České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jan Kubíček**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Nasazení a údržba restauračního systému**

Pokyny pro vypracování:

Navrhněte, naprogramujte, nasaďte, otestujte a dokumentujte mechanismy, které budou zajišťovat možnost instalace a upgrade existujícího restauračního systému pouze oprávněným uživatelům. Dále dle pokynů zadavatele odstraňte chyby v systému, které znemožňují jeho produkční nasazení. Vývoj provádějte iterativně.

Seznam odborné literatury:

[1] LARMAN, Craig a Chris RUPP. Applying UML and patterns: introduction to object-oriented analysis and design and iteractive development. 3rd ed. New Jersey: Prentice-Hall, 2005, xviii, ISBN 01-314-8906-2.
[2] FOWLER, Martin.Destilované UML. Praha: Grada, 2009, 173 s.ISBN 978-80-247-2062-3.

Vedoucí: Ing. Martin Komárek

Platnost zadání: do konce letního semestru 2014/2015

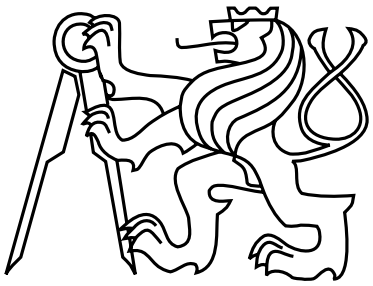
doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 28. 2. 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Nasazení a údržba restauračního systému

Jan Kubíček

Vedoucí práce: Ing. Martin Komárek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

26. května 2014

Poděkování

Chtěl bych poděkovat mé rodině za veškerou podporu při studiu, panu Ing. Martinu Komárkovi za vedení této bakalářské práce a ostatním členům týmu, kteří pracovali na projektu CashBob. Rád bych také poděkoval všem lidem, kteří mě přímo či nepřímo podporovali a pomáhali během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 23. 5. 2014 .....

Abstract

The main goal of this bachelor thesis is the analysis, design and implementation of the application CashBob deployment and preventing unauthorized access to the application. As a part of the deployment is a build solution, simple installation and running application. Further, the licensing system is designed to prevent unauthorized use of the application.

Abstrakt

Hlavním cílem bakalářské práce je analýza, návrh a implementace nasazení aplikace CashBob a zamezení neoprávněnému používání aplikace. V rámci nasazení je řešeno sestavení aplikace, jednoduchá instalace a spuštění aplikace. Dále je řešen systém licencování, který zamezí neoprávněnému používání aplikace.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

S informačními technologiemi se v dnešní době setkáváme téměř na každém kroku a v budoucnu se budeme určitě setkávat čím dál více. Vývojáři vymýšlejí nové aplikace, systémy a technologie jak k usnadnění práce, tak i pro zábavu. Jednou z těchto aplikací zaměřující se hlavně na usnadnění práce v restauračních zařízení je i aplikace CashBob [2].

Aplikace CashBob se vyvíjí zhruba 5 let a pracovalo na ní mnoho studentů (vývojářů) v rámci svých semestrálních projektů a bakalářských prací, kteří přidávali funkcionalitu nebo opravovali chyby předchozích vývojářů. Během této doby se z CashBobu stala rozsáhlá aplikace starající se o komplexní správu celého podniku.

CashBob obsahuje několik modulů pro usnadnění správy podniku. Každý modul se zaměřuje na odlišnou část správy podniku. CashBob obsahuje moduly zajišťující funkčnost pokladny, správy zaměstnanců, zboží, stolů a menu, evidence zboží na skladě a plánování směn. Některé moduly jsou navrhnuté pro dotykové terminály, které by měli zrychlit a zjednodušit práci obsluhy podniku. CashBob nabízí také aplikaci pro mobilní telefony, která je určena hlavně zákazníkům restaurací, pomocí které získají uživatelé přehled o novinkách restaurace, jídelním lístku, nabízených produktech, akcích atd.

CashBob je vyvíjen v jazyce Java a je založen na síťové architektuře klient-server. Serverová část aplikace je čistě funkční bez uživatelského rozhraní a slouží především k provádění aplikační logiky a komunikaci s databází. Klientská část poskytuje grafické uživatelské rozhraní a slouží ke komunikaci se serverovou částí aplikace prostřednictvím síťového rozhraní.

Cílem mé bakalářské práce je nasazení aplikace CashBob do reálného provozu viz kapitola 3 a zamezit její neoprávněné používaní viz kapitola 2. Způsob, jakým lze zamezit neoprávněné používání aplikace, je vytvoření licenčního systému. Nasazení se týká jak správného sestavení aplikace, tak i vytvoření instalačních balíčků, aby uživatel, který si aplikaci CashBob pořídí, ji dokázal co nejjednodušeji nainstalovat a spustit.

Kapitola 2

Licencování

Licencování [30] je způsob, jak pořídit software a legálně ho používat a mít k němu podporu. Licencování je také způsob ochrany softwaru před pirátství nebo alespoň jeho ztížení. Na licencování lze nahlížet ze dvou různých pohledů. První pohled je z právního hlediska a druhý pohled je z hlediska implementace.

Nejdůležitější věcí z právního hlediska je, že software vlastně není prodáván, ale je poskytována pouze licence. Licence je v zásadě souhlas autora, který vás opravňuje používat jeho duševní vlastnictví (software) v souladu s určitými omezeními, která jsou stanovena v licenční smlouvě. I když si software koupíte, stále je majetkem společnosti (vývojáře, vývojářů), která jej vyvinula.

Z hlediska implemenentace jde o vyvinutí subsystému, který umožní užívat software až po získání licence. Principy, jak vytvořit tento subsystém a tím zabezpečit software proti neautorizovanému používání, se budu zabývat v následujících sekcích. Jelikož v aplikaci CashBob subsystém licencování nebyl zatím nijak řešen, analyzuji jednotlivé způsoby, jak správného licencování docílit, navrhnu a zrealizuji jeho řešení.

2.1 Analýza

Tato kapitola se věnuje sběrem podkladů, jak správně ochránit software licencováním.

2.1.1 Licence

V tomto kontextu se budeme bavit o licenci jako o instanci, která obsahuje informace potřebné k ověření platnosti a která je nějakým způsobem uložená např. jako soubor v počítači, jako záznam v databázi, atd. Co je vhodné, aby licence obsahovala, jaké typy licencí se většinou používají a jejich základní rozdělení. Jelikož licence má většinou nějaké omezení, podle kterého se určuje, jestli je licence platná nebo není, je potřeba ji zabezpečit, aby nešlo toto omezení neautorizovaně modifikovat.

2.1.1.1 Parametry licence

Parametry nám blíže specifikují licenci. Podle parametrů licence budeme ověřovat, zda je licence platná nebo neplatná. Je to pouze na nás, jaké parametry budeme chtít ukládat

v licenci. Ale přeci jen zde uvedu výčet některých parametrů, které se v licenci většinou objevují [24].

- Unikátní identifikátor (licenční klíč) - Viz sekce 2.1.2
- Vlastník licence - Název společnosti/osoby, která si licenci pořídila
- Předmět licence - K čemu/jakému produktu se licence váže
- Vydavatel licence - Název společnosti, která tuto licenci vydala
- Vznik licence - Datum vzniku licence
- Platnost od - Od jakého data licence platí
- Platnost do - Do jakého data licence platí
- Počet licencí - Použití u floating licence[1]
- Další parametry - verze, typ, atd.

2.1.1.2 Typy licencí

Typů licencí [1] je celá řada. U typů licence jde většinou o nějaká omezení. Proto se zde pokusím uvést základní typy omezení, které se u licencí většinou vyskytují. Je pouze na nás jaké omezení využijeme. Můžeme si vymyslet jakékoliv omezení, které potřebujeme.

1. Omezení koupí/registrací licence
   - Placená - Licenci si musíme zaplatit, abychom mohli používat software.
   - Registrovaná - Abychom získali licenci, tak se musíme zaregistrovat.
   - Volná - Zpřístupňuje plně funkční software bez poplatků.
2. Omezení na počet počítačů - Tento typ omezení licence můžeme ještě rozdělit na používání softwaru pouze pro jednoho uživatele nebo současné užívání více uživateli.
   - Individuální - Software může být použit pouze na jednom počítači. Aby tomu tak bylo, musíme si zjistit informace o počítači 2.1.3.
   - Určitý počet - Licence umožňuje nainstalovat software na určitý počet počítačů.
   - Určitá lokace - Licence poskytuje přístup k softwaru v určité lokalitě např. firma, škola.
   - Neomezená - Software může používat neomezený počet počítačů.
3. Další omezení - Časové omezení licence (Uživatel může používat software pouze po určitou dobu), atd.

[1] licenci lze souběžně spustit z libovolného počtu počítačů, nejvýše však do počtu vámi zakoupených licencí

2.1.1.3 Zabezpečení licence

Jelikož licence obsahuje některé informace, které by uživatel neměl mít možnost přečíst natož měnit, je potřeba licenci zabezpečit. Zabezpečením licence se zamezí neautorizovanému šíření licence, prodloužení platnosti, atd.

Šifrování licence Aby uživatel neměl možnost získávat informace z licence, je ji potřeba zašifrovat. Šifrování [25] je druh kryptografie zabezpečující data. Data se převedou do nesrozumitelné formy, a pokud neznáme klíč, nesestavíme původní význam dat.

Elektronický podpis licence Zamezení změny informací v licence zajistíme elektronickým podpisem. Elektronický podpis v informatice zastupuje vlastnoruční podpis při ověřování pravosti dat. Elektronický podpis je aplikací asymetrické kryptografie [28].

- Princip vzniku elektronického podpisu
  1. Výpočet otisku dat - Je vypočítán hash z dat, což je poměrně kratké číslo, typicky několik stovek bitů.
  2. Zašifrování hashe - Hash je zašifrován autorovým privátním klíčem, čímž vznikne elektronický podpis.
- Princip ověření pravosti dat
  1. Výpočet otisku dat - Z dat vypočítáme hodnotu hashe.
  2. Dešifrování elektronického podpisu - Pomocí veřejného klíče autora podpisu je dešifrován obsah elektronického podpisu.
  3. Porovnání výsledků - Výsledek vypočteného otisku dat je porovnán s výsledkem dešifrovaného podpisu.Pokud se výsledky shodují data nebyla změněna.

V našem případě útočník nemůže nijak licenci změnit, ani zašifrovanou, ani kdyby se mu podařilo licenci dešifrovat. Kdyby chtěl obsah licence změnit, musel by změnit i elektronický podpis a musel by tak znát autorův privátní klíč. Útočník zná autorův veřejný klíč, ze kterého lze získat privátní klíč, ale v současnosti nebyl nalezen žádný algoritmus, který by to dokázal v rozumném čase. Je tedy téměř nemožné získat privátní klíč a tím i vytvořit platný elektronický podpis.

Důvěryhodnost platného elektronického podpisu je nutné zjistit pomocí principů přenosu důvěry z třetí strany. K tomuto účelu je využíván digitální certifikát.

2.1.2 Unikátní identifikátor licence

Aby nešlo licenci neautorizovaně šířit je potřeba unikátní identifikátor licence. V případě licencí se o unikátním identifikátoru mluví jako licenčním klíči, produkčním klíči nebo seriovém čísle. Licenční klíč přesně specifikuje určitou instanci licence. Používá se k ověření pravosti a platnosti licence. Unikátní identifikátor [31] je jakýkoliv identifikátor, u kterého je zaručeno, že je jedinečný mezi všemi identifikátory použitých pro konkrétní objekty, v našem případě pro licence.

2.1.3 Informace o počítači

Informace o počítači potřebujeme zjistit v případě, pokud chceme aby licence šla použít pouze pro jeden počítač. Pokud budeme chtít používat floating licenci, informaci o počítači vůbec zjišťovat nemusíme. Pro získání těchto informací se používájí většinou sériová čísla počítačového hardwaru nebo MAC adresa [27] síťove karty, jelikož by tyto informace měli být unikátní. To znamená, že by na světě neměla existovat dvě stejná seriová čísla nebo dvě stejné MAC adresy. Jelikož MAC adresy se dají změnit v OS, tak to není úplně pravda.

2.1.4 Získání a ověřování licence

Získat a ověřit licenci lze dvěma způsoby. A to licencovat offline nebo online [21].

2.1.4.1 Online licencování

Online licencování znamená, že software může přistupovat k internetu. V tomto případě probíhá komunikace mezi klientskou aplikací a licenčním serverem 2.1.5.

Postup komunikace mezi klientem, aplikací a licenčním serverem

1. Prodejce produktu obdrží a ověří požadavek na získání licence od klienta a nastaví tento požadavek na licenční server. Každá licence by měla mít nějaký unikátní identifikátor, který definuje prodejce.
2. Unikatní identifikátor je poslán klientovi.
3. Klient do nainstalované aplikace zadá unikátní identifikátor a tím vytvoří licenci.
4. Knihovna, která se stará o licenci, kontaktuje licenční server a ten zjistí, jestli je tato licence platná pod tímto identifikátorem.
5. Aplikace může zjistit informace o počítači, na kterém aplikace běží, a tím omezit používání softwaru pouze na tento jeden počítač.
6. Aplikace
   (a) uloží licenci s parametry na file systém do zašifrovaného souboru s elektronickým podpisem.
   (b) pošle licenci a její parametry na licenční server .
7. Při každém spuštění aplikace
   (a) načte zašifrovaný soubor z file systému a ověří licenci.
   (b) pošle požadavek k přístupu do aplikace na licenční server, který licenci ověří.

Komunikace mezi aplikací a webovým serverem by měla probíhat přes webovou službu např. REST. Aby klient získal unikátní identifikátor, je možné použít email, ale asi by bylo lepší, aby i toto probíhalo přes webovou službu. Místo unikátního identifikátoru můžeme rovnou poslat licenci nebo soubor s licencí. Výhodou je, že logika vytváření licence je na

licenčním serveru. Pokud chceme licenci uzamknout jenom na jeden počítač, je potřeba zjistit některé unikátní informace o počítači. Ale je zde i možnost, že se nebudeme zajímat o to na jakém počítači aplikace poběží. Bude nás zajímat, kolik instancí licence právě běží. Jelikož aplikace může stále komunikovat s licenčním serverem, bude pro prodejce jednodušší získat informace z aplikace a sledovat tak určité statistiky.

Výhody

- Jednodušší správa vydaných licencí a jejich vlastníků
- Těžší obejití ochrany licencováním, protože se většina operací k vytvoření licence provadí na licenčním serveru
- Možnost floating licencování

Nevýhody

- Nutný přístup k internetu

2.1.4.2 Offline licencování

Offline licencování znamená, že software nemůže přistupovat k internetu. Aplikace nemůže komunikovat s licenčním serverem, takže klient musí komunikovat většinou prostřednictvím nějakého webu s licenčním serverem, nebo nekomunikuje vůbec.

- **Klient má přístup na internet z jiného počítače**

  Postup komunikace mezi klientem a licenčním serverem

  1. Klient si nechá vygenerovat aplikací unikátní kód z informací o počítači a odešle ho na licenční server.
  2. Licenční server vygeneruje z unikátního kódu unikátní identifikátor a odešle ho klientovi např. emailem.
  3. Klient zadá tento identifikátor do aplikace.
  4. Aplikace vygeneruje licenci s parametry a uloží ji do file systému do zašifrovaného a elektronicky podepsaného souboru.
  5. Při každém spuštění aplikace načte zašifrovaný soubor z file systému a ověří licenci.

  Místo unikátního identifikátoru můžeme rovnou poslat soubor s licencí. Výhodou je, že logika vytváření licence je na licenčním serveru. Nevýhodou je, že klient bude muset zkopírovat soubor na přesně určené místo. Unikátní kód je potřeba, aby nešlo šířit volně aplikaci mezi více počítači. Offline licencování lze udělat také přes proxy server, který je zabudován v aplikaci a slouží jako licenční server. Klient si stáhne aplikaci i s proxy serverem a není poté potřeba posílat unikátní identifikátor přes email. Je však potřeba zadat nějaké unikatní informace při stahování aplikace.

Výhody

- Aplikace nemusí mít přístup k internetu

Nevýhody

- Nemožnost floating licence

- **Klient nemá vůbec přístup na internet**
  V takovémto případě můžou nastat dvě možnosti. Klient si koupí aplikaci již s licenčním klíčem např. napsaným na obalu CD a po zadání toho klíče se mu licence vytvoří. Nebo se mu při spuštění aplikace vytvoří licence, většinou nějak časově omezená.

  Výhody

  - Klient nemusí mít přístup k internetu

  Nevýhody

  - Žádná správa vydaných licencí a jejich vlastníků
  - Snažší obejití ochrany licencováním

2.1.5 Licenční server

Licenční server [26] je centralizovaný softwarový systém, který poskytuje přístupové údaje, licence, licenční klíče atd. klientům s cílem používat licencovaný software na svých počítačích. Funkčnost licenčního serveru se liší v závislosti online nebo offline licencování. Licenční server může ukládat informace o vydaných licencí a jejich vlastnících k potřebám různých statistik nebo pro znovuzaslání licence.

2.2 Návrh

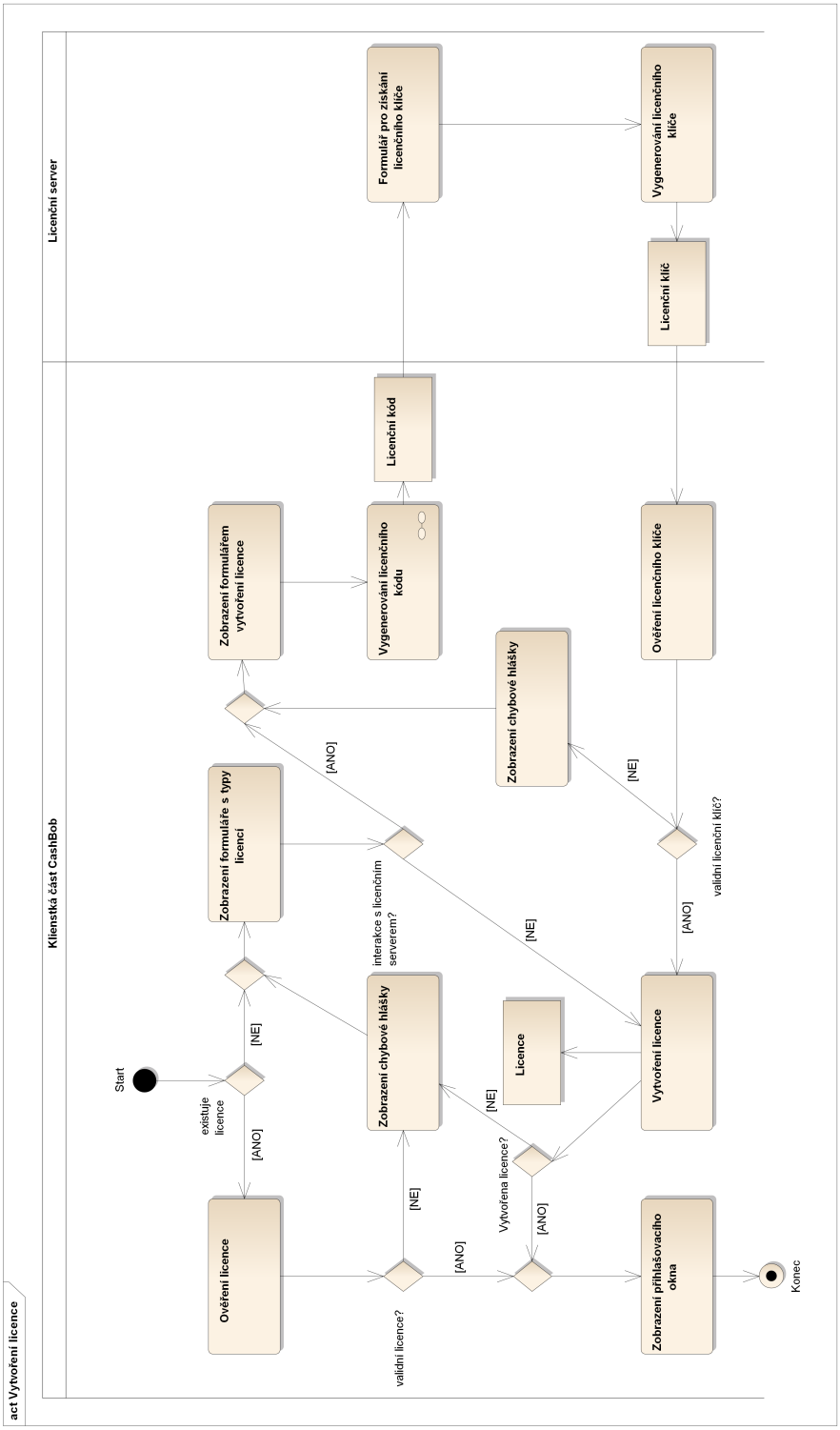
V této části uvedu návrh řešení licencování softwaru z podkladů, které jsem uvedl v předchozí části 2.1.

Jelikož se předpokládá, že ne všude, kde se aplikace CashBob bude používat, bude přístup k internetu, vybrali jsme způsob licencování offline. Dále se předpokládá, že klient bude mít přístup k internetu, aby mohl komunikovat s licenčním serverem. Proto jsme zvolili variantu licencování offline s přístupem klienta na internet z jiného počítače s individuálním typem licence.

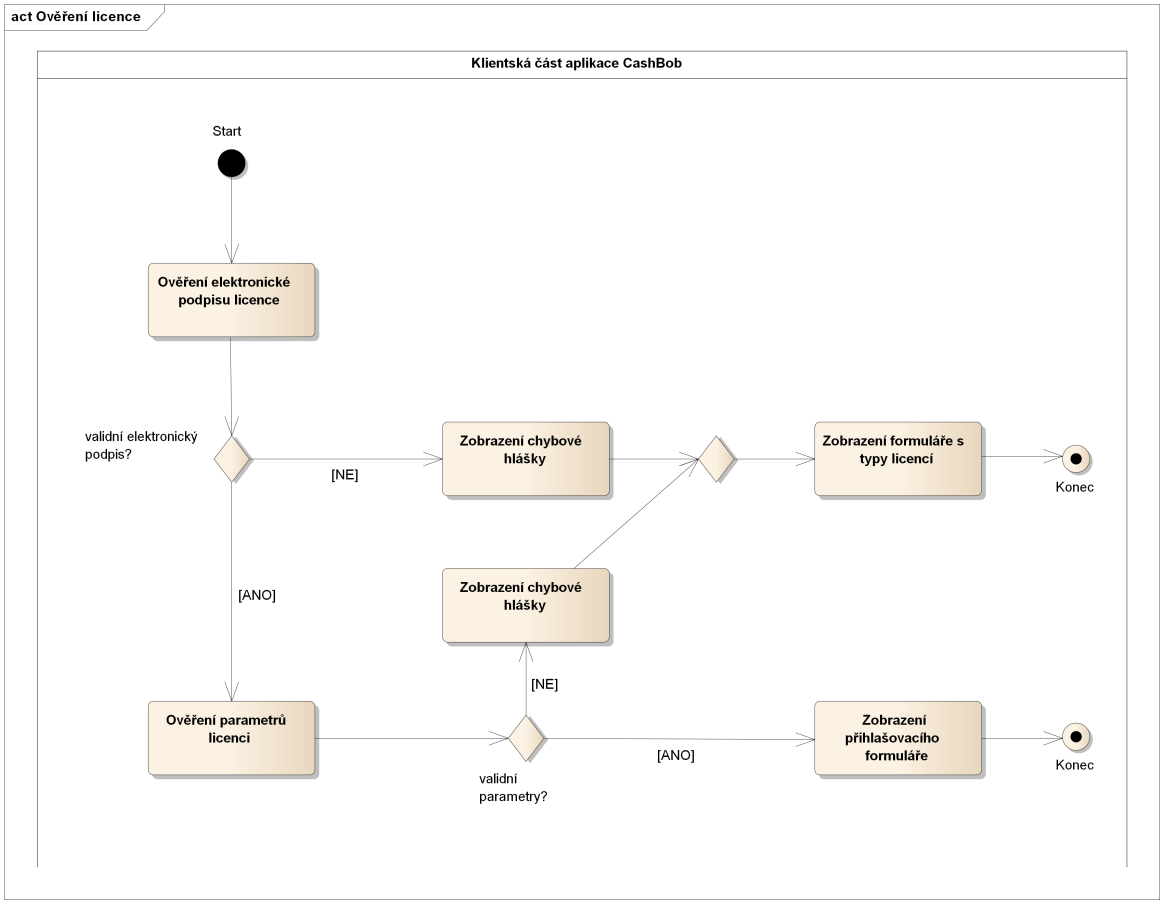
Také zde navrhnu objekty a postupy, které budou potřeba ke správné funkci licencování. Dále popíši, jak bude probíhat interakce mezi uživatelem a systémem.

2.2.1 Proces vytvoření licence

- Start - Spustíme aplikaci.
- Existuje licence? - Existuje soubor s licencí?
- Ověření licence, Validní licence? - Viz sekce 2.2.7.4.
- Zobrazení formuláře s typy licencí - Zobrazí se formulář, kde si uživatel vybere jaký typ licence chce.
- Interakce s licenčním serverem? - Podle vybraného typu licence se rozhodujeme dále, co bude následovat.
- Zobrazení formuláře vytvoření licence - Zobrazí se formulář, kde si uživatel může vygenerovat licenční kód a zadat licenční klíč.
- Vygenerování licenčního kódu, Licenční kód - Viz sekce 2.2.4.
- Formulář pro získání licenčního klíče - Zobrazí se formulář s položkami potřebnými k vygenerovaní licenčního klíče.
- Vygenerování licenčního klíče, Licenční klíč - Viz sekce 2.2.5.
- Ověření licenčního klíče, Validní licenční klíč? - Viz sekce 2.2.6.
- Vytvoření licence, Licence - Viz sekce 2.2.7.3.
- Vytvořena licence? - Je vytvořen soubor, kde je uložena licence?
- Zobrazení přihlašovacího okna - Zobrazí se přihlašovací okno a aplikace běží dále.
- Zobrazení chybové hlášky - Aplikace zobrazí dialog s chybovou hláškou.



Obrázek 2.1: Proces vytvoření licence



Obrázek 2.2: Proces ověření licence

2.2.2 Proces ověření licence

- Start - Licenci i s pomocnými soubory máme uloženou v filesystému a aplikace ji je schopna najít.
- Ověření elektronického podpisu, Validní elektronický podpis? - Viz sekce 2.1.1.3.
- Ověření parametrů licence, Validní parametry? - Viz sekce 2.2.7.4.
- Zobrazení chybové hlášky - Aplikace zobrazí dialog s chybovou hláškou.
- Zobrazení formuláře s typy licencí - Zobrazí se formulář, kde si uživatel vybere jaký typ licence chce.
- Zobrazení přihlašovacího okna - Zobrazí se přihlašovací okno a aplikace běží dále.

2.2.3 Entropie

Entropii, jinak řečeno míru neurčitosti, potřebujeme k zjištění informací o počítači, na kterém aplikace CashBob běží. Z entropie se generuje licenční klíč 2.2.5. Podle entropie se

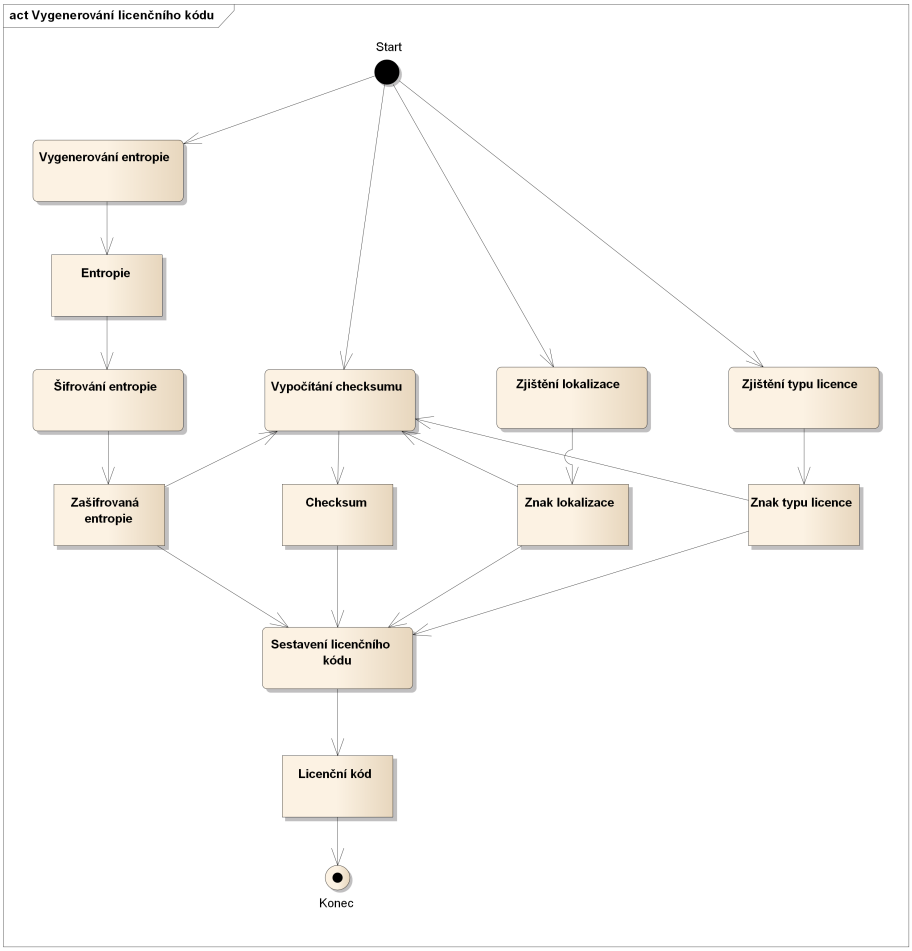
také ověřuje platnost licence.

Entropii získáme tak, že ze síťového rozhraní počítače získáme MAC adresu a tu zahashujeme. Entropie bude mít hodnotu pole bytů. Jelikož od JDK 1.6 má Java přístup k informacím síťové karty prostřednictvím třídy *NetworkInterface*, nebude problém zjistit hodnotu MAC adresy. Nevýhodou získání entropie podle MAC adresy je, že uživatel si může změnit MAC adresu síťového rozhraní. Avšak většina uživatelů tuto možnost nepoužívá. Pokud by uživatel chtěl změnit hodnotu MAC adresy a tím chtěl obejít ochranu licencováním, předpokládá se, že to ve většině případů bude v rámci jedné sítě, kde budou všechny počítače připojené k jednomu switchi, musel by mít pokročilé znalosti v síťování, protože by se mu objevili v rámci jedné sítě dvě stejné MAC adresy a síť by přestala správně fungovat. Bylo by lepší generovat entropii např. podle sériového čísla základní karty, ale zde je problém, že získání sériového čísla karty probíhá na každém OS jinak.

2.2.4 Licenční kód

Licenční kód je potřeba ke generování licenčního klíče, protože nese informaci o entropii a typu licence. Licenční kód uživatel používá ke komunikaci s licenčním serverem 2.2.8. Licenční kód je generován z entropie. Je to vlastně zašifrovaná entropie s dalšími pomocnými parametry.

- Vygenerování entropie, Entropie - Viz sekce 2.2.3.
- Šifrování entropie - Jelikož entropie je pole bytů, je potřeba nějakým způsobem převést toto pole do uživatelsky přívětivého formátu nejlépe do řetězce znaků.
- Zašifrovaná entropie - Výstup získáný ze šifrování entropie.
- Zjištění lokalizace - Zjištění zvoleného jazyka v aplikaci z konfiguračního souboru.
- Znaky lokalizace - Jsou to znaky, které určují nastavení jazyka v aplikaci. Tyto znaky se používá ke statistickým údajům.
- Zjištění typu licence - Zjistí se, jaký typ licence si uživatel vybral.
- Znak typu licence - Je to znak, který určuje vybraný typ licence. Licenční server podle tohoto znaku pozná typ licence a tedy jaký licenční klíč má vygenerovat.
- Vypočítání checksumu - Checksum se počítá jako suma hodnot znaků zašifrované entropie.
- Checksum - Hodnota, která se používá ke kontrole licenčního kódu, jestli nedošlo k překlepu při zadávání licenčního kódu na licenčním serveru.
- Sestavení licenčního kódu - Složení řetězců a znaků v tomto pořadí: zašifrovaná entropie, znaky lokalizace, znak typu licence a checksum.
- Licenční kód - Řetězec znaků, vytvořený během sestavení licenčního kódu.



Obrázek 2.3: Proces generování licenčního kódu

2.2.5 Licenční klíč

Účel licenčního klíče je zabránit kopírování, sdílení nebo jinak nelegálního používání softwaru nelicencovaným uživatelům. Aby mohla být v aplikaci vytvořena licence, musí být zadán validní licenční klíč. Licenční klíč generuje licenční server z daného licenčního kódu. Licenční klíč slouží také k ověření licence. Licenční klíč je vyjádřen řetězcem znaků.

2.2.6 Generování a validace licenčního klíče

Jelikož tato problematika je složitá a stojí na ní skoro celá bezpečnost licencování, začal jsem hledat možná řešení na internetu. Jako nejlepší řešení mně přislo generování a validace licenčního klíče po částech. Princip tohoto řešení jsem našel na adrese [18].

Ve stručnosti popíši, jak tento princip generování a validace licenčního klíče funguje.

- Generování - Generovaný klíč se skládá ze tří částí, ze seedu, klíčových bytů a checksumu. Seed je číslo zvolené libovolně, klíčové byty získáme bitovými operacemi volitelných

parametrů a bytů entropie se seedem a checksum je kontrolní součet předešlých částí. Složením všech těchto částí zapsaných v hexadecimálním tvaru nám vznikne licenční klíč.

- Validace - Validují se všechny části, které jsem zmínil v části generování. Při validaci známe licenční klíč, volitelné parametry a byty entropie. Z licenčního klíče zjistíme seed, klíčové byty a checksum. Nejprve ověříme checksum, poté ověříme seed naproti blacklistu seedů, pak ověříme, jestli souhlasí volitelné parametry, a nakonec ověříme správnost entropie.

Hodnoty volitelných parametrů musí být stejné v generátoru i ve validátoru licenčního klíče. Výhody toho algoritmu jsou, že pokud by někdo zveřejnil svůj licenční kíč, můžeme jeho seed dát do blacklistu[2]. Pokud by útočník dokázal vytvořit keygen[3], stačí pouze změnit hodnoty volitelných parametrů a keygen nebude fungovat v nové verzi aplikace.

V licenčním klíči bude zakódováno datum, do kdy bude licence platná.

2.2.7 Licence

Aby uživatel při každém startu aplikace nemusel zadávat licenční klíč bude vytvořena licence. Tato licence bude uložena v zabezpečeném souboru. Podle vybraného typu bude licence vytvořena buď po zadání validního licenčního klíče, nebo rovnou. Ověření licence bude probíhat při každém startu aplikace.

2.2.7.1 Parametry licence

Při ověřovaní licence budeme ověřovat platnost parametrů. Bude nás zajímat o jaký typ licence se jedná, do kdy je její platnost a jestli je licence vůbec platná.

- Platnost do - Časový údaj, do kdy je licence platná.
- Licenční klíč - Viz sekce 2.2.5.
- Typ licence - Údaj, který určuje typ licence.

2.2.7.2 Typy licence

Navrženy jsou 3 typy licencí. Podle typu licence budeme, nebo nebudeme muset zadat licenční klíč a licence budou buď zdarma, nebo placené. Všechny typy licencí budou časově omezené. Licenční klíč získáme na licenčním serveru.

- TRIAL - Tento typ licence je tzv. zkušební. Licence je zdarma a nepotřebujeme zadávat licenční klíč, takže nebude potřeba komunikace s licenčním serverem. Tato licence bude mít nejkratší dobu trvaní a bude s nejvíce omezeními funkčnosti aplikace. Licence bude moci být vytvořena jenom jednou a po uplynutí platnosti nebude možnost vytvořít další licenci tohoto typu.

[2] seznam obsahující něco zakázaného
[3] Key Generator je program, který generuje licenční klíče

- FREE - Licence je zdarma, ale pro vytvoření licence bude potřeba zadat licenční klíč. Tato licence bude mít omezení funkčnosti aplikace, ale ne tak rozsáhlá jako typ TRIAL.
- FULL - Tato licence bude placená a při vytvoření licence bude potřeba zadat licenční klíč. Licence bude bez omezení funkčnosti aplikace.

Proč máme FREE licenci, proč nestačí TRIAL? Výhoda FREE licence je, že uživatel musí komunikovat s licenčním serverem a tím pádem můžeme vést statistiky.

2.2.7.3 Vytvoření licence

Vytvořit licenci znamená nastavit ji parametry a uložit ji do souboru.

2.2.7.4 Ověření licence

Ověření licence probíhá tak, že ověříme správnost a platnost parametrů licence. Musí správně proběhnout validace licenčního klíče a časový údaj platnosti licence musí být menší než současný čas.

2.2.7.5 Zabezpečení licence

Soubor s licencí bude zašifrován a elektronicky podepsán.

2.2.8 Licenční server

Jediná funkcionalita licenčního serveru je generování licenčního klíče z licenčního kódu, který uživatel zadá do formuláře. Po vygenerování licenčního klíče je tento klíč poslán na uvedený email.

2.2.9 Případy užití licencování

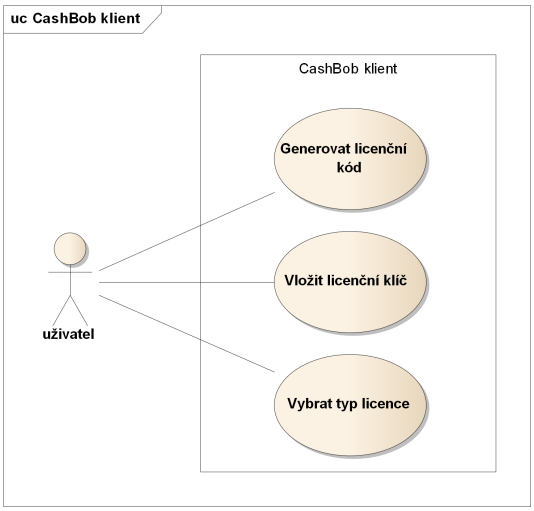
Jelikož aplikace nedokáže sama zajistit získání licence, je potřeba, aby uživatel provedl několik kroků k získání licence. V této sekci ukážu, jak bude probíhat interakce mezi uživatelem a aplikací.

2.2.9.1 Případy užití na klientské části

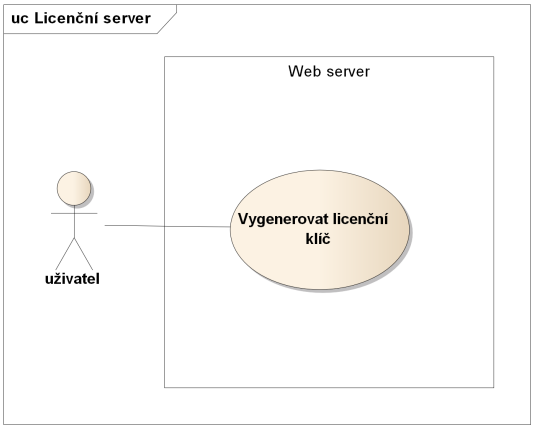
- Generovat licenční kód - Uživatel si nechá aplikací vygenerovat licenční kód.
- Vložit licenční klíč - Uživatel může vložit licenční klíč. Jestliže je licenční klíč validní, aplikace vytvoří licenci.
- Vybrat typ licence - Uživatel si může vybrat, jaký typ licence chce.

2.2.9.2 Případy užití na licenčním serveru

- Vygenerovat licenční klíč - Uživatel si může nechat vygenerovat licenční klíč po zadání povinných údajů.



Obrázek 2.4: Případy užití na klientské části



Obrázek 2.5: Případy užití na licenčním serveru

2.3 Realizace

V této kapitole je popsána implementace licencování a použití jednotlivých knihoven. Implementace má základ v návrhové části 2.2, které se snaží držet, nebo v případě nalezení nových poznatků najít ideální řešení.

Jelikož celé licencování závisí na licenčním klíči a licenci, v následujících sekcích popíšu realizaci těchto objektů a postupy, jak k ním dojít a jak s nimi pracovat.

2.3.1 Entropie

Jak jsem se zmínil v návrhu 2.2.3, abychom zjistili entropii, musíme znát MAC adresu síťového rozhraní počítače. Tu zjistíme prostřednictvím třídy *NetworkInterface* z balíčku *java.net*. Jelikož z entropie se generuje a validuje licenční klíč, musíme přidat parametr, který určuje entropii typu licence. Kdybychom tam tento parametr nepřidali, všechny typy licencí, které potřebují zadat licenční klíč, by měli tento klíč stejný. Entropie vznikne zahashováním MAC adresy a typu licence. Jako hashovací funkci používáme SHA-512. Vznikne nám pole 64 bytů. Kdybychom takto dlouhou entropii zašifrovali, vznikl by nám licenční kód o délce větší než 100 znaků. Proto selekcí prvků pole vytvoříme výslednou entropii o velikosti 15 bytů. Počet všech možností entropie je $n = 256^{15}$, a tedy pravděpodobnost, že 2 entropie budou stejné je $P = 1/n \doteq 7,5 \times 10^{-37}$. Jak je vidět z výsledku, je téměř nulová pravděpodobnost, že 2 různí uživatelé budou mít stejnou entropii.

2.3.2 Licenční kód

Jelikož k vytvoření licenčního klíče potřebujeme znát entropii, musíme nějak tuto entropii zjistit. Licenční klíč generuje licenční server, takže mu nějakým způsobem musíme dodat entropii, přesně řečeno uživatel mu ji musí dodat. Proto je důležitý licenční kód. Licenční kód je zašifrovaná entropie do uživatelsky přívětivé podoby. V licenčním kódu je také zašifrovaný typ licence, lokalizace a checksum.

- Zašifrovaná entropie - Zašifrovaná entropie vznike šifrováním entropie pomocí Base32[4] typu Crockford [22]. Z entropie vznikne řetězec 24 znaků.
- Lokalizace - Do licenčního kódu přidáme 2 znaky lokalizace. Slouží pouze ke statistickým účelům.
- Typ licence - Součástí licenčního kódu bude 1 znak, který určuje typ licence. Díky tomuto znaku bude vědět licenční server o jaký typ licence se jedná.
- Checksum - Checksum získáme sumarizací hodnot všech znaků licenčního kódu a převedením do hexadecimální soustavy. Tento údaj slouží k validaci licenčního kódu, jestli nedošlo k překlepu.

Licenční kód je tedy řetězec 30 znaků.

[4]Base32 je šifrování, kdy převádíme 5 bitů na 1 znak.

2.3.3 Licenční klíč

Generování a validace licenčního klíče realizujeme podle návrhu 2.2.5. Inspiroval jsem se implementací na adrese [20].

Uvedu zde jednotlivé části licenčního klíče. Všechny části jsou zapsány v hexadecimálním tvaru.

- Seed - Seed je číslo typu integer a získáme ho z UNIX Time[5]. Seed je v klíči zastoupen 8 znaky.
- Volitelné parametry - Jako volitelné parametry jsme zvolili dvourozměrné pole bytů 4x3 s libovolnými hodnotami. Volitelné parametry zabírají 8 znaků.
- Byty entropie - Byty entropie získáme při generování klíče z licenčního kódu, při validaci nám je zjistí aplikace sama. Byty entropie zastupuje 10 znaků.
- Datum - Datum se skládá ze dne, měsíce a roku a určuje nám, do kdy je licence platná. Den je zakódován na 2 znaky, měsíc na 1 znak a rok na 2 znaky. Od roku je odečtena určitá hodnota, jinak by se nám to nevešlo do rozmezí 2 znaků. V našem případě odečítám hodnotu 2000 od aktuálního roku. Př. potřebujeme vyjadřit platnost do 27.11.2014 =>1BB0E (1B=27, B=11, 0E=14), 1.3.2100 => 01364
- Checksum - Checksum je kontrolní součet, který se používá při validaci, a tvoří se ze všech předešlých částí. Obsahuje 4 znaky.

Celková délka licenčního klíče je 35 znaků. Počet všech možností licenčního klíče, pokud nepočítáme datum, je $n = 16^{26}$, a tedy pravděpodobnost, že 2 licenční klíče budou stejné je $P = 1/n \doteq 4,9 \times 10^{-32}$. Pokud bychom brali v potaz i datum, to znamená, že 2 uživatelům bychom vygenerovali stejný klíč i stejné datum, pravděpodobnost se ještě zmenší.

2.3.4 Licence

Pro práci s licencemi jsem použil předpřipravenou knihovnu dostupnou na adrese [24]. Licence je objekt, který je uložen v zabezpečeném souboru. Umístění souboru licence je závislé na OS. OS Windows ukládá licenci podle hodnoty systémového proměnného prostředí ALLUSERSPROFILE, ostatní OS ukládají licenci podle domovského adresáře uživatele.

2.3.4.1 Parametry licence

Parametry jsou uloženy v objektu licence jako třídní proměnné.

Výčet parametrů

- Platnost do - Tento parametr je uložen ve formátu UNIX Timestamp (long)
- Licenčtní klíč - Licenční klíč je uložen jako řetězec znaků.
- Typ licence - Typ licence je uložen jako *Feature* objekt s názvem licence.

[5]Unix Time je způsob zápisu času. Je definován jako počet sekund uběhlých od 1. ledna 1970 00:00:00

2.3.4.2 Zabezpečení licence

Když vytváříme novou licenci, vytváříme tak soubor s licencí. Tento soubor je zašifrován a elektronicky podepsán. O šifrování a podepisování se nám stará knihovna skoro sama. My pouze musíme implementovat interface *PasswordProvider*, kterým nastavíme klíč k šifrování, a musíme vytvořit privátní a veřejný klíč, který se používá k podepisování. O vytvoření privátní a veřejného klíče se stará třída *KeysGenerator*, která i tyto klíče uloží jako soubory do stejného adresáře jako je licence. I tyto klíče jsou zašifrovány pomocí *PasswordProvideru*.

2.3.4.3 Typy licence

Jednotlivé typy licence implementujeme kombinací delegace a návrhového vzoru Strategy [13] používající výčtový typ enum[6].

Každý typ licence musí být uveden ve výčtovém typu *LicenseTypeEnum* a implementovat rozhraní *ILicenseType*. Pokud budeme chtít v budoucnu přidat typ licence, stačí implemetovat toto rozhraní a uvést typ licence ve výčtovém typu a nic jiného upravovat nemusíme. Proto zde uvádím ukázky kódu.

Typy licencí máme podle návrhu 2.2.7.2 tři. Jsou to TRIAL, FREE a FULL.

- TRIAL - U tohotu typu licence je pevně nastavená doba trvání a to 30 dní. Jelikož tento typ bude moci být vytvořen pouze jednou, musíme si uložit informaci, že licence byla vytvořena. Tohoto dosáhneme uložením informace do registru[7] ve formátu zašifrovaného data platnosti v Base32. Pokud si uživatel licenci vymaže, může si ji znovu vytvořit s datem platnosti, který je uložen v registru.
- FREE a FULL - Co se týče realizace, tyto dva typy licence se budou vytvářet na stejném principu, bude potřeba zadat validní licenční klíč. Dobu platnosti zjistíme z klíče. Validace bude probíhat také stejně.

ILicenseType je rozhrání, které musí implementovat každý typ licence. Každý typ licence musí definovat metody *createLicense* a *validateLicense*.

- createLicense - Tato metoda vytváří novou licenci. Jako parametry ji předáváme licenční klíč a název typu licence. Na ukázce kódu metody můžeme vidět, jak se vytváří FULL typ licence.

```
public License createLicense(String licenseKey, String name)
    throws LicenseException {
        long goodBeforeDate = LicenseKeyParser.
            getGoodBeforeDate(licenseKey);
        if(goodBeforeDate < System.currentTimeMillis()){
```

[6] Umožňuje definovat vlastní datové typy, které mohou nabývat určité hodnoty. Jedna hodnota se rovná právě jednomu výčtu

[7] Je to databáze, do které si Windows ukládá všechna svá nastavení. Ekvivalent registrů v linuxu jsou textové soubory, které jsou uložené v /etc nebo v domovském adresáři jako skryté soubory.

```
            throw new ExpiredLicenseKeyException("License key
                has expired.");
        }
        License license = new License.Builder().
                withProductKey(licenseKey).
                withGoodBeforeDate(goodBeforeDate).
                addFeature(name).
                build();
        return license;
}
```

- validateLicense - Tato metoda ověřuje licenci. Jako parametry ji předáváme existující licenci a entropii typu licence. Na ukázce kódu metody můžeme vidět, jak se validuje FULL typ licence.

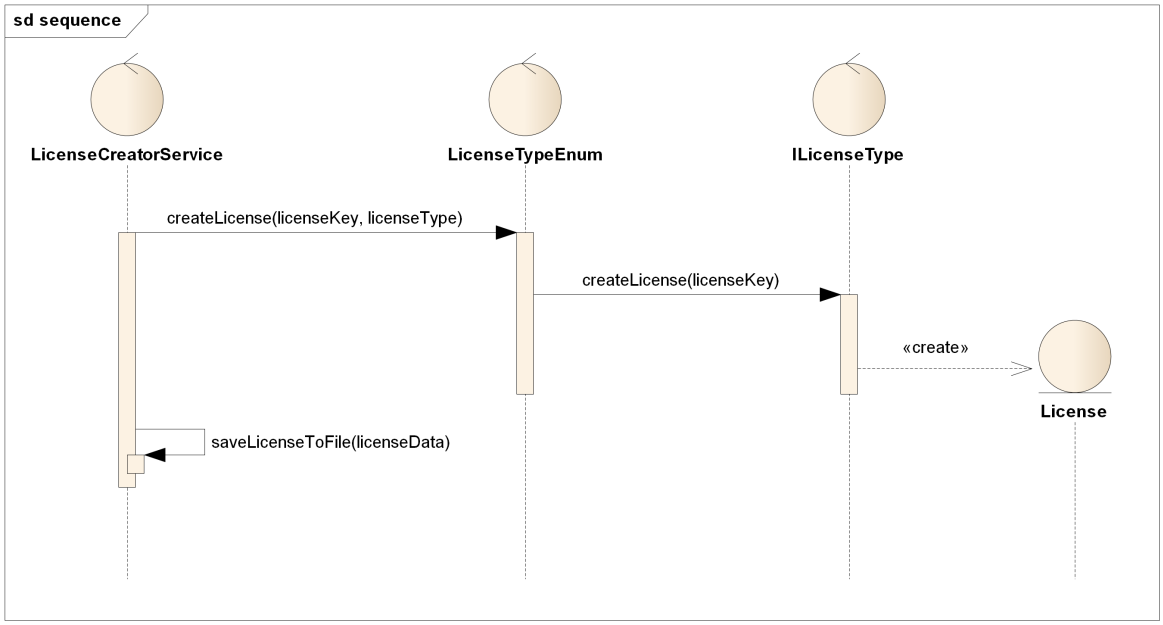
```
public void validateLicense(License license, byte[]
    licenseEntropy) throws InvalidLicenseException {
        ILicenseKeyValidator lkValidator = new
            LicenseKeyValidator(new Entropy());
        boolean result = lkValidator.validate(license.
            getProductKey(),licenseEntropy);
        if (!result) {
            throw new InvalidLicenseException("License is not
                valid.");
        }
        if (license.getGoodBeforeDate() < System.
            currentTimeMillis()) {
            throw new ExpiredLicenseException("License has
                expired.");
        }
    }
```

LicenseTypeEnum je výčový typ, kde jsou uvedeny všechny typy licencí. Obsahuje pomocné proměnné název typu licence, entropie licence, znak typu licence a *ILicenseType*. Poskytuje metody pro získání těchto proměnných a metody pro vytvoření a ověření licence, které delegují svoje chování na třídy typu *ILicenseType*. Na příkladu kódu můžete vidět všechny typy licencí a jak se nový enum vytváří.

```
TRIAL ("trial", new byte[]{}, "1", new TrialLicense()),
FREE ("free", new byte[]{1,1,1,1}, "2", new FreeLicense()),
FULL ("full", new byte[]{2,2,2,2}, "3", new FullLicense())
```

2.3.4.4 Vytvoření licence

Vytvoření licence má na starost třída *LicenseCreatorService*. Třída licenci vytvoří a uloží ji do souboru. Dále je ukázán postup spolupráce objektů.



Obrázek 2.6: Sekvenční diagram vytvoření licence

2.3.4.5 Ověření licence

Ověření licence má na starost třída *LicenseCreatorService*. Třída licenci načte ze souboru a ověří ji. Spolupráce objektů probíhá na stejném principu jako u vytvoření licence.

2.3.5 LicenseController

LicenseController je třída implementovaná podle návrhového vzoru Singleton [29], to znamená, že v celém programu běží pouze jedna její instance. Co se týče licencí, veškerá komunikace probíhá přes třídu *LicenseController*. Controller umožňuje vytvářet licence, ověřovat licence, kontrolovat typ licence a zjišťovat dobu platnosti licence.

2.3.6 Licenční server

Licenční server je psán v jazyce PHP. Jediná úloha licenčního serveru je generovat licenční klíč z daného licenčního kódu a posílat ho na email. Email a licenční kód se zadává do formuláře. Licenční server je dostupný na adrese <www.cashbob.php5.cz>.

2.4 Testování

Testování je nedílnou součástí celého vývoje. Odhaluje chyby ve vývoji a zabraňuje tak tvoření nových chyb. Proto se v této kapitole budu věnovat typům testů a technologiím jaké jsem použil k testování.

2.4.1 Jednotkové testy

Pro jednotkové testy jsem použil frameworku JUnit [15], který je určen k testování jednotek pro programovací jazyk Java. Otestoval jsem generování a validaci licenčního klíče a tvorbu a validaci *License* objektu.

2.4.2 Statická analýza kódu

Ke statické analýze kódu jsem použil testovací nástroj PMD [16]. Tímto jsem odstranil např. nepotřebné importy a proměnné. PMD hlasí jako chybu, pokud je v metodě vícekrát uveden return. Tuto chybu jsem ignoroval, protože mi nepřijde jako špatný programovací standard, mít více returnů v jedné metodě. Někdy to kód může zjednodušit a zpřehlednit.

2.4.3 Akceptační testy

Pomocí testování funkčnosti jsem zjišťoval, jestli aplikace dělá to, co má dělat. Postupoval jsem podle scénaře 2.4.3.

Akce	**Očekávaný výsledek**	**Status**
Zadání správného licenčního kódu na licenční server	Vygenerovaný licenční klíč je poslán na uvedený email	OK
Zadání nesprávného licenčního kódu na licenční server	Zobrazení hlášky: Wrong license code!	OK
Zadání nesprávného licenčního klíče	Zobrazení hlášky: Invalid license key!	OK
Zadání licenčního klíče, kterému vypršela platnost	Zobrazení hlášky: Expired license key!	OK
Licenci vypršela doba platnosti	Zobrazení hlášky: Your license has expired!	OK
Doba platnosti licence je menší než 7 dní	Zobrazení informace o zbývající době platnosti licence	OK
Vytvoření licence typu TRIAL	Vytvoření složky licenses se souberem license.lic	OK
Vytvoření licence typu FREE	Vytvoření složky licenses se souberem license.lic	OK
Vytvoření licence typu FULL	Vytvoření složky licenses se souberem license.lic	OK
Vytvoření licence typu TRIAL podruhé	Vytvoření licence s původní dobou platnosti	OK
Vytvoření licence typu TRIAL podruhé když vypršela platnost	Zobrazení hlášky: Your trial license has expired and another trial license cannot be created!	OK
Změna dat v souboru license.lic	Zobrazení hlášky: Invalid license!	OK

Tabulka 2.1: Scénář akceptačních testů

Kapitola 3

Nasazení

Nasazení softwaru [17] je proces, který připravuje aplikaci pro uvedení na trh. Nově vytvořená aplikace může fungovat správně na našem počítači, ale to neznamená, že je opravdu připravena k použití pro ostatní. Je zde mnoho pomocných programů, skriptů, dokumentů, které my jako vývojář nebudeme potřebovat, ale ostatním uživatelům tyto pomocné programy, skripty, dokumenty mohou výrazně pomoci s prací s naší aplikací.

3.1 Analýza

V této kapitole se budu zabývat jaké jsou možnosti pro usnadnění práce uživatele s aplikací a zanalyzuji možnosti, jak dosáhnout správného nasazení aplikace z pohledu vývojáře.

3.1.1 Pomocné programy, skripty, dokumenty

Je důležité zahrnout nějaké pomocné programy, skripty, dokumenty k aplikaci, aby jsme usnadnili uživatelům práci s aplikací a my se chránili proti pirátství. Uvedu zde výčet programů, skriptů, dokumentů, které bychom měli poskytnout při nasazení aplikace.

- Pomocné dokumenty - Jednou z nejdůležitějších oblastí pro aplikaci je jednoduchost jejího použití. Proto je důležité, dodat spolu s aplikací nějakou pomoc, návod k použití, tutoriál nebo nápovědu. Můžeme vytvořit jednoduchý textový dokument, html soubor, pdf soubor, atd. se screenshoty a instrukcemi, jak používat aplikaci. V každém případě, dobrý pomocný dokument usnadní používání aplikaci uživatelům.

- Spustitelné soubory - Spustitelný soubor zajišťuje snadné spuštění aplikace. Dnešní uživatelé jsou zvyklí jednoduše dvakrát kliknout na ikonku ke spuštění aplikace a ne zadávat příkazy do příkazové řádky. Zadávat příkaz do příkazové řádky ke spuštění aplikace je pro uživatele velmi nepraktické a je dost možné, že by si s tím mnoho uživatelů neporadilo a tím by ani nepoužívali naši aplikaci. Proto je důležité poskytnout uživateli nějaký program, skript, na který jednoduše klikne a ten se postará o odeslání příkazu a potřebných parametrů ke spuštění aplikace.

- Ikony, zástupci - Ikony a zástupci nejsou nijak důležité k běhu programu, ale aplikace vypadá více profesionálně, když máme svůj obrázek (ikonu) u souboru a je snadnější ji spustit prostřednictvím zástupce. Uživatelům se může stát, že nainstalují aplikaci do file systému a zapomenou k ní cestu. Pokud máme zástupce na ploše nebo ve skupině programů je velice jednoduché pro uživatele spustit aplikaci a to i bez znalosti cesty k ní.

- Licenční podmínky - Jeden způsob, jak chránit aplikaci proti pirátství je souhlas s licenčními podmínky. V licenčních podmínkách bychom měli uvést naše vlastnictví a co může uživatel dělat se zdrojovým kódem aplikace.

- Instalační program - Viz sekce 3.1.3

- Odinstalační program - Je důležité, aby uživatel mohl jednoduše odinstalovat program, proto je potřeba poskytnout program, který toto umožňuje. Program by měl odstranit všechny soubory, které byly při instalaci zkopírovány do počítače.

3.1.2 Sestavení aplikace

Aby vůbec mohla aplikace fungovat na cílovém počítači, je potřeba nejdříve nějakým způsobem sestavit její soubory, které ke své činnosti potřebuje. Sestavení (angl. build) [19] aplikace je proces, kdy soubory převedeme do určité podoby, nejčastěji do takové podoby, aby bylo možné aplikaci spustit. Proces sestavení aplikace se většinou skládá z několika části a ke sestavení se používají buildovací nástroje.

3.1.2.1 Typy souborů potřebné k běhu aplikace

Aplikace se skládá z různých typů souborů. Výčet nejdůležitějších souborů uvedu dále.

- Zdrojové soubory - Ve zdrojových souborech je uložen zdrojový kód aplikace.

- Class soubory - Class soubory jsou zkompilované zdrojové souboru obsahující byte kód, který může být vykonán.

- Artefakty - Artefakt je v podstatě JAR soubor. O artefaktu se mluví také jako o knihovně třetí strany nebo o sestaveném projektu.

- Konfigurační soubory - Konfigurační soubory slouží k nastavení variabilních vlastností aplikace. Soubory jsou napsané většinou ve značkovacím jazyku např. XML nebo jako prosté textové properties soubory.

- Skripty - Skript je textový soubor obsahující sérii příkazů. Skripty se používají jako spouštěcí soubory, skripty k nastavení databáze, atd.

- Pomocné soubory - Pomocné soubory jsou např. obrázky a různé šablony, které jsou potřeba k běhu aplikace, nebo pomocné dokumenty.

3.1.2.2 Výsledek sestavení aplikace a uspořádání souborů

Když programujeme aplikaci, většinou nepracujeme pouze se zdrojovými soubory, ale přebíráme artefakty z třetí strany, načítáme informace z konfiguračních souborů, používáme skripty a pokud máme GUI aplikace, načítáme obrázky. Při sestavení aplikace proto musíme dávat pozor na to, abychom tyto soubory umístili tam, kde je aplikace očekává a také si musíme dát pozor na nastavení classpath[1].

Jelikož aplikace CashBob se skládá z modulů, uvedu zde 3 možnosti sestavení a uspořádání souborů ve výsledné aplikaci pro multi-modulární aplikaci.

1. **Uber JAR**

   Uber JAR je jeden velký JAR soubor, ve kterém jsou zabaleny všechny balíčky(angl. packages) a závislosti(angl. dependencies) projektu. Výhoda je, že se nemusíme starat, jestli jsou nainstalovány všechny závislosti, protože je máme všechny v JARu a také nemusíme řešit classpath. Nevýhoda nastává hlavně při upgradu, jelikož musíme vyměnit celý JAR soubor.

2. **Všechny artefakty odděleně**

   V tomto uspořádání budou všechny artefakty projektu oddělené jak moduly, tak i knihovny třetích stran. Výhoda je v upgradování. Pokud změníme nějaký artefakt, stačí pouze nahradit tento artefakt. Nevýhodou je, že se musíme starat o classpath.

3. **Moduly jako balíčky a odděleně artefakty třetích stran**

   Tato možnost uspořádání souborů je hodně podobná uspořádání všechny artefakty odděleně. Jediný rozdíl je, že artefakty modulů budou rozbalené, tudíž budeme moci rovnou přistupovat k jednotlivým souborům modulů, což nám zjednoduší upgrade. Artefakty třetích stran necháme v JARu, protože se nepředpokládá, že bychom nějak zasahovali do implementace těchto artefaktů.

3.1.3 Instalační program

Pokud máme sestavenou aplikaci, potřebujeme nějaký nástroj, kterým by si uživatel jednoduše tuto aplikace na svém počítači zprovoznil. K tomuto účelu slouží instalační program. Instalační program seskupuje všechny potřebné soubory ke spuštění aplikace a pomocí průvodce instalací je uživatel schopen si tyto soubory jednoduše nainstalovat.

Jednotlivé kroky, které může instalační průvodce zahrnovat

- Souhlas s licenčními podmínky - Uživatel musí souhlasit s licenčními podmínky, jinak ho instalační průvodce nepustí dále. Jde hlavně o ochranu proti neautorizovanému použití.
- Výběr cíle - Uživatel si bere umístění ve file systému, kam chce aplikaci nainstalovat.

[1] Proměnné prostředí, které, říká Java Virtual Machine nebo Java kompileru, kde má hledat uživatelem definované třídy a balíčky.

- Výběr balíků - Uživatel si může vybrat pouze nějaké části aplikace, které chce nainstalovat.
- Postup instalace - Uživateli se zobrazí indikátor průběhu instalace.
- Vytvoření zástupců - Uživatel si může zvolit, jestli chce vytvořit zástupce na ploše nebo ve skupině programů.
- Informace o aplikaci - Uživatelovi se zobrazí tzv. README (informace o aplikaci).

Instalační program nemusí být zaměřen pouze na nainstalování potřebných souborů, ale může také spouštět skripty, další programy, kontrolovat, jestli vůbec bude možné aplikaci na počítači spustit a vytvářet odinstalační programy.

3.1.4 Původní stav

Jelikož se vývoj aplikace CashBob trvá zhruba 5 let, studenti Štěpán Tesař [32] a Jaroslav Rohlíček [23] v rámci svých bakalářských prací zkoušeli nasadit aplikaci. Postup sestavení aplikace a instalační programy, které používali, mně nepřišli příliš vhodné. Problém sestavení aplikace jak ho implementoval Štěpán Tesař je, že je platformově závislý, protože se spouští přes exe soubory, které lze spustit pouze na OS Windows. Tento způsob sestavení použil i Jaroslav Rohlíček. Od dob kdy na aplikaci pracoval Štěpán Tesař, se hodně věcí změnilo, takže jeho instalačním nástrojem se vůbec nebudu zabývat. Jaroslav Rohlíček navrhl používat jako instalační nástroj Install4j [3], který je placený. Další problémy jsou duplikace kódu v každém modulu v POMu[2] a zcela nevyužité maven pluginy.

[2] Je XML reprezentace Maven projektu.

3.2 Návrh

V této části navrhnu řešení nasazení aplikace CashBob z podkladů z předchozí části 3.1.

Jelikož aplikace CashBob je vyvíjena v jazyce Java a tím pádem je možnost ji nasadit na kterýkoliv OS, který je s Javou kompatibilní, bylo by zbytečné ji dělat jako platformově závislou. Existuje spousta platformově nezávislých instalačních programů pro Javovské aplikace, takže i při sestavování aplikace se zaměřím na to, aby aplikace byla platformově nezávislá. S panem Ing. Martin Komárkem jsme se domluvili, že klientská i serverová část aplikace se nasadí na stejný počítač.

3.2.1 Pomocné programy, skripty, dokumenty

Aby uživatel, který si pořídí aplikaci CashBob, ji mohl co nejdříve a co nejjednodušeji používat, je potřeba mu poskytnout možnost si ji jednoduše nainstalovat, spustit a používat. Proto zde navrhnu programy, skripty, dokumenty, které by aplikace CashBob měla obsahovat.

- Návod k použití - Návod k použití je dokument obsahující text a screenshoty. Obsahuje návod jak s aplikací správně pracovat a všechny možnosti jejího použití.
- Spustitelné soubory - Aby i nezkušený uživatel, který neví, že soubor typu JAR lze spustit příkazem z příkazové řádky, si mohl jednoduše spustit aplikaci, je potřeba mu dodat typy spustitelných souborů, na který je zvyklý. To znamená pro uživatele OS Windows dodat soubory typu exe a pro linuxové uživatele soubory typu sh.
- Ikony, zástupci - Nejjednoduší způsob, pro uživatele, na který je většina z nich zvyklá, je spouštět aplikace přes zástupce na ploše nebo v programové skupině, který má nějaký specifický obrázek (ikonu). Proto i aplikace CashBob by měla mít tuto možnost spouštění.
- Licenční podmínky - Souhlas s licenčníma podmínkama je ochrana nás vývojařů před neautorizovaným používáním.
- Instalační program - Viz sekce 3.2.3
- Odinstalační program - Viz sekce 3.2.3

3.2.2 Sestavení aplikace

Aplikace CashBob je rozdělena na klientskou a serverovou část, proto při sestavení aplikace budeme muset sestavit zvlášť klienta a zvlášť server. Jak klient, tak i server jsou závislé na jednotlivých modulech, je tedy potřeba nejdříve sestavit tyto moduly. Pro sestavení aplikace budeme používat buildovací nástroj Maven a jeho pluginy. Po domluvě s panem Ing. Martinem Komárkem jsme se dohodli, že pro sestavení klientské i serverové části použijeme Uber JAR.

3.2.2.1 Sestavení modulů

Jelikož se aplikace CashBob skládá z modulů a tyto moduly jsou na sobě závislé, musíme nejdříve sestavit každý modul, aby aplikace fungovala. Budeme potřebovat zkompilovat zdrojové soubory a tyto zkompilované soubory společně se zdroji(angl. resources) přidat do JARu. K tomuto účelu nám poslouží Maven [4] a jeho pluginy, které definujeme v POMu.

3.2.2.2 Klientská část

Pro sestavení klientské části použijeme Uber JAR. Klientská část je závislá na modulech *CashBob, CashBobStorage, CashBobPokladna, CashBobLibrary, CashBobRestLib, CashBobWorhShift* a na artefaktech třetích stran. Všechny tyto artefakty musíme přidat do Uber JARu. Klientská část také používá soubory ze složky config a images, které načítá z relativní cesty, proto i tyto složky musíme přidat k výslednému sestavení, aby klientská část fungovala správně. K sestavené aplikaci přidáme pomocné programy, skripty a dokumenty a výsledná klientská část je připravena k nasazení.

3.2.2.3 Serverová část

Sestavení serverové části funguje na stejném principu jako sestavení klientské části. Také pro sestavení budeme používat Uber JAR. Serverová část je závislá na modulech *CashBobServer, CashBobLibrary, CashBobRestLib*. Takže artefakty těchto modulů spolu s artefakty závislostí musíme přidat do Uber JARu. Serverová část ke svému správnému běhu potřebuje soubory ve složce config, takže i tuto složku musíme přidat do výsledného sestavení. Přidáme pomocné programy, skripty a dokumenty a serverová část je připravená k nasazení.

3.2.3 Instalační program

Aby uživatel, který si pořídil aplikaci CashBob ji mohl začít používat, nejprve si ji musí nainstalovat. Proto je potřeba instalační program, který uživatele provede instalací a on bude schopen si aplikaci co nejjednodušeji nainstalovat a také spustit. Navrhnu zde řešení, jak by takový program měl vypadat a jaké funkce by měl obsahovat.

Instalační program bude obsahovat sestavenou serverovou i klientskou část aplikace v balících, které si uživatel nainstaluje na svůj počítač.

Instalační program bude obsahovat průvodce instalací, který provede uživatele veškerými kroky instalace. Jednotlivé kroky uvedu dále.

- Souhlas s licenčními podmínky - Je to hlavně pro nás jako vývojáře ochrana proti pirátství. Uživateli se zobrazí text s licenčními podmínky a pokuď tyto podmínky neodsouhlasí, instalace se přeruší a aplikace nebude nainstalována.
- Výběr cíle - V tomto kroku se uživateli zobrazí okno, kde bude mít možnost si vybrat umístění, kam chce aplikaci nainstalovat.
- Výběr balíků - Uživatel si bude moci vybrat balíky jaké chce nainstalovat (klientská část a serverová část).

- Postup instalace - Uživatelovi si se zobrazí průběh instalace.
- Vytvoření zástupců - Uživatel si bude moci zvolit, jestli chce vytvořit zástupce na ploše nebo ve skupině programů.

Jelikož aplikace je psána v jazyce Java, je potřeba mít na počítači nainstalováno JRE. Proto musí instalační program zjistit, jestli uživatel na svém počítači má dostupné JRE. Pokud by neměl, musí instalační program tuto událost uživateli oznámit a uživatel si JRE musí doinstalovat.

Když si aplikaci uživatel nainstaluje, musí mít i možnost si ji jednoduše odinstalovat. Proto po nainstalování aplikace, se vytvoří i odinstalační program, prostřednictvím kterého bude moci uživatel jednoduše aplikaci odinstalovat.

3.2.4 Původní stav

V sekci 3.1.4 jsem zmínil problémy, se kterými jsem se setkal v projektu. První problém je, že aplikace je sestavována pouze jako exe soubory. Tento problém vyřešíme tak, že pro sestavenou aplikaci, aby byla platformově nezávislá, použijeme JAR soubory viz sekce 3.2.2. Druhý problém je, že jako instalační program je navržený Install4j, který je placený. Existuje spousta volně dostupných platformově nezávislých instalačních programů a proto jeden z nich použijeme. Problém s duplikací kódu v POMu vyřešíme přes dědičnost. Jelikož POM soubory dovulují dědičnost, stačí jednotlivé pluginy uvést pouze v rodičovském POMu. Nevyužité pluginy, jako třeba zabalení zdrojových souborů a vytvoření Uber JARu z každého modulu aplikace, které jsou pro sestavení aplikace úplně zbytečné, jednoduše vymažeme.

3.3 Realizace

V této části popíš řešení implementace nasazení aplikace CashBob z podkladů z předchozí části 3.3.

Téměř veškerá implementace týkající se nasazení je realizována prostřednictvím Maven pluginů. Vše probíhá úpravami POMu a deskriptorů jednotlivých pluginů. Pouze pomocné programy, skripty, dokumenty nejsou realizované pomocí pluginů.

3.3.1 Pomocné programy, skripty, dokumenty

Pomocné programy, skripty, dokumenty přímo nesouvisí s funkčností aplikace CashBob, ale usnadňují prácí s ní a nás ochraňují před neoprávněným zneužitím aplikace.

Znění licenčních podmínek jsem převzal od studentů, kteří na aplikaci CashBob pracovali přede mnou.

Návod k použití jsem nijak nerealizoval, protože už mi nezbýval čas. I když tento dokument nijak nesouvisí s funkčností aplikace, myslím si, že je hodně důležitý pro úspěch aplikace, protože se uživatel dozví o všech možných funkcí aplikace, na které by třeba pouhým zkoušením vůbec nepřišel. Proto by určitě bylo vhodné, aby v budoucnu nějaký návod vznikl.

O instalačním, odinstalačním programu a zástupcích se budu zabývat v sekci 3.3.3.

3.3.1.1 Spustitelné soubory

Tyto soubory zjednušují spouštění aplikace CashBob. Jedná se o skripty, které přes příkazový řádek zpustí JAR soubor. Je to nejjednodušší způsob, jak spustit Java aplikaci. Nevýhodou je, že na daném OS musí být nainstalované JRE. Touto problematikou se budu zabývat v sekci 3.3.3.3. Jelikož je aplikace rozdělena na klientskou a serverovou část musí existovat spouštěcí skript pro každou část. Tyto skripty jsou platformově závislé. Skripty jsem implementoval pro OS Windows a OS Linux.

1. **OS Windows**

   Pro OS Windows jsem neimplementoval přímo skripty, ale dávkové soubory. Pro klientskou a serverovou část jsem napsal jednoduchý dávkový soubor, který zjistí, jestli je JRE dostupné a spoustí JAR soubory jednotlivých částí. Třetí skript, který se stará o spuštění jak serverové, tak i klientské části pouze zavolá předešlé dva skripty. Jelikož uživatelé Windows jsou zvyklí spouštět aplikace přes exe soubory, pomocí konvertoru jsem převedl dávkový soubor (přípona .bat) na exe soubor. Výhoda exe souboru je také, že k němu lze připojit ikonu.

2. **OS Linux**

   Na stejném principu jako u OS Windows jsem vytvořil skripty pro spouštění aplikaci na Linuxu. Nevýhoda skriptů v Linuxu je, že k nim nelze připojit ikonu.

Pokud bychom chtěli nasadit aplikaci i na jinou platformu než je Windows nebo Linux a chtěli uživatelům poskytnout spouštěcí soubory, museli bychom doimplementovat skripty závislé na platformě.

3.3.2 Sestavení aplikace

K sestavení aplikace budu používat Maven pluginy. Nejdříve budeme muset sestavit jednotlivé moduly. Poté sestavíme serverovou a klientskou část aplikace.

3.3.2.1 Sestavení modulů

Sestavení modulu můžeme rozdělit do několika fází ve kterých bude používat odlišné pluginy.

1. process-resources (plugin resources [11]) - Zdroje projektu jsou zkopírovány do výstupní složky.

2. compile (plugin compiler [7]) - Zdrojový kód projektu je zkompilován a jsou vytvořené class soubory. Jako JDK používám verzi 1.7.

3. test (plugin surfire [14]) - V této fázi probíhají unit testy. Tato část není pro sestavení aplikace důležitá.

4. package (plugin jar [10]) - Z class souborů a zdrojů je vytvořen JAR soubor.

5. install (plugin install [8]) - V této části je artefakt (JAR) přidán do lokálního repozitáře pro použití jako závislost v ostatních modulech.

3.3.2.2 Klientská část

Sestavení klientské části se provádí přes modul *CashBob*. Pro sestavení jsem použil plugin Shade [12], který dokáže vytvořit Uber JAR. Plugin Shade načte všechny balíčky projektu a jeho závislosti a sestaví jeden velký JAR soubor. Abych mohl JAR spustit, musel jsem uvést hlavní třídu. Hlavní třída se nastaví v konfiguraci pluginu Shade. Aby aplikace fungovala je potřeba k Uber JARu přidat soubory ve složkách config, templates a images a spustitelné soubory. Pomocí pluginy Assembly [6] a jeho dekskriptoru jsem tyto složky a soubory zkopíroval do stejné složky jako je Uber JAR a klienstká část aplikace byla připravená k nasazení.

```
CashBobClient
 |--- CashBobClient.exe (CashBobClient.sh)
 |--- CashBob-1.0-shaded.jar
 |--+ config
 |--+ templates
 |--+ images
 |--- LICENSE
```

3.3.2.3 Serverová část

Sestavení serverové části se provádí přes modul *CashBobServer*. Princip sestavení je úplně stejný jako pro klientskou část. Jediný rozdíl je, že serverová část nepotřebuje přidat složku images.

```
CashBobServer
 |--- CashBobServer.exe (CashBobServer.sh)
 |--- CashBobServer-1.0-shaded.jar
 |--+ config
 |--- LICENSE
```

3.3.3 Instalační program

Jako instalační program jsem použil IzPack [9], který měl velmi dobré ohlasy na internetu. Výhoda IzPacku je, že lze použít jako Maven plugin, takže vytváření instalačního programu je automatizované, a že je zcela zdarma. Nevýhoda je, že je to Javovská aplikace, takže ke svému běhu potřebuje JRE.

Instalační program vytvářím pomocí IzPacku v modulu *CashBobDistribution*. Chování IzPacku se definuje v jeho deskriptoru (install.xml). IzPack pracuje s balíky, které mu musíme předat. Jako jeden balík jsem použil sestavenou klientskou část aplikace CashBob a jako druhý balík servrovou část aplikace. Třetí balík obsahuje pomocné skripty ke spuštění instalačního programu, skript ke spuštění klientské a serverové části aplikace naráz a ikony zástupců.

IzPack má předdefinované GUI panely, které uživatele provedou různými kroky instalace. Jednotlivé panely, která jsem použil uvedu dále v pořadí, v jakém se objevují při instalaci.

1. HelloPanel - Uvítací panel, který zobrazuje uživateli, že si spustil instalační program aplikace CashBob.
2. LicencePanel - V tomto panelu se uživateli zobrazí licenční podmínky, se kterými musí souhlasit, jinak ho instalační průvodce nepustí dále.
3. TargetPanel - Zde si uživatel může vybrat, kam chce aplikaci umístit.
4. PacksPanel - V tomto panelu si uživatel může vybrat, jaké balíky chce nainstalovat. Jsou zde tři balíky CashBobClient, CashBobServer a CashBob. V současné době jsou všechny balíky povinné.
5. SummaryPanel - Zobrazí uživateli informaci o umístění aplikace a o balících, které si vybral.
6. InstallPanel - Panel zobrazuje průběh instalace.
7. ShortcutPanel - V tomto panelu si uživatel může nechat vytvořit zástupce na ploše nebo v programové skupině.
8. InfoPanel - Tento panel zobrazí základní informace o aplikaci.
9. SimpleFinishPanel - Ukončovací panel, který uživateli zobrazí informaci, že aplikace byla nainstalována.

3.3.3.1 Odinstalační program

IzPack automaticky po dokončení instalace vytvoří odinstalační program. Je to javovská aplikace, která dokáže odinstalovat soubory a složky, které instalační program vytvořil.

3.3.3.2 Vytvoření zástupce

Pokud uživatel zvolí, že chce vytvořit zástupce, IzPack ho dokáže vytvořit jak na ploše, tak i v programové skupině. Vytvoření zástupců se řidí nastavením v xml souboru. Jelikož zástupci se vytvářejí jinak na Windows a jinak na Linuxu, bylo potřeba napsat pro každého zástupce jeden xml soubor.

3.3.3.3 Spouštěcí skripty instalačního programu

Jelikož instalační program je javovská aplikace, potřebuje mít k běhu dostupné JRE. Proto jsem musel před spuštěním aplikace nějak zjistit, jestli JRE je na počítači dostupné. K tomuto účelu jsem implementoval skript, který to dokáže zjistit. Tyto skripty jsou platformě závislé, proto bylo potřeba napsat jeden skript (dávkový soubor) pro Windows a jeden skript pro Linux, ale princip funkčnosti je stejný. Postup kroků jaké vykonává skript uvedu níže.

1. Je dostupné JRE?
   - Windows - Zjistíme, jestli je JRE dostupné v PATH, pokud není, prohledáme registry, jestli je zde uvedená cesta k JRE.
   - Linux - Pomocí příkazu *which java* zjistíme, jestli je JRE dostupné.
2. JRE není dostupné - Uživateli se zobrazí informace, že si musí doinstalovat JRE.
3. JRE je dostupné - Spustí se aplikace Instalační program.

3.3.3.4 Adresářová struktura nainstalované aplikace

```
CashBob
|
|--+CashBobClient
|   |--- CashBobClient.exe (CashBobClient.sh)
|   |--- CashBob-1.0-shaded.jar
|   |--+ config
|   |--+ templates
|   |--+ images
|   |--- LICENSE
|
|--+ CashBobServer
|   |--- CashBobServer.exe (CashBobServer.sh)
|   |--- CashBobServer-1.0-shaded.jar
|   |--+ config
|   |--- LICENSE
|
|--- CashBob.exe (CashBob.sh)
|
|--+ icon
|   |--- win_icon.ico (icon.png)
|   |--- win_uninstall.ico (uninstall.png)
|
|--+ Uninstaller
    |--- uninstaller.jar
    |--- uninstall.exe (uninstall.sh)
```

3.4 Testování

Nasadit aplikaci CashBob jsem zkoušel na OS Windows 7, OS Windows XP a Ubuntu 12.04. Manuálně jsem testoval, jestli proběhne správně instalace a jestli aplikace půjde spustit. Když se aplikace spustí, jestli se půjde přihlásit do aplikace a nakonec jestli aplikace půjde odinstalovat.

3.4.1 Manuální testování instalačního programu

Nainstalovat aplikaci jsem se pokusil pomocí spouštěcího skriptu instalačního programu s jak dostupným JRE, tak i když JRE nainstalováné na počítači nebylo. V případě nenainstalovaného JRE skript vypsal hlášku, že si musím doinstalovat JRE. Pokud jsem měl na počítači nainstalované JRE, instalační program se spustil a zobrazil se instalační průvodce. Pomocí průvodce jsem aplikace nainstaloval na určené místo a vytvořil zástupce ve skupině programů a na ploše. Instalační průvodce zobrazil informace, jak se do aplikace přihlásit a poté skončil. Během instalace se vytvořil i odinstalační program.

3.4.2 Manuální testování spuštění a přihlášení se do aplikace

Nainstalovanou aplikaci jsem se pokusil spustit jak pomocí zástupců, tak i pomocí spouštěcích skriptů. Jak klientská, tak i serverová část se spustila. Pro přihlášení do aplikace jsem použil údaje zobrazené během instalace.

3.4.3 Manuální testování odinstalování aplikace

Během instalace se vytvořil odinstalační program, který jsem použil k odinstalování aplikace. Spustil jsem odinstalační program pomocí skriptu a byly odinstalovány ty části, které během instalace byly vytvořeny, tzn. soubory a zástupci.

3.4.4 Závěr manuálního testování

Aplikaci jsem úspěšně nainstaloval, spustil, přihlásil a odinstaloval. Jediný problém nastal při instalaci. Pokud si chceme vytvořit zástupce pouze na ploše, instalační program IzPack nevytvoří žádného zástupce (bug IzPacku).

Kapitola 4

Závěr

Hlavním cílem mé bakalářské práce bylo nasadit aplikaci CashBob a zamezit jejímu neoprávněnému používání. Většinu z cílů zadaní jsem realizoval, některé se mi však splnit nepodařilo. Všechny cíle, které se mi podařilo realizovat, jsem zanalyzoval, navrhl, implementoval a otestoval.

V rámci nasazení se mi podařilo sestavit klientskou i serverovou část aplikace CashBob tak, aby aplikace byla platformově nezávislá a sestavení bylo automatizované. Aby aplikace šla jednoduše nainstalovat, vytvořil jsem platformově nezávislý instalační program. Věřím že instalační program je přívětivý jak pro uživatele, tak i pro vývojaře. Instalační program provede uživatele jednoduchým průvodcem instalace, kde si uživatel může zvolit kam chce aplikaci nainstalovat a jestli chce vytvořit zástupce, kterým aplikaci jednoduše spustí. Hlavní výhoda pro vývojáře je, že instalační program se vytvoří automaticky při sestavení aplikace. Co se týče nasazení, nepodařilo se mi realizovat upgradovací nástroj aplikace, kterým by si uživatel mohl jednoduše aktualizovat aplikaci na nejnovější verzi. Jedinou možnost, jakou má uživatel, pokud si chce pořídit nejnovější verzi aplikace, je si ji znovu nainstalovat.

Dále jsem vytvořil systém licencování, který zamezí neoprávněnému používání aplikace CashBob. Vytvořil jsem několik typů licencí, které uživatel může získat volně nebo po zadání povinných údajů v aplikaci a na licenčním serveru. V budoucnu, pokud bude aplikace komerčně využívána, je potřeba realizovat systém plateb, který spojí platbu se získáním licence.

Myslím si, že se mi podařilo splnit hlavní cíle bakalářské práce. Pro mně osobně byla práce velmi přínosná, i když velmi časově náročná (odhadem 450-500 hodin). Seznámil jsem se s novými technologiemi a měl jsem možnost si je vyzkoušet v praxi. Cennou zkušeností pro mě byla práce v týmu na rozsáhlejším projektu, který už se nějakou dobu vyvíjí.

Literatura

[1] *Software License Types* [online]. [cit. 15. 4. 2014]. Dostupné z: <https://tulane.edu/tsweb/software/software-license-types.cfm>.

[2] *CashBob* [online]. 2014. [cit. 15. 4. 2014]. Dostupné z: <https://redxml.felk.cvut.cz/>.

[3] *The Powerful Multi-Platform Java Installer Builder* [online]. [cit. 13. 5. 2014]. Dostupné z: <https://www.ej-technologies.com/products/install4j/overview.html>.

[4] *Apache Maven Project* [online]. [cit. 12. 5. 2014]. Dostupné z: <http://maven.apache.org/>.

[5] *Flyway Database Migrations Made Easy* [online]. [cit. 22. 5. 2014]. Dostupné z: <http://flywaydb.org/>.

[6] *Apache Assembly Plugin* [online]. [cit. 14. 5. 2014]. Dostupné z: <http://maven.apache.org/plugins/maven-assembly-plugin/>.

[7] *Maven Compiler Plugin* [online]. [cit. 14. 5. 2014]. Dostupné z: <http://maven.apache.org/plugins/maven-compiler-plugin/>.

[8] *Apache Maven Install Plugin* [online]. [cit. 14. 5. 2014]. Dostupné z: <http://maven.apache.org/plugins/maven-install-plugin/>.

[9] *IzPack* [online]. [cit. 14. 5. 2014]. Dostupné z: <http://docs.codehaus.org/display/IZPACK/Home>.

[10] *Maven JAR Plugin* [online]. [cit. 14. 5. 2014]. Dostupné z: <http://maven.apache.org/plugins/maven-jar-plugin/>.

[11] *Maven Resources Plugin* [online]. [cit. 14. 5. 2014]. Dostupné z: <http://maven.apache.org/plugins/maven-resources-plugin/>.

[12] *Apache Maven Shade Plugin* [online]. [cit. 14. 5. 2014]. Dostupné z: <http://maven.apache.org/plugins/maven-shade-plugin/>.

[13] *Combining Delegation- and Strategy-Pattern using an Enum* [online]. 2011. [cit. 21. 4. 2014]. Dostupné z: <http://jensonjava.wordpress.com/2011/10/31/combining-delegation-and-strategy-pattern-using-an-enum/>.

[14] *Maven Surfire Plugin* [online]. [cit. 14. 5. 2014]. Dostupné z: <http://maven.apache.org/surefire/maven-surefire-plugin/>.

[15] *JUnit, A programmer-oriented testing framework for Java.* [online]. 2014. [cit. 26. 4. 2014]. Dostupné z: <http://junit.org/>.

[16] *PMD* [online]. [cit. 26. 4. 2014]. Dostupné z: <http://pmd.sourceforge.net/>.

[17] ANNETTE, G. *What is Software Deployment?* [online]. 2011. [cit. 10. 5. 2014]. Dostupné z: <http://www.godtlandsoftware.com/word-press/2011/03/26/what-is-software-deployment/>.

[18] BRANDON, S. *Implementing a Partial Serial Number Verification System in Delphi* [online]. 2007. [cit. 20. 4. 2014]. Dostupné z: <http://www.brandonstaggs.com/2007/07/26/implementing-a-partial-serial-number-verification-system-in-delphi/>.

[19] CORY, J. *Build* [online]. [cit. 10. 5. 2014]. Dostupné z: <http://www.techopedia.com/definition/3759/build>.

[20] DAVID, S. *Licensing Module in Java* [online]. 2012. [cit. 21. 4. 2014]. Dostupné z: <http://afewguyscoding.com/2012/02/licensing-module-java/>.

[21] DOMINIC, H. *The 5 Trial-Licensing Scenarios Java Developers Can Encounter - and How to Support Each One* [online]. 2011. [cit. 14. 4. 2014]. Dostupné z: <http://java.dzone.com/articles/5-trial-licensing-scenarios>.

[22] DOUGLAS, C. *Base32 Encoding* [online]. 2002. [cit. 20. 4. 2014]. Dostupné z: <http://www.crockford.com/wrmg/base32.html>.

[23] JAROSLAV, R. *Úprava a příprava na reálné nasazení restauračního systému CashBob.* Bakalářská práce, ČVUT, 2013.

[24] NICK, W. *License Manager* [online]. 2014. [cit. 15. 4. 2014]. Dostupné z: <http://java.nicholaswilliams.net/LicenseManager/>.

[25] Přispěvatelé Wikipedie. *Šifrování dat* [online]. 2014. [cit. 16. 4. 2014]. Dostupné z: <http://en.wikipedia.org/wiki/Encryption>.

[26] Přispěvatelé Wikipedie. *Software license server* [online]. 2013. [cit. 17. 4. 2014]. Dostupné z: <http://en.wikipedia.org/wiki/Software_license_server>.

[27] Přispěvatelé Wikipedie. *MAC address* [online]. 2014. [cit. 17. 4. 2014]. Dostupné z: <http://en.wikipedia.org/wiki/MAC_address>.

[28] Přispěvatelé Wikipedie. *Elektronický podpis* [online]. 2014. [cit. 16. 4. 2014]. Dostupné z: <http://cs.wikipedia.org/wiki/Elektronický_podpis>.

[29] Přispěvatelé Wikipedie. *Singleton pattern* [online]. 2014. [cit. 21. 4. 2014]. Dostupné z: <http://en.wikipedia.org/wiki/Singleton_pattern>.

[30] Přispěvatelé Wikipedie. *Software license* [online]. 2014. [cit. 16. 4. 2014]. Dostupné z: <http://en.wikipedia.org/wiki/Software_license>.

[31] Přispěvatelé Wikipedie. *Unique identifier* [online]. 2014. [cit. 16. 4. 2014]. Dostupné z: <http://en.wikipedia.org/wiki/Unique_identifier>.

[32] ŠTEPáN, T. *Úprava a příprava na reálné nasazení systému CashBob*. Bakalářská práce, ČVUT, 2012.

Příloha A

Seznam použitých zkratek

GUI	Graphic User Interface
JAR	Java Archive
JDK	Java Development Kit
JRE	Java Runtime Environment
MAC	Media Access Control
OS	Operating System
PHP	Hypertext Preprocessor
POM	Project Object Model
REST	Representational State Transfer
VM	Virtual Machine

Příloha B

Obsah přiloženého CD

```
CD
|
|--- README.txt.....................stručný popis obsahu CD
|
|--+ install.......................adresář s instalačními programy
|
|--+ src
|  |
|  |--+ imp........................zdrojové kódy implementace
|  |
|  |--+ thesis.....................zdrojová forma práce ve formátu LATEX
|
|--+ text
   |
   |--- thesis.pdf.................text práce ve formátu PDF
```

Příloha C

Upgrade aplikace CashBob

V závěru jsem se zmiňoval, že se mi nepodařilo zrealizovat upgradovací nástroj aplikace CashBob. Jedinou možností pro uživatele, jak si pořídit novou verzi aplikace, je si ji znovu nainstalovat.

Při upgradování na novou verzi aplikace je nejdůležitější, aby zůstala databáze zachována. Pro práci s databází je použit framework Flyway [5], který zjistí, jestli je databáze vytvořená, a pokud není databáze vytvořená, vytvoří novou databázi. Umístění databáze záleží na OS. Umístění databáze v OS Windows záleží na systémovém proměnném prostředí (ALLUSERSPROFILE) a databáze je uložena ve složce CashBob. V ostatních OS je databáze uložena v domovském adresáři uživatele ve složce .CashBob.

Kam bude databáze uložena záleží na parametru *platform*, který se předává VM (př. -Dplatform="win"). Implementoval jsem 4 typy uložení databáze.

1. platform="win"- ALLUSERSPROFILE/CashBob - OS Windows.
2. platform="linux"- /var/lib/CashBob - Zde nastal problém, že zapisovat do této cesty může pouze root, takže jsem nakonec tuto možnost nevyužil.
3. platform="common"- user.home/.CashBob - Pro všechny OS, jelikož proměnná user.home by měla být platformě nezávislá.
4. platform=""- ./ - Pro testovací účely.

Pokud nezadáme parametr *platform* aplikace se pokusí zjistit na jakém OS běží a podle toho nastaví parametr. Pro OS Windows nastaví parametr na "win", jinak nastaví parametr na "common".