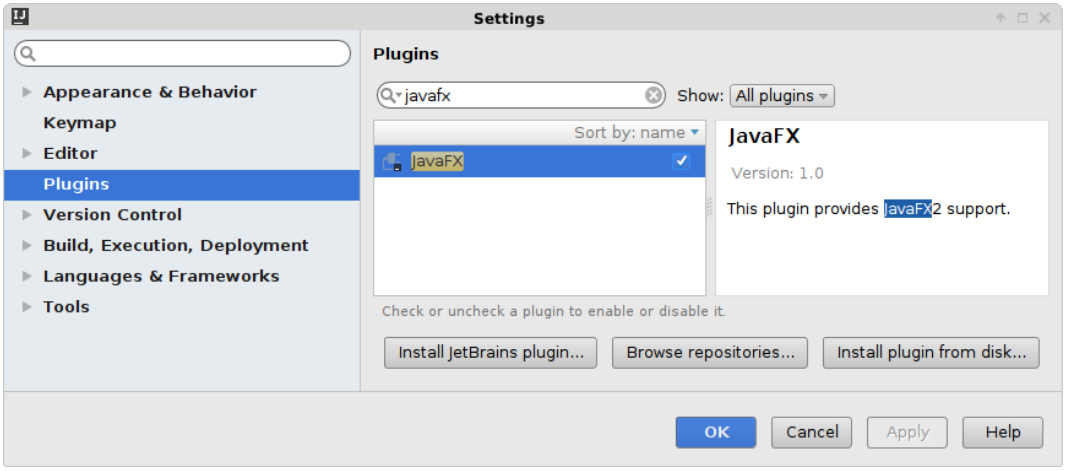
[8]http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

6. Nastavit cestu k JavaFX Scene Builder v IntelliJ IDEA (viz Obrázek 4.4).

7. V nastavení IntelliJ IDEA je nutné ověřit, že je povolen JavaFX plugin[9] (viz Obrázek 4.5). Cesta k nastavení je *File* > *Settings* > *Plugin*.



Obrázek 4.4: IntelliJ IDEA - Nastavení cesty k nástroji Scene Builder



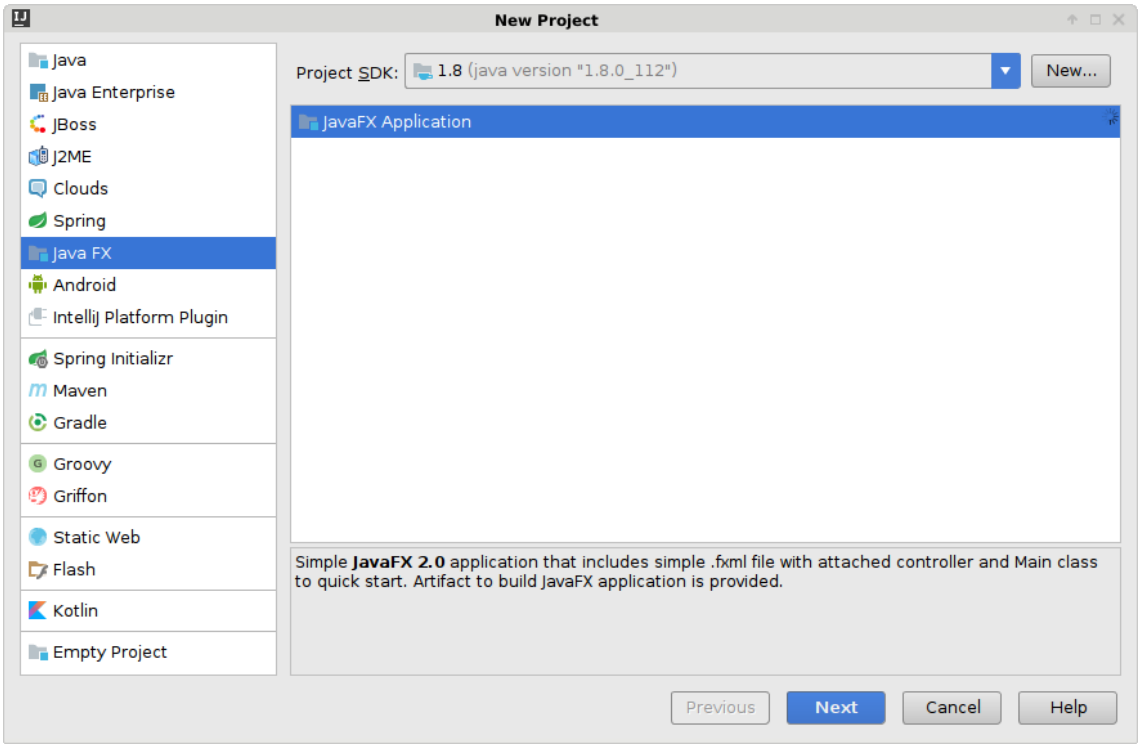
Obrázek 4.5: IntelliJ IDEA - Povolení JavaFX pluginu

[9]Bývá povolen v rámci defaultního nastavení.

4.2.2 Vytvoření projektu

Vývojové prostředí je nastavené podle výše popsaných kroků. Nyní můžeme přistoupit k samotnému vytvoření projektu.

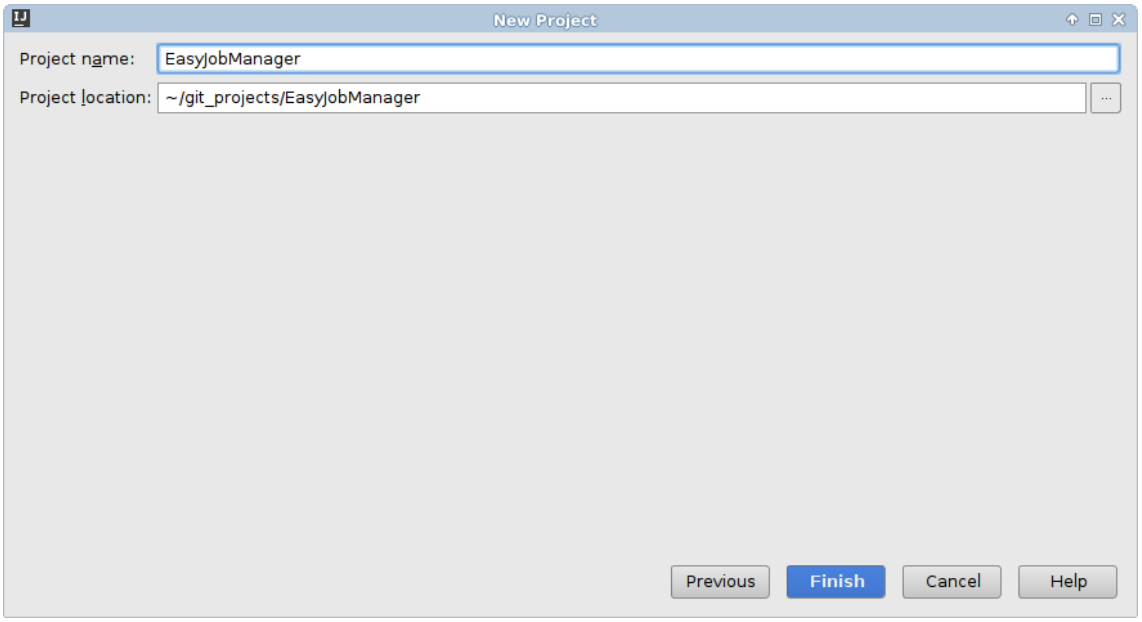
Výběrem volby *File* > *New* > *Project...* z hlavního menu otevřeme dialog pro vytvoření nového projektu. V levém menu otevřeného dialogu vybreme možnost Java FX.



Obrázek 4.6: Nový JavaFX projekt - krok 1

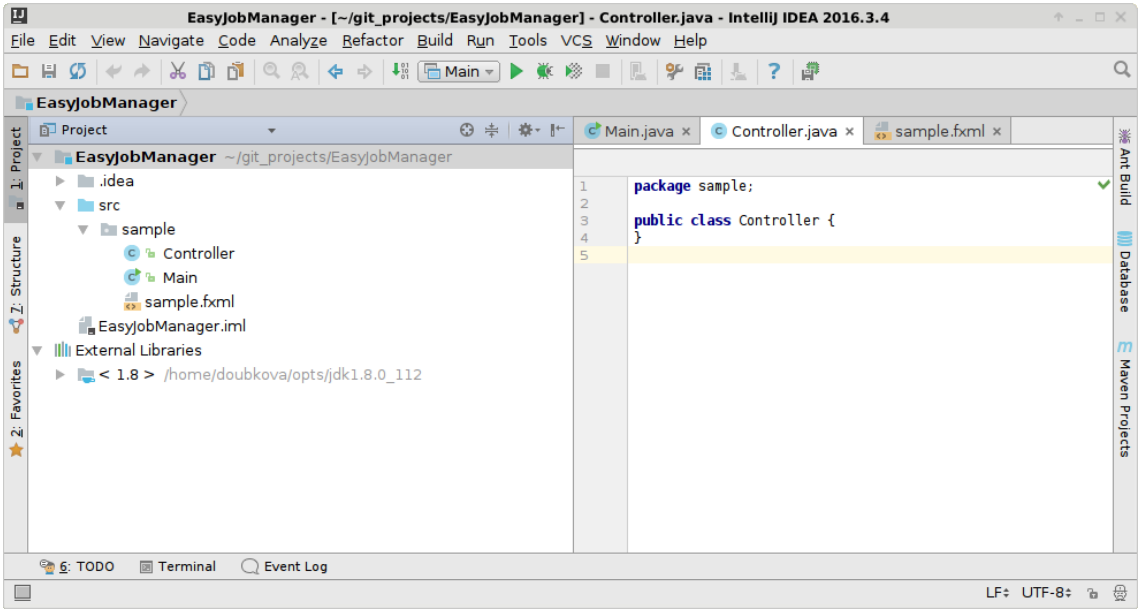
V hlavním okně dialogu se zobrazí informace, že se jedná o projekt JavaFX. Nad touto informací nalezneme pole pro výběr verze SDK, kterou chceme pro daný projekt použít. Vybereme požadovanou verzi a pokračujeme volbou *Next*.

V dalším kroku zvolíme název projektu a jeho umístění. Volbou *Finish* ukončíme dialog a vytvoříme JavaFX projekt se základní strukturou.



Obrázek 4.7: Nový JavaFX projekt - krok 2

Nově vytvořený projekt má jednoduchou strukturu. Jak je vidět na obrázku 4.8 obsahuje pouze dvě javovské třídy a jednu fxml šablonu.



Obrázek 4.8: Nový JavaFX projekt - struktura

Jednou z javovských tříd je třída `Main.java`. Jak bylo řečeno v teoretické části této práce, jedná se o vstupní bod pro všechny JavaFX aplikace.

Druhou třídou je `Controller.java`, který je propojený se šablonou `sample.fxml`. Umožňuje měnit vlastnosti komponent této šablony a reagovat na jejich události.

Tato základní struktura se postupným přidáváním balíčků a tříd během vývoje značně změnila..

4.2.3 Komunikace se serverem

Další fází po vytvoření projektu je samotná implementace. Jako první byla naimplementována logika komunikace se serverem. Do automaticky vygenerované struktury byl přidán balíček (anglicky package) obsahující řešení této problematiky.

Z jeho názvu je patrné, jaké třídy obsahuje a o co se starají. Balíček byl pojmenován `server` a obsahuje jeden vnořený balíček s názvem `communication`. Celá cesta ke třídám v tomto balíčku je `cz.myq.ejm.server.communication`.

Komunikace se serverem je realizovaná pomocí HTTP požadavků a UDP komunikace. O volání HTTP požadavků se stará třída `WebServiceAPI`, třída `HttpConnection` představuje HTTP spojení a UDP komunikaci zajišťuje třída `UdpClient`.

Třída WebServiceAPI

Třída `WebServiceAPI` je naimpletovaná jako návrhový vzor singleton. V celém programu běží pouze jedna instance této třídy, která se stará o volání HTTP požadavků na server.

Instance je vytvořena uvnitř třídy `WebServiceAPI` a je uložena do privátní statické proměnné `instance` tak, jako na následující ukázce.

```
public class WebServiceAPI {
  private static WebServiceAPI instance;
```

```
    public static WebServiceAPI getInstance() {
        if (instance == null) instance = new WebServiceAPI();
        return instance;
    }

    private WebServiceAPI() {}

    .
    .
    .
}
```

Třída HttpConnection

Nejdůležitější metodou této třídy je metoda `getHttpConnection`, která vrací objekt typu `HttpURLConnection`.

```
private HttpURLConnection getHttpConnection(String urlS) throws
    IOException, KeyManagementException, NoSuchAlgorithmException {
     boolean ssl = Boolean.parseBoolean(PropertiesManager.
        getInstance().getProperty("sslEnabled"));
     String hostname = PropertiesManager.getInstance().getProperty("
        serverAddress");
     String protocol = ssl ? HTTPS_PROTOCOL : HTTP_PROTOCOL;
     int port = Integer.parseInt(PropertiesManager.getInstance().
        getProperty("serverPort"));

     URL url = new URL(protocol, hostname, port, urlS);

     if (ssl) {
         initSSLContext();
         HttpsURLConnection httpsURLConnection = (HttpsURLConnection
             ) url.openConnection();
         httpsURLConnection.setSSLSocketFactory(sslContext.
             getSocketFactory());
         httpsURLConnection.setHostnameVerifier(hostnameVerifier);
         return httpsURLConnection;
     } else {
         return (HttpURLConnection) url.openConnection();
     }
}
```

Prostřednictvím třídy `PropertiesManger` jsou z nastavení aplikace EJM postupně vyčteny všechny parametry potřebné pro vytvoření objektu typu URL. Zavoláním metody `openConnection` na tomto objektu je vytvořena instance třídy `HttpUrlConnection`, která je následně vrácena.

Třída `HttpConnection` poskytuje dvě veřejné metody. Jsou jimi `sendGet` a `sendPost`. Obě metody jsou volané třídou `WebServiceAPI`. Metoda `sendGet` je volaná například v případě potřeby ověření, že je server dostupný. Metoda `sendPost` v připadech, kdy jsou zároveň v těle dotazu posílána data. Jedná se například o požadavek pro získání seznamu tiskových úloh, kde je nutné blíže specifikovat, pro jakého uživatele je vyžadovaná odpověď.

Třída UdpClient

Třída `UdpClient` obstarává UDP komunikaci se serverem. Běží v samostatném vlákně a poslouchá na specifikovaném portu na příchozí paket. Po přijetí paketu zpracuje přijatá data a dále poslouchá na specifikovaném portu až do doby ukončení aplikace nebo do doby, kdy se server stane nedostupným.

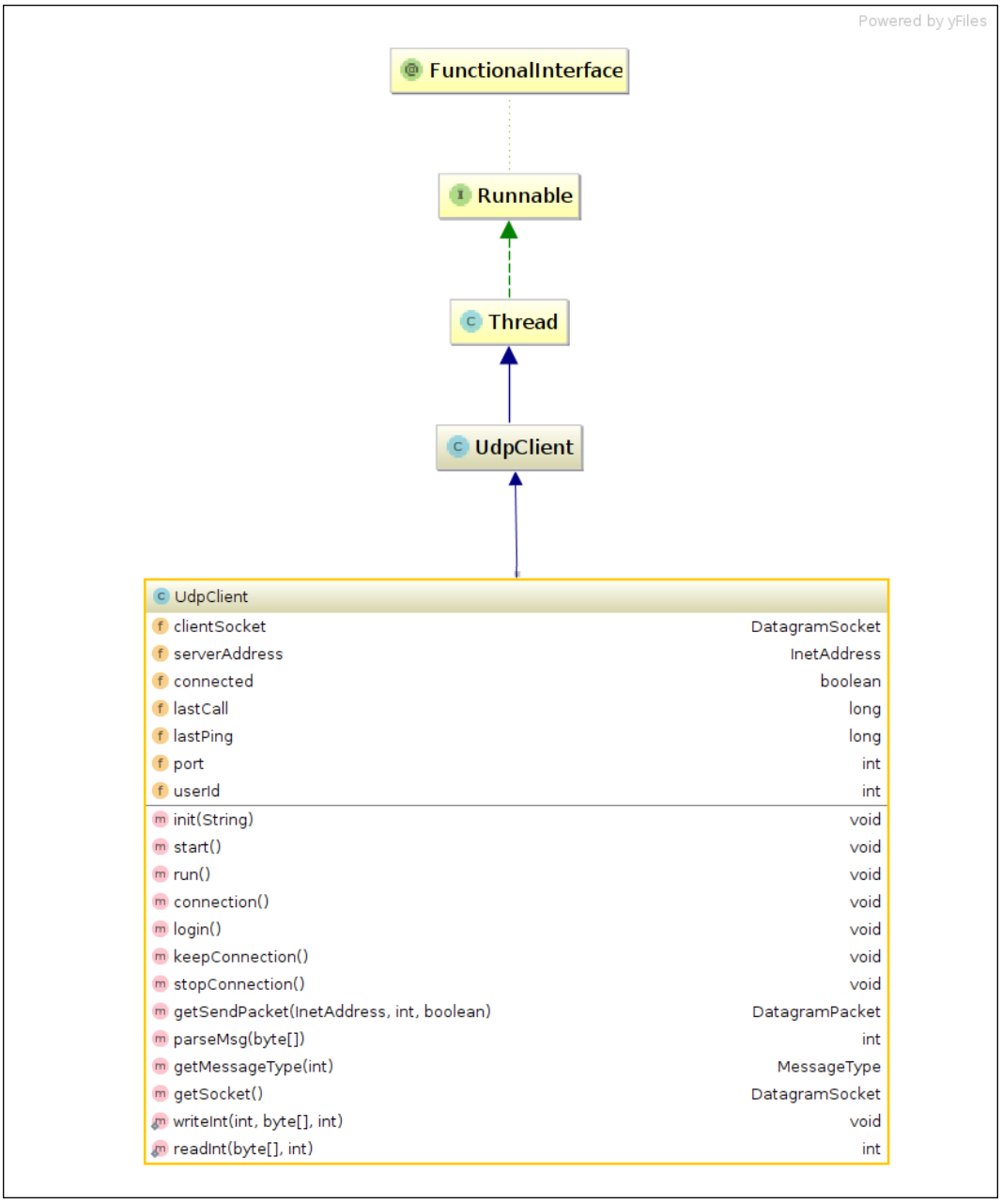
Klient je spuštěn z hlavního vlákna aplikace po startu aplikace vytvořením instance třídy `UdpClient` a následným zavoláním metody `start` na vytvořenou instanci tak, jak je to uvedeno v následující ukázce.

```
udpClient = new UdpClient(user.getId(), PropertiesManager.
    getInstance().getProperty("serverAddress"));
udpClient.start();
```

Volání metody start vyvolá vykonání metody `run` naimplementované ve třídě `UdpClient`.

```
public void run() {
    try {
        connection();
    } catch (Exception e) {
        System.err.println("Exception in UdpClient thread: " + e);
    } finally {
        System.out.println("UdpClient thread closed.");
    }
}
```

Tím je celý proces spuštění vlákna obstarávajícího UDP komunikaci se serverem ukončen. Na obrázku 4.9 je zobrazen diagram třídy `UdpClient` včetně atributů a metod, které obsahuje. Z obrázku je dále patrné, že rozšiřuje třídu `Thread`, která implementuje rozhraní `Runnable`.



Obrázek 4.9: Diagram třídy UdpClient *[vygenerováno v prostředí IntelliJ IDEA]*

4.2.4 Třída DialogsManager

V projektu aplikace EJM jsou použity dva druhy dialogových oken. Ty, které jsou vytvořeny jako fxml šablona, a ty, které jsou generované programově. Příkladem fxml šablony je hlavní okno aplikace popsané v sekci 4.1.3. O všechna ostatní dialogová okna se stará třída `DialogsManager` umístěná v balíčku `cz.myq.ejm.managers`, který prostřednictvím sedmi veřejných metod umožňuje vygenerovat a zobrazit ten který dialog. Přehled metod obsahuje tabulka 4.1.

Metoda	**Popis**
`settingsWindow`	Zobrazí dialog s nastavením aplikace. Před jeho zobrazením je uživatel požádán o zadání přihlašovacích údajů privilegovaného uživatele.
`projectsWindow`	Je-li při tisku vyžadován výběr projektu, zobrazí dialog se seznamem projektů.
`aboutWindow`	Zobrazí dialog se základními informacemi o aplikaci.
`authWindow`	Je-li vyžadovaná autorizace tiskové úlohy, zobrazí dialog pro zadání uživatelského jména a hesla nebo dialog pro zadání PINu.
`uploadConfigWindow`	Zobrazí dialog pro nahrání konfiguračního souboru. Před jeho zobrazením je uživatel požádán o zadání přihlašovacích údajů privilegovaného uživatele.
`getInfoDialog`	Zobrazí informační dialog.
`getErrorDialog`	Zobrazí dialog s chybovou hláškou.

Tabulka 4.1: Třída DialogsManager - přehled veřejných metod

Následující ukázka obsahuje dvě metody s názvem `uploadConfigWindow`. První z nich je umístěná ve třídě `DialogsManager` a způsobuje vyvolání dialogu pro zadání přihlašovacích údajů privilegovaného uživatele. Jako parametr předává volané metodě konstantu akce, která se má vykonat v případě správného zadání údajů, a na jejímž základě

dojde k volání druhé z uvedených metod. Ta je umístěna ve třídě `AdminActionManager` starající se o akce, které může provést pouze privilegovaný uživatel. Pro zobrazení dialogu pro nahrání konfiguračního souboru je volána třída `FileChooser`, která je standardní součástí balíčku `javafx.stage`.

```
public static void uploadConfigWindow() {
    AdminActionsManager.getInstance().requireSudoPasswd(
        AdminActionsManager.ACTION_UPLOAD_CONFIG, null);
}

private void uploadConfigWindow() {
    FileChooser fileChooser = new FileChooser();
    File selectedFile = fileChooser.showOpenDialog(null);
    if (selectedFile != null) {
        if (PropertiesManager.getInstance().uploadPropertiesFile(
            selectedFile)) Main.forceUpdate();
    }
}
```

Druhá ukázka zobrazuje implementaci metody `getErrorDialog`. Uvnitř této metody je vygenerován a zobrazen dialog typu `Alert` zobrazující chybovou hlášku, jejíž text je předán v parametru této metody. Zajímavostí na této metodě je použití ikony patřící do externí knihovny FontAwesomeFX.

```
public static void getErrorDialog(String message) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    DialogPane dialogPane = alert.getDialogPane();
    dialogPane.getStyleClass().add("custom-alert-dialog");
    dialogPane.getStyleClass().add("error");
    alert.setHeaderText(message);
    alert.setGraphic(GlyphsDude.createIcon(FontAwesomeIcon.TIMES\
        _CIRCLE, "28px"));
    dialogPane.getStylesheets().add(Main.class.getClassLoader().
        getResource("styles/styles.css").toExternalForm());
    Stage main = Main.getStage();
    if (main.getScene() != null) {
        alert.initOwner(Main.getStage());
        alert.showAndWait();
    }
}
```

4.2.5 Internacionalizace

Pro podporu více jazyků, nebo-li internacionalizaci[10], je použita třída `ResourceBundle` obsažená v balíčku `java.util`. Tato třída umožňuje načítat překlady textů použitých v programu z textových souborů. Princip spočívá v tom, že každému textovému řetězci je přiřazen unikátní kód, se kterým se v programu pracuje místo konkrétního textového řetězce, a na základě kterého je použit správný překlad v daném jazyce.

Pro zabudování podpory více jazyků do aplikace EJM byl pro každý podporovaný jazyk vytvořen textový soubor s překlady. Názvy jednotlivých souborů jsou ve tvaru *bundle_ZKTRATKA_JAZYKA.properties*. Každý soubor obsahuje seznam textových řetězců. Každý řetězec má přiřazený unikátní kód. Následuje ukázka části souboru bundle_en.properties:

```
MYQ_EJM=MyQ Easy Job Manager
SELECT_PROJECT_TITLE=Assigning projects
WITHOUT_PROJECT_OPTION=Without project
ALL=All
USERNAME_LBL=User name
PASSWORD_LBL=Password
```

Řetězce lze volat z fxml šablon za použití unikátního kódu, který následuje po znaku % následovně:

```
<Label fx:id="statusLbl" styleClass="bold" text="\%STATUS\_LBL" />
```

Další možností je předat řetězce do šablon z javovských tříd předáním ResourceBundle do FXMLoader:

```
new FXMLLoader(Main.class.getClassLoader()
  .getResource(PATH + "managerPage.fxml"), getResourceBundle()).load();
```

Řetězce lze volat přímo v javovských třídách tak, jako je na ukázce nastaven text na tlačítku pro odhlášení:

```
logoutBtn.setText(resourceBundle.getString("LOGOUT\_BTN"));
```

[10]Známou pod označením **i18n**.

4.2.6 Sestavování aplikace

pom.xml

V této části jsou vybrány ukázky důležitých úseků skriptu `pom.xml`. Výstupem tohoto skriptu je soubor `ejm-{cislo_verze}.jar`. K jeho generování je použit `maven-jar-plugin`, kterému je předána cesta k hlavní třídě projektu.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>2.6</version>
    <configuration>
        <archive>
            <manifest>
                <mainClass>cz.myq.ejm.app.Main</mainClass>
            </manifest>
        </archive>
    </configuration>
</plugin>
```

Další důležitou částí jsou závislosti projektu na externích knihovnách, kterými jsou fontawesomefx nabízející sadu ikon, podpora formátu json a aqaufx téma pro Mac OS.

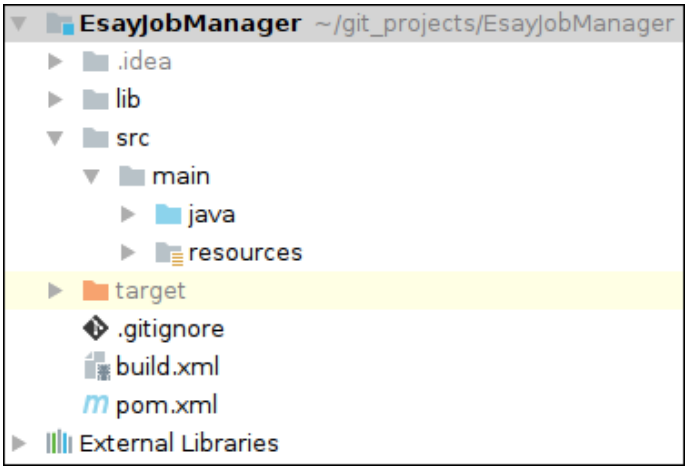
```
<dependencies>
    <dependency>
        <groupId>de.jensd</groupId>
        <artifactId>fontawesomefx</artifactId>
        <version>8.9</version>
    </dependency>
    <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
        <version>20160810</version>
    </dependency>
    <dependency>
        <groupId>com.aquafx-project</groupId>
        <artifactId>aquafx</artifactId>
        <version>0.2</version>
    </dependency>
</dependencies>
```

build.xml

Celý soubor `build.xml` je součástí tohoto dokumentu jako Příloha B. Vstupem tohoto skriptu je soubor `ejm-{cislo_verze}.jar` vygenerovaný skriptem `pom.xml` a výstupem je archiv `Easy Job Manager.tar.gz`, který obsahuje spustitelný soubor `.app`.

4.2.7 Struktura aplikace

Tato část popisuje výslednou strukturu aplikace, která je zobrazena na obrázku 4.10.



Obrázek 4.10: Výsledná struktura projektu

Strukturu lze rozdělit do pěti následujících částí:

1. Adresář *lib*, kde je umístěn jar soubor aplikace Java Application Bundler používaný při vytváření balíčku aplikace pro Mac OS.

2. Adresář *src*, který obsahuje zdrojové kódy aplikace.

3. Adresář *target*, do nějž jsou automaticky generovány výstupy buildu (sestavovacího procesu).

4. Konfigurační soubory `build.xml` a `pom.xml`.

5. Adresář *External Libraries* obsahující externí knihovny, na nichž má projekt Easy Job Manager závislost.

src

Javovské zdrojové kódy jsou umístěné ve zdrojovém adresáři *java*, balíčku **cz.myq.ejm**, který obsahuje pět dalších balíčků. Jedná se o tyto balíčky:

1. **app** - obsahuje třídy týkající se obecně celé aplikace. Nejdůležitější třídou v tomto balíčku je třída `Main`, která je vstupním bodem aplikace.

2. **controllers** - jedná se o soubor tříd, které jsou provázané s fxml šablonami. Tyto třídy obsahují všechna pole, která mají v fxml šabloně vyplněný atribut `fx:id`. Tato pole jsou v provázané třídě reprezentována atributem stejného typu a se stejným názvem jako pole. Dále tyto třídy obsahují metody obsluhující zpracování událostí navázaných na nějaký element fxml šablony (například kliknutí na prvek v horním menu aplikace). Atributy a metody provázané s fxml šablonou jsou označené anotací @FXML.

3. **entities** - do tohoto balíčku jsou zahrnuty třídy představující entity, se kterými se pracuje v rámci aplikace. Jedná se o třídy `Project`, `User` a `Job`. Třídy jsou jednoduché, neimplementují žádné rozhraní, ani nedědí od žádné třídy. Obsahují bezparametrický konstruktor a soubor privátních atributů, ke kterým umožňují přístup prostřednictvím getterů a setterů. Třidu s takovými vlastnosti označujeme jako Plain Java Object, zkráceně POJO.

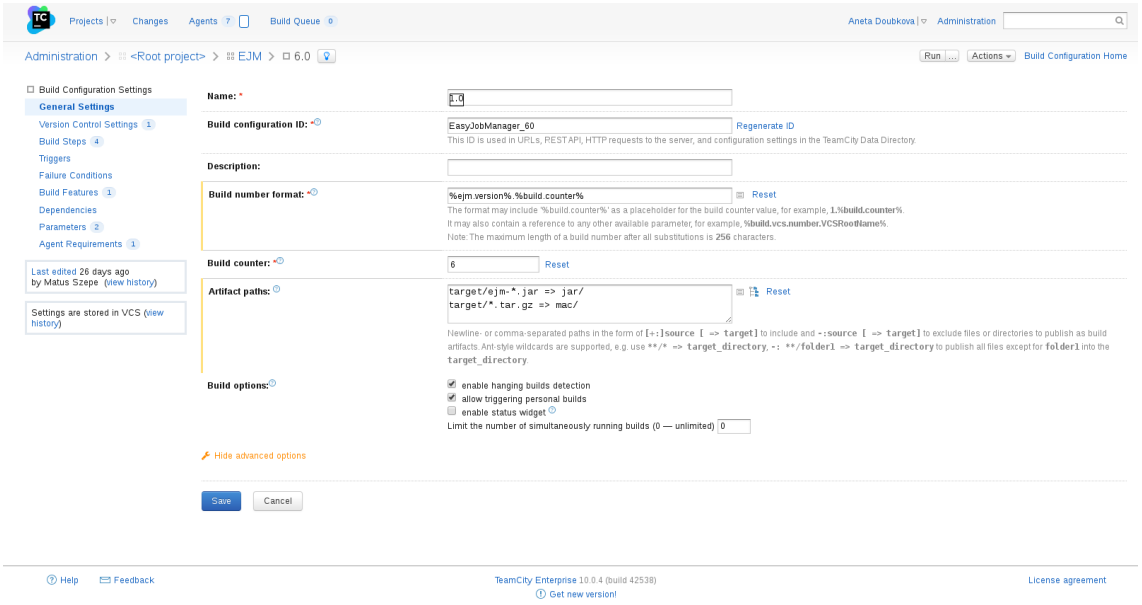
4. **managers** - v tomto balíčku jsou umístěné třídy, které centrálně řídí akce týkající určité problematiky. Některé z nich byly zmíněny v předchozím popisu. Například `DialogsManager`, který se stará o dynamické generování a zobrazování dialogových oken nebo `PropertiesManager`, který má na starosti správu nastavení aplikace (ukládá a vyčítá nastavení jak je potřeba).

5. **server** - do tohoto balíčku je vnořen balíček `communication`, který byl popsán výše, konkrétně v části 4.2.3.

Součástí adresáře src je dále adresář *resources*, který obsahuje použité šablony, styly, obrázky a překlady potřebné pro internacionalizaci.

4.3 Verze na TC

Jednotlivé verze aplikace jsou sestavovány a šířeny v rámci týmu prostřednictvím buildovacího serveru TeamCity. Server byl propojen s repozitářem Git prostřednictvím pluginu.

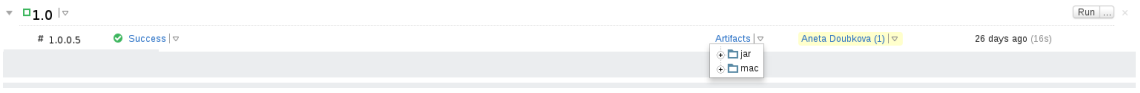


Obrázek 4.11: TC - Konfigurace

Administrační rozhraní pro konfiguraci projektu je zobrazeno na obrázku 4.11. V záložce *General Settings* jsou nastaveny základní informace o projektu jako je jeho název, číslo verze či cesta ke generovaným výstupům. V dalších záložkách je nastaveno propojení s repozitářem, kde jsou uloženy zdrojové kódy (*Version Control Settings*), jednotlivé

kroky sestavování aplikace [11] a volitelné konfigurační parametry projektu.

Jak je vidět na obrázku 4.12, výstupy (nazývané artefakty, anglicky artifacts) této konfigurace jsou dva adresáře *jar* a *mac*. Adresář *jar* obsahuje výstup skriptu *pom.xml* a je určen pro šíření na distribuce Linuxu. Adresář *mac* obsahuje výstup skriptu *build.xml* a je určen pro šíření na Mac OS.



Obrázek 4.12: TC - Výstup

[11] Zde jsou použity konfigurační soubory build.xml a pom.xml

Kapitola 5

Zhodnocení výsledků

V této kapitole jsou zhodnoceny a popsány výstupy praktické části práce.

5.1 Testování

Vytvořená aplikace byla předána na QA[1] oddělení k otestování. Testy potvrdily, že byly splněny všechny uživatelské požadavky stanovené na začátku celého procesu, vyjma požadavků týkajících se správy tiskových úloh. Z těchto požadavků byl splněn požadavek na mazání a tisk úloh. Požadavky na přehled, multiselekci a řazení tiskových úloh nebyly splněny z důvodu změny workflow [2].

Testy dále odhalily následující chyby:

- Změna jazyka aplikace není trvalá. Projeví se pouze bezprostředně po provedení. Restart aplikace způsobí vyresetování aplikace do defaultně nastaveného jazyka.
- Pole pro zadání PINu nemá na sobě focus po otevření dialogu pro autorizaci tiskové úlohy. Uživatel tak musí nejprve pole označit kliknutím myši dříve, než může začít psát.

[1] QA - Quality Assurance

[2] Nové worklow neobsahuje seznam tiskových úloh.

- Pokud je na straně serveru změněno jméno uživatele, neprojeví se tato změna ve spuštěné aplikaci.
- Některé položky horního menu nejsou přeložené do češtiny.

Na základě testů byly doporučeny následující úpravy:

- V případě neúspěšné autorizace tiskové úlohy by mělo být po potvrzení chybového dialogu znovu zobrazen dialog pro přihlášení.
- Odstranění čísla verze buildu z dialogu *O aplikaci*.
- Uživatel by měl mít možnost měnit číslo portu v nastavení aplikace[3].
- Po úspěšném odeslání všech úloh na tisk by měla být celá aplikace minimalizována do ikony umístěné v notifikačním panelu operačního systému.

Nalezené chyby byly odstraněny, doporučení byla zapracována. Výsledná aplikace je popsána v části 5.2.

5.2 Výsledná aplikace

Po několika změnách oproti původní aplikaci dostal nový Easy Job Manager finální podobu. Běží jako TrayApp a má ikonu v notifikačním panelu. Ikona má pod sebou schované menu s rychlým přístupem k základním akcím (zobrazení hlavního okna aplikace, nastavení, dialogu O aplikaci a možnost ukončení aplikace).

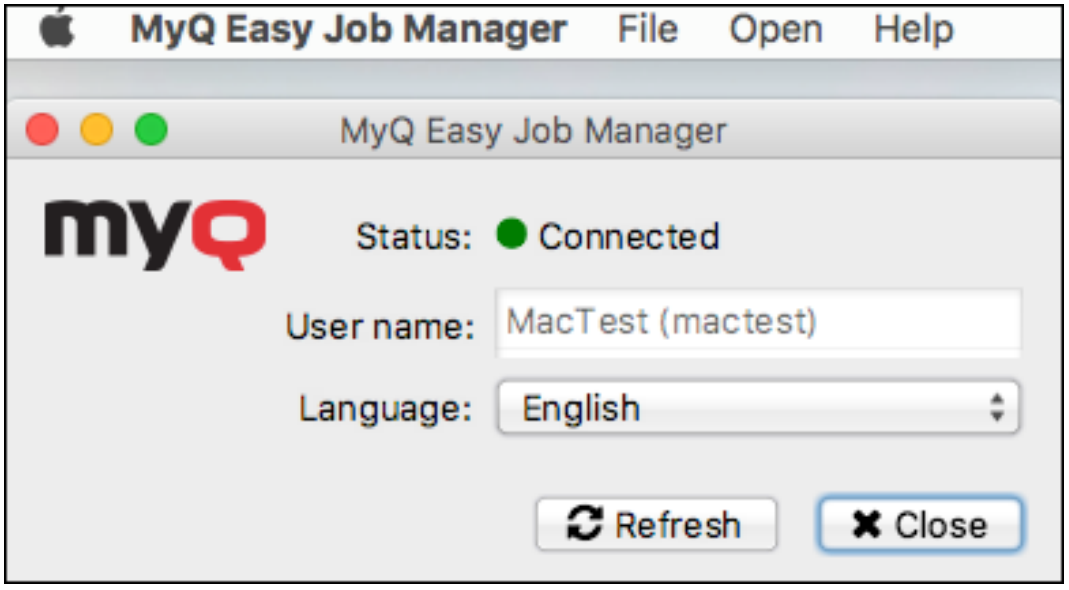
Jedná se o aplikaci založenou na následujících pěti dialogových oknech, která jsou uživateli zobrazována pouze v případě nutnosti:

1. Hlavní okno

   Jak je patrné z obrázku 5.1, hlavní okno, dále dialog, obsahuje informaci, zda bylo

[3]Nyní se číslo portu mění automaticky podle toho, zda je povolené SSL spojení (port 8080) či nikoliv (port 8090).

navázáno spojení se serverem (Status: connected / Status: not connected), dále přihlašovací jméno uživatele, pod kterým bylo spojení navázáno a vybraný jazyk. Uživatel má možnost tento dialog zavřít tlačítkem *Close* či tlačítkem *Refresh* obnovit spojení se serverem (dojde ke stažení projektů, tiskových úloh, apod.).



Obrázek 5.1: EJM pro Mac OS - hlavní okno aplikace

V případě, že byla provedena autorizace tiskové úlohy a byly zapamatovány přihlašovací údaje, bude hlavní dialog vypadat trošku jinak. Na místo přihlašovacího jména uživatele, pod kterým bylo navázáno spojení se serverem, bude uvedeno přihlašovací jméno autorizovaného uživatele (pokud byla autorizace provedena pod uživatelským jménem a heslem) nebo *PIN user* (pokud byla autorizace provedena PINem) a přibyde tlačítko *Logout* pro odhlášení autorizovaného uživatele.

2. Autorizace tiskové úlohy

   Autorizace může být provedena dvěma způsoby. Zda se zobrazí dialog pro zadání PINu nebo uživatelského jména a hesla, záleží na nastavení fronty na serveru[4]. Oba dialogy umožňují nastavit volbu *Remember* na jednu z následujících možností:

   - nikdy
   - do odhlášení
   - 1 minuta
   - 5 minut
   - 10 minut

   Je-li vybrána jakákoliv jiná možnost, než nikdy, nebude až do odhlášení [5] autorizovaného uživatele při tisku zobrazován autorizační dialog.

3. Výběr projektu

   V případě, že jsou na serveru zapnuté projekty, je tento dialog zobrazován pro každou tiskovou úlohu. Z toho důvodu je v záhlaví uveden název tiskové úlohy. Dále je zde k dispozici seznam dostupných projektů. Defaultně je vybrán poslední použitý projekt. V případě, že ještě žádný projekt nebyl použit, je vybrán první projekt ze seznamu. Není-li dialog potvrzen tlačítkem *Print*, je tisková úloha smazána.

4. Nastavení

   Tento dialog umožňuje administrátorovi změnit adresu serveru a povolit bezpečné připojení (SSL). Nastavení může být změněno také nahráním konfiguračního souboru (jedna z možností v hlavním menu aplikace). Konfigurační soubor bude v rámci rozšíření do budoucna generován na serveru na vyžádání.

[4] Jak bylo popsáno v úvodu teoretické části této práce.

[5] Odhlášení může být provedeno manuálně stiskem tlačítka *Logout* nebo automaticky po vypršení vybrané doby či ukončení aplikace

5. O aplikaci

   Informační dialog se základními informacemi o aplikaci a odkazem na příslušné webové stránky.

Kapitola 6

Závěr

Hlavním cílem práce bylo analyzovat současný stav v programování aplikací v jazyce Java. Protože se jedná o velmi rozsáhlou platformu, byla nejprve provedena analýza stávající aplikace EJM pro OS Windows.

Pomocí analýzy byly identifikovány požadavky kladené na novou aplikaci EJM určenou primárně pro Mac OS a na jejich základě mohl být zúžen okruh analyzovaných technologií. Technologie vybrané pro realizaci praktické části jsou následující:

- Framework JavaFX pro tvorbu uživatelského rozhraní.
- Téma AquaFX pro nastylování aplikace jako nativní Mac OS aplikace.
- Balíček java.net pro realizaci komunikace klient - server.
- Nástroje Ant a Maven pro sestavení aplikace.
- Výjové prostředí IntelliJ IDEA.
- Git jako verzovací systém pro správu zdrojových kódů.

Praktická část práce je postavená na poznatcích získaných v teoretické části a ukazuje použití vybraných technologií na implementaci konkrétní aplikace. Popisuje návrh uživatelského rozhraní a workflow aplikace včetně tvorby prototypu. Dále popisuje

nejdůležitější části samotné implementace, sestavování aplikace a způsob vydávání verzí. Výstupem praktické části práce je první verze reálné aplikace, která byla nasazena do provozu na podzim roku 2016.

Seznam použitých zdrojů

[1] MyQ. *MyQ Produktová brožura*. Praha: MyQ, spol. s r.o., 2015.

[2] Andrew Binstock. *Nejlepší nástroje na programování v Javě* [online]. 2011. [cit. 2017-02-19]. Dostupné z: `http://computerworld.cz/vyvoj/nejlepsi-nastroje-na-programovani-v-jave-42939`.

[3] ARLOW, J. – NEUSTADT, I. *UML2 a unifikovaný proces vývoje aplikací*. : Brno: Computer Press, a.s., první vydání, 2008. 567 s. ISBN: 978-80-251-1503-9.

[4] Bradley Mitchell. *How to Network a Printer* [online]. 2016. [cit. 2017-01-07]. Dostupné z: `https://www.lifewire.com/networking-a-printer-817579`.

[5] DARWIN, I. F. *Java Kuchařka programátora*. Brno : Computer Press, a.s., 2007. 798 s. ISBN: 978-80-251-0944-5.

[6] David Čápka. *Úvod do JavaFX* [online]. 2016. [cit. 2017-02-22]. Dostupné z: `http://www.itnetwork.cz/java/javafx/java-tutorial-uvod-do-javafx/`.

[7] FRANTIšEK ZEZULKA, O. H. Průmyslový Ethernet II: Referenční model ISO/OSI. *AUTOMA*. 2007.

[8] HEROUT, P. *Učebnice jazyka Java*. České Budějovice : KOPP, 2000. 349 s. ISBN: 80-7232-115-3.

[9] HEROUT, P. *Učebnice jazyka Java*. České Budějovice : KOPP, 5., rozš. vyd., 2010. 386 s. ISBN: 978-80-7232-398-2.

[10] HEROUT, P. *Java a XML.* České Budějovice : KOPP, první vydání, 2007. 313 s. ISBN: 978-80-7232-307-4.

[11] HIGHTOWER, R. et al. *Professional Java Tools for Extreme Programming: Ant, XDoclet, JUnit, Cactus, and Maven.* 2004. 768 s. Dostupné z: `https://books.google.cz/books?id=shqNrxbIOBYC`. ISBN : 978-076-4572-74-6.

[12] HYNAR, M. *Java - nástroje.* Praha : Neocortex spol. s r.o., 2004. 325 s. ISBN: 978-80-86330-16-8.

[13] Jiří Peterka. *Báječný svět počítačových sítí, část III. - Síťové architektury* [online]. 2015. [cit. 2016-10-19]. Dostupné z: `http://www.earchiv.cz/b05/b0500001.php3`.

[14] Jiří Peterka. *Rodina protokolů TCP/IP, verze 3.0 - Transportní protokoly* [online]. 2015. [cit. 2016-10-19]. Dostupné z: `http://www.earchiv.cz/l225/slide.php3`.

[15] LAWRENCE, P. – PRAVEEN, M. *Beginning JavaFX.* : Apress, 2010. 336 s. ISBN: 978-1-4302-7199-4.

[16] ORACLE. *Class DatagramSocket* [online]. 2016. [cit. 2017-02-19]. Dostupné z: `https://docs.oracle.com/javase/8/docs/api/java/net/DatagramSocket.html`.

[17] ORACLE. *Package java.net* [online]. 2016. [cit. 2017-02-19]. Dostupné z: `https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html`.

[18] Oracle and/or its affiliates. *JavaFX: Getting Started with JavaFX* [online]. 2014. [cit. 2017-02-22]. Dostupné z: `http://docs.oracle.com/javase/8/javafx/get-started-tutorial/`.

[19] Oracle and/or its affiliates. *Working with the JavaFX Scene Graph* [online]. 2013. [cit. 2017-02-23]. Dostupné z: `http://docs.oracle.com/javafx/2/scenegraph/jfxpub-scenegraph.htm`.

[20] PECINOVSKÝ, R. *Myslíme objektově v jazyku Java.* : Grada Publishing, a.s., 2009. 570 s. ISBN: 978-80-247-2653-3.

[21] Přispěvatelé Diffen.com. *TCP vs UDP* [online]. [cit. 2016-10-19].

[22] Přispěvatelé printertechs.com. *Network Printing Tutorial* [online]. 2017. [cit. 2017-01-07]. Dostupné z: `http://www.printertechs.com/other-instructions/network-printing-tutorial/321-network-printing-tutorial-part-1`.

[23] Ricoh Support. *Spool Printing* [online]. 2008. [cit. 2017-01-07]. Dostupné z: `http://support.ricoh.com/bb_v1oi/pub_e/oi_view/0001036/0001036157/view/printer/unv/0130.htm`.

[24] SCHILDT, H. *mistrovství Java.* : Computer Press, 2014. 1224 s. ISBN: 978-80-251-4145-8.

[25] Sparx Systems Team. *Activity Diagram* [online]. 2017. [cit. 2017-03-12]. Dostupné z: `https://www.sparxsystems.eu/resources/project-development-with-uml-and-ea/activity-diagram/`.

[26] TeamCity Team. *Build Agent* [online]. 2016. [cit. 2016-12-01]. Dostupné z: `https://confluence.jetbrains.com/display/TCD10/Build+Agent`.

[27] TeamCity Team. *Continuous Integration with TeamCity* [online]. 2016. [cit. 2016-12-01]. Dostupné z: `https://confluence.jetbrains.com/display/TCD10/Continuous+Integration+with+TeamCity`.

[28] TeamCity Team. *Getting Started with TeamCity* [online]. 2016. [cit. 2016-12-01]. Dostupné z: `https://confluence.jetbrains.com/display/TCD10/Getting+Started+with+TeamCity`.

[29] VOS JOHAN, C. S. I. D. G. W. – JAMES, W. *Pro JavaFX 8: A Definitive Guide to Building Desktop, Mobile, and Embedded Java Clients.* : Apress, 2014. 616 s. ISBN: 978-1-4302-6574-0.

Příloha A

Seznam použitých zkratek

ANT Another Neat Tool

CI Continuous Integration

CSS Cascading Style Sheets

CSV Comma Separated Values

GUI Graphical User Interface

ISO International Standard Organization

HTTP HyperText Transfer Protocol

IDE Integrated Development Environment

JDK Java Development Kit

JRE Java Runtime Environment

JVM Java Virtual Machine

OOP Object-Oriented Programming

OS Operating System

SDK Software Development Kit

SVN Subversion

TCP Transmission Control Protocol

UDP User Datagram Protocol

URL Uniform Resource Locator

XML eXtensible Markup Language

Příloha B

Konfigurační soubor build.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="ejm" basedir="." default="all">
    <target name="all" depends="bundle-myq-ejm" />

    <taskdef name="bundleapp"
             classname="com.oracle.appbundler.AppBundlerTask"
             classpath="lib/appbundler-1.0.jar" />

    <target name="init">
        <mkdir dir="target" />
    </target>

    <target name="bundle-ejm" depends="init">
        <mkdir dir="target/${short}" />
        <bundleapp outputdirectory="target/${short}"
                   name="${long}"
                   icon="src/main/resources/imgs/${icon}.icns"
                   displayname="${long}"
                   identifier="cz.myq.ejm"
                   mainclassname="cz.myq.ejm.app.Main">
            <classpath file="target/ejm-${short}.jar" />
        </bundleapp>
        <tar destfile="target/${long}.tar.gz"

             compression="gzip">
            <tarfileset dir="target/${short}" filemode="0755">
                <include name="**/MacOS/JavaAppLauncher" />
            </tarfileset>
            <tarfileset dir="target/${short}" >
```

```
                <include name="**/*" />
                <exclude name="**/MacOS/JavaAppLauncher" />
            </tarfileset>
        </tar>

        <delete dir="target/${short}" />
    </target>

    <target name="clean">
        <delete dir="target" />
    </target>

    <target name="bundle-myq-ejm">
        <antcall target="bundle-ejm">
            <param name="short" value="myq" />
            <param name="icon" value="q" />
            <param name="long" value="MyQ Easy Job Manager" />
        </antcall>
    </target>
</project>
```

Příloha C

Instalační a uživatelská příručka

Příručka obsahuje postupně tyto části:

- C.1 Instalaci JRE8 a aplikace EJM.
- C.2 Možnosti spuštění aplikace EJM.
- C.3 Přehled zkratek podporovaných aplikací EJM.

Následující postupy jsou určeny pro OS X 10.8 (Mountain Lion) a vyšší.

C.1 Instalace

1. Ověřte, zda máte nainstalovanou Javu SE Runtime Environment 8 update 111:

```
$ java -version
  java version "1.8.0_122-ea"
  Java(TM) SE Runtime Environment (build 1.8.0_122-ea-b04)
  Java HotSpot(TM) 64-Bit Server VM (build 25.122-b04, mixed mode)
```

2. (a) Pokud ano, pokračujte bodem 12.

(b) Pokud ne, stáhněte[1] z následujícího odkazu soubor `.dmg`: `http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`

3. Dvojklikem na stažený soubor otevřete balíček `.dmg`.

4. Dvojklikem na ikonu `.app` souboru spusťte instalačního průvodce. Budete požádáni o zadání přihlašovacích údajů privilegovaného uživatele[2].

5. Vyplňte požadované údaje a potvrďte.

6. Je zobrazeno úvodní okno, pokračujte volbou *Next*.

7. Je zobrazena nabídka programů partnerských společností Oracle, pokračujte volbou *Next*.

8. Je zahájena instalace a zobrazeno okno s jejím průběhem.

9. Potvrďte dialog, který byl zobrazen po dokončení instalace.

10. Je spuštěn applet, který zkontroluje nainstalovanou verzi Javy.

11. Instalace Javy byla úspěšně dokončena, smažte `.dmg` soubor.

12. Rozbalte soubor (archiv) MyQ Easy Job Manager.tar.gz příkazem v aplikaci Terminal:

```
$ tar -zxvf MyQ\ Easy\ Job\ Manager.tar.gz
```

13. Obsah rozbaleného archivu je spustitelný soubor `.app`. Ten nakopírujte do adresáře, odkud budete chtít aplikaci EJM spouštět.

14. Instalace aplikace EJM byla úspěšně dokončena. Pro úsporu místa smažte soubor `.tar.gz`.

[1] Před stažením je nutné potvrdit souhlas s licenčními podmínkami.

[2] Důvodem je, že instalace se netýká pouze jednoho uživatele, ovlivní všechny uživatelské účty.

C.2 Spuštění aplikace

Možnosti, jak spustit aplikaci EJM jsou dvě:

1. Dvojklikem na ikonu spoustitelného souboru `MyQ Easy Job Manager.app`.

2. Z příkazové řádky spuštěním `.jar` souboru.

   ```
   $ java -jar MyQ\ Easy\ Job\ Manager.app/Contents/Java/ejm-myq.jar
   ```

   Aplikaci je možné spustit bez argumentů, jak je uvedeno výše. V takovém případě bude po spuštění vyžadováno zadání přihlašovacích údajů privilegovaného uživatele a následně otevřeno okno s nastavením.
   Při spuštění aplikace je možné předat tři argumenty: IP adresu serveru, číslo portu, boolean hodnotu true/false (podle toho, zda je povoleno zabezpečené připojení).

   ```
   $ java -jar MyQ\ Easy\ Job\ Manager.app/Contents/Java/ejm-myq.jar
   10.15.6.148 91 true
   ```

   Předáním těchto parametrů dojde k nastavení automaticky již při spuštění a nebude vyžadováno po uživateli po startu aplikace.

Aplikace je spuštěna jako TrayApp s ikonou v notifikačním panelu operačního systému.

C.3 Podporované zkratky

Akce	**Zkratka**
Nastavení	CMD + ,
Zavřít	CMD + W
Ukončit	CMD + Q
Pole pro rychlé hledání v menu	menu bar > help > search

Tabulka C.1: Přehled podporovaných zkratek

Příloha D

Obsah přiloženého CD

```
.
├── dist                                - adresář s aplikací pro Mac OS
│   └── MyQ Easy Job Manager.tar.gz     - aplikace pro Mac OS
├── README                              - popis jednotlivých adresářů a souborů
└── text                                - text DP
    ├── doubkova.pdf                    - text DP ve formátu pdf
    └── latex                           - text DP ve formátu latex
```

Obrázek D.1: Obsah přiloženého CD