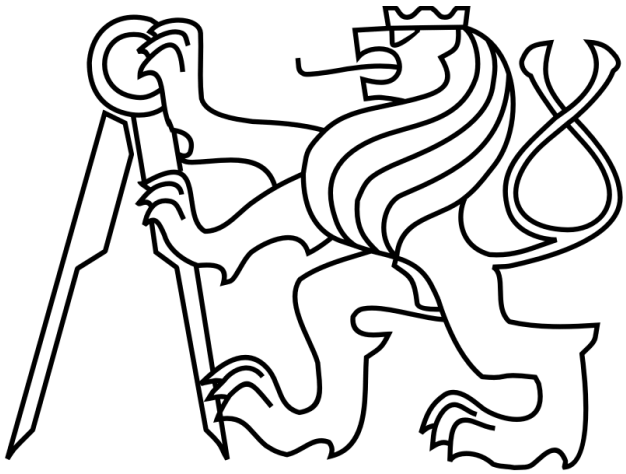
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA POČÍTAČŮ



BAKALÁŘSKÁ PRÁCE

NASAZENÍ A PROPAGACE RESTAURAČNÍHO SYSTÉMU

Vít Samek

Vedoucí práce: Ing. Martin Komárek

Studijní program: Softwarové technologie a management, Bakalářský
Obor: Softwarové inženýrství

PODĚKOVÁNÍ
Děkuji Ing. Martinu Komárkovi za odborné vedení, všestrannou pomoc a cenné rady při vedení mé bakalářské práce. Také děkuji své rodině a mým přátelům za poskytnuté zázemí a podporu během celého studia.

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 22. 05 .2015

Abstrakt

Aplikace Cashbob je pokladní systém pro restaurace a kavárny. Pro přípravu uvedení produktu na trh bylo nutné vyhodnotit výsledky testování v usability laboratoři a opravit nalezené chyby. Jedním z hlavních úkolů bylo vytvořit samostatný soubor pro spuštění celé aplikace, protože do této doby bylo nutné spustit nejdříve serverovou část aplikace a následně klientskou. Pro vetší pohodlnost při registraci aplikace bylo nutné umožnit registraci online, která povede k nížšímu počtu potřebných kroků k provedení registrace. Pro uživatele bylo naplánováno vyrobit videonávody a manuál k aplikaci, komunikační kanály jako Twitter, Facebook a připravit webové stránky produktu.

Opravou nalezených chyb došlo k odstranění závažných problémů, které bránily v používání aplikace v reálném provozu. Došlo také k rozšíření funkcionality, například implementací paměti pro rozpracované objednávky. Vzniklo zjednodušení implementace dotykové klávesnice a jejího použití napříč aplikací. Změnilo se používání konfiguračních souborů a byla vytvořena grafická komponenta, která kontroluje vlastní hodnotu pomocí přiřazeného validátoru. Podařilo se vytvořit jeden spouštěcí soubor pro celou aplikaci. Spuštění aplikace se rozšířilo o zamykání spuštěné aplikace a tedy lze mít pouze jednu spuštěnou instanci programu.Byly vytvořeny webové stránky dostupné na adrese www.cashbob.cz a účty na sociálních sítích. Vzniklo 7 základních videonávodů umístěných na YouTube.com a manuál doplňující videonávody.V závěru proběhla kontrola funkčnosti tisku a implementoval se tisk pro nové objednávky. Pro lepší použitelnost došlo k vytvoření distribuční databáze, která obsahuje pouze základní data.

Abstract

Cashbob is an application for restaurants and Cafés, which is being developed for more than 5 years by students of CTU in Prague. The aim of this bacherol thesis was to improve the application and prepare it for futher real-life use. It was necessary to remove all the errors found in the past usability test of the preexisting version. One of the most important tasks was to redesign the application as just one executable application file, because two separated files were needed till now. Next we wanted to simplify the registration proccess. We wanted the application to offer the online registration, meaning decreasement of steps to be done for the registration. Also video tutorials and manual handbook should have been produced. Web presentation and social media accounts (e.g. Facebook and Twitter) were meant to be the integral part of the project.

Removing the errors made possible the application to be published. The functionality of the program was extended by "Memory of unfinished orders". Also the implementation of screen keyboard was simplified. The method of reading configuration files was changed. There was created a new self-controllable GUI component having its own assigned validator, that can recognize, whather filled-in value is correct or not. Creation of only one executable file was successful and the start of the program was extended by locking of running application instance. The bill printing functionality was checked in the end of the work and was extended by printing the new order. Also a database containing simple sample data for users, was prepared. Web presentation is available on www.cashbob.cz. The application has its own social media accounts and 7 basic video tutorials can be seen on YouTube channel of the application.

Seznam obrázků

Seznam ukázek kódu

Obsah

1 Úvod

V poslední době lze vidět nastupující trend využívání elektronických pokladen a mobilních zařízení v restauracích a barech, kdy obsluha vyřizuje objednávky přes mobilní telefony nebo malé tablety. Systém Cashbob [1] je řešení pro restaurace a podniky, které uvažují právě o přechodu na elektronickou pokladnu. Systém v základní verzi nabízí pokladní modul pro vyřizování objednávek, přizpůsobený pro dotykové ovládání.

Aplikace CashBob [1] se skládá z několika modulů (Pokladní modul, Správa podniku, Plánovaní směn), kde každý modul rozšiřuje funkčnost základní verze. V aplikaci je také do budoucna připravena možnost rozšíření právě o mobilní zařízení, přes která půjde provést objednávku.

Práce na projektu Cashbob trvají více než 5 let a během této doby se vystřídalo na projektu velké množství lidí a aplikace prošla velkými změnami. Rozšiřováním funkčnosti přibývaly v aplikaci chyby, které branily reálnému nasazení.

V minulém roce se však zastavila večkerá implementace nové funkcionality a začaly se odstraňovat chyby, které bránily správnému fungování. Také se odstranily funkce, které se prozatím nepoužívaly, nebo nefungovaly zcela správně a provedlo se testování aplikace.

Mým úkolem je opravit chyby nalezené během testování. Dále se zaměřím na rozšíření možnosti registrace uživatele, spuštění aplikace jedním souborem, výrobu videonávodů a manuálu aplikace, vytvoření webových stránek a vytvoření komunikačních kanálů pro uživatele.

2 Rozšíření a úprava stávajícího kódu

Jedním z cílů této práce je upravit a vylepšit některé implementační části aplikace tak, aby bylo použití těchto částí do budoucna jednodušší a nevznikalo mnoho duplicitního kódu. Výsledkem takovýchto úprav je přehledný kód, který se dobře čte a dělá přesně to, co programátor očekává. [2] Výhodou přehledného kódu je například rychlé zaučení nového programátora při vstupu do projektu, rychlejší implementace nové funkcionality a díky tomu ušetřené peníze. [3]

V první části se zaměřím na dotykovou klávesnici, která se zobrazuje uživateli pokaždé, když je nutné zadat vstupní hodnoty. Dále se zaměřím na konfigurační soubory aplikace a jako poslední věc vytvořím validaci prvků ve formulářích, protože nyní neexistuje jednotná kontrola vstupních hodnot.

2.1 Úprava dotykové klávesnice

Jednou z komponent pro snadnější ovládání je dotyková klávesnice, která se uživateli zobrazí vždy, když potřebuje zadat hodnotu do vstupního pole (přihlašovací údaje, založení nového účtu, vytvoření nové položky,...).

2.1.1 Analýza stávajícího řešení

Klávesnice je reprezentovaná třídou *KyboardDialog.java,* která pracuje s přiřazeným vstupním polem třídy *JtextField*. Konkrétní přiřazení spočívá v definování události - kliknutím myši na dané vstupní pole. Programátor musí využít buď interface *MouseListener* a implementovat všech pět metod událostí (vytvoření anonymní implementace tohoto interfacu) nebo může využít třídy *MouseAdapter*, kde pouze zvolí konkrétní metodu, tedy metodu pro kliknutí. Obě tyto možnosti přináší generování mnoha zbytečného kódu. Ukázku stávajícího řešení vidíme v úkazce *kódu č.1*. Pokud se programátor rozhodne pro použití první možnosti, registrace jednoho vstupního elementu zabere 18 řádků a u druhé možnosti pouze řádků 8.

```
1.  public final class EditItemDialog extends AbstractFullScreenDialog {
2.  // ...
3.      public EditItemDialog(CellGridPanel form) {
4.          // ...
5.          jTextField1.addMouseListener(new MouseListener() {
6.              @Override
7.              public void mouseClicked(MouseEvent e) {
8.                  PokladnaViewController.requestKeyboardInput(jTextField1);
9.              }
10.
11.             @Override
12.             public void mousePressed(MouseEvent e) {}
13.
14.             @Override
15.             public void mouseReleased(MouseEvent e) {}
16.
17.             @Override
18.             public void mouseEntered(MouseEvent e) {}
19.
20.             @Override
21.             public void mouseExited(MouseEvent e) {}
22.         });
23.
24.         jTextField2.addMouseListener(new MouseListener() {
25.             @Override
26.             public void mouseClicked(MouseEvent e) {
27.                 PokladnaViewController.requestKeyboardNumericInput(jTextField2);
28.             }
29.
30.             @Override
31.             public void mousePressed(MouseEvent e) { }
32.
33.             @Override
34.             public void mouseReleased(MouseEvent e) {}
35.
36.             @Override
37.             public void mouseEntered(MouseEvent e) {}
38.
39.             @Override
40.             public void mouseExited(MouseEvent e) { }
41.         });
42.     }
43.
```

Kód 1: Ukázka registrace klávesnice pro vstupní pole

2.1.2 Návrh řešení

Třída reprezentující klávesnici by měla poskytovat rozhraní, které umožní jednoduché zaregistrování vstupního pole bez nutnosti řešení další logiky okolo, jako je výše uvedené vytváření událostí pro myš [4].

Důvodem, proč by měla sama třída řešit akci zobrazení klávesnice je, že se tato akce opakuje stále dokola pro všechny přiřazené komponenty. Proto můžeme tuto akci implementovat v této třídě jednou a neřešit přiřazení pokaždé, kdy je do formuláře přidáno nové vstupní pole.

2.1.3 Řešení

Ve třídě *KeyboardDialog.java* jsem vytvořil metody pro registraci klasické klávesnice (klávesnice s písmeny a s čísly), numerické klávesnice a klávesnice určené pro psaní hesla (při psaní textu se písmena nahrazují zástupnými znaky, např: *).Ukázku metody pro registraci klávesnice můžeme vidět v *kódu č. 2.* Vstupní parametr funkce je pole objektů typu *JTextField.java*, který reprezentuje vstupní pole ve formulářích. Uvnitř

metody se pak nachází jednoduchý iterační cyklus, který každé komponentě přiřadí reakci na událost kliknutím myši. Obdobně vypadají i metody pro další dva typy klávesnic.

```
        // ...
1.      public void registerStandardKeyboard(JTextField ... components){
2.          for(final JTextField component : components){
3.              component.addMouseListener(new MouseAdapter() {
4.                  @Override
5.                  public void mouseClicked(MouseEvent e) {
6.                      setTextField(component);
7.                      setVisible(true);
8.                  }
9.              });
10.         }
11.     }
        // ...
```

Kód 2: Metoda pro registraci klávesnice

Výsledkem přesunutí registrace události na kliknutí do třídy reprezentující klávesnici je zjednodušené používání, ušetření počtu řádků kódu a lepší čitelnost kódu. *Kód č. 3* ukazuje nový způsob, jakým lze registrovat klávesnici. V ukázce č. 4 vidíme, že i pro více vstupů zaregistrujeme klávesnici pomocí jednoho řádku.

```
1.  public final class EditItemDialog extends AbstractFullScreenDialog {
2.      // ...
3.      public EditItemDialog(CellGridPanel form) {
4.          PokladnaViewController.requestKeyboardInput(jTextField1);
5.          PokladnaViewController.requestKeyboardNumericInput(jTextField2);
6.      }
7.      // ...
8.  }
```

Kód 3: Výsledek přesunutí registrace akce

```
PokladnaViewController.requestKeyboardInput(jTextField1,jTextField2,jTextField3,jTextField4,jTextF
ield5);
```

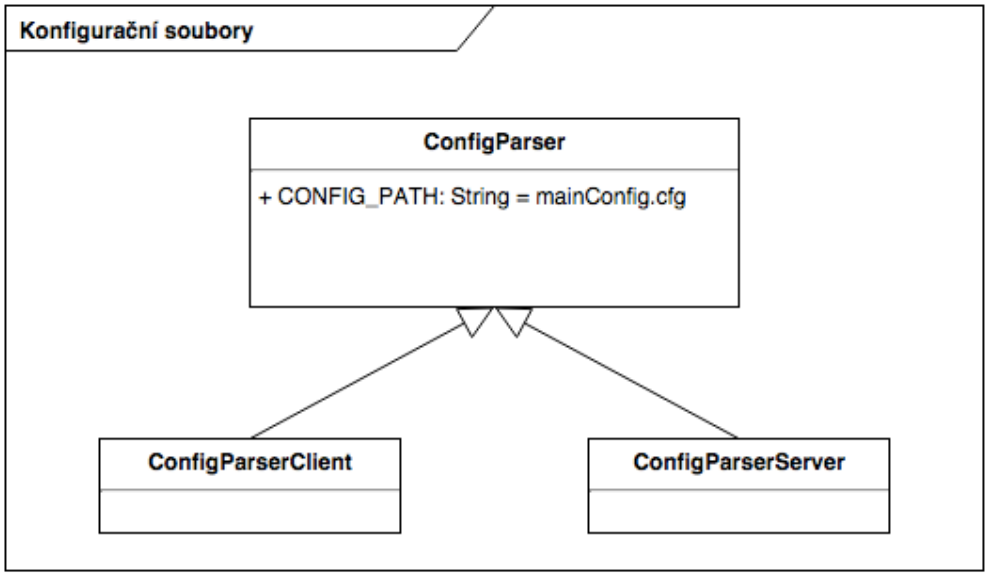
Kód 4: Ukázka registrace klávesnice pro více vstupů

2.2 Konfigurační soubory aplikace

Součástí aplikace cashbob jsou konfigurační soubory. V těchto souborech najdeme například hodnoty reprezentující cestu k tiskovým šablonám, nastavení lokalizace (čeština, angličtina), IP adresu zařízení, na kterém je spuštěn server a další. Konfigurační soubor slouží k nastavení vlastností počítačových programů a můžou být napsány formou prostého textu tak, aby je programátor nebo uživatel mohl jednoduše měnit. Nejčastěji se setkáme se strukturou, kdy jeden řádek reprezentuje jednu hodnotu zapsanou ve formě (Klíč=hodnota)[5].

2.2.1 Analýza stávajícího řešení

V aplikaci existovala abstraktní řída *ConfigParser.java* (UML ukázka na obrázku č. 1), která definovala metody pro operace s konfiguračním souborem, getProperty(key), *saveProperty(key,value)*, *makeDefaultConfig()*.



Obr. 1: Původní implementace čtení konfiguračních souborů

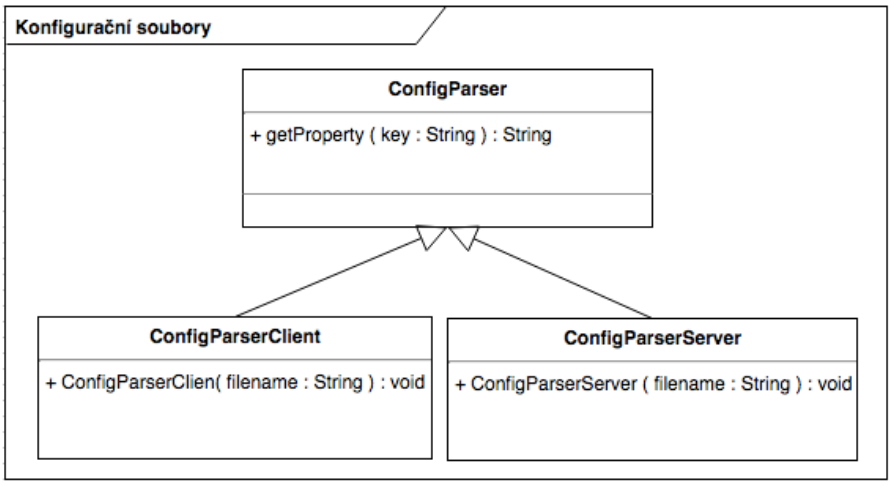
Konkrétní implementace tříd (*ConfigParserClient*, *ConfigParserServer*) měly v těle definované proměnné, mapované na proměnné z konfiguračního souboru (lokalizace, registrační adresa, ...) a jejich gettery a settery. Místo toho, aby se v kódu při potřebě získat hodnotu z konfiguračního souboru používala základní metoda *getProperty(key)*, se používaly gettery, které v těle volaly právě tuto metodu - *getProperty(key)*. Docházelo tedy k duplicitě zbytečného kódu.

Dále abstraktní třída přímo definovala jméno každého konfiguračního souboru na „mainConfig.cfg“ způsobující nejasné určení, do kterého konfiguračního souboru má uživatel doplnit případné nové hodnoty.

Poslední problém souvisí s kapitolou č.6 [6]. V okamžiku, kdy se aplikace spustí pomocí jednoho souboru, dojde z důvodu stejného pojmenování, k přepsání jednotlivých konfiguračních souborů - nejdříve vznikne soubor pro server a následně pro klienta.

2.2.2 Návrh řešení a implementace

Pro jednoduchost a čitelnost jsem ponechal pouze základní metody existujícího řešení z třídy *Properties*. Nyní pro získání hodnoty zavoláme metodu *getProperty (key:String)* na konkrétní třídě. Použitím jedné obecné metody jsme se zbavili zbytečného generování getterů viz [2.2.1]. Název konkrétního konfiguračního souboru by měla určit až konkrétní implementace. Zde tedy navrhuji delegovat rozhodnutí až do dědících tříd. Toto povede k tomu, že všechny konfigurační soubory budou mít jiné jméno a v okamžiku, kdy se aplikace bude spouštět z jiného kontextu, nebude docházet k přepisování jednotlivých souborů. Změna struktury je zobrazena na obrázku č. 2[6].



Obr. 2: Nová struktura konfiguračních tříd

2.3 Validace vstupních polí

2.3.1 Analýza

V aplikaci neexistoval jednotný způsob validace vstupních hodnot. Uživatel se o špatně vyplněném poli dozvěděl až v okamžiku zpracování formuláře. Ideální způsob by byl, kdyby sám prvek informoval o tom, zda je, či není správně vyplněn. Příkladem je vyplnění pole pro email. Pokud uživatel zadává špatný formát emailu, samo pole již může informovat o špatném formátu vstupních hodnot, například výstražným orámováním.

2.3.2 Implementace

Vytvořil jsem interface *Controllable* (kód č. 5), definující metody potřebné pro obecný vstupní prvek.

```
1.  public interface Controllable {
2.      public String getValue();
3.      public String getName();
4.      public void setValidator(InputValidator validator);
5.      public boolean isCorrect();
6.      public void setText(String text);
7.  }
```

Kód 5: Interface Controllable pro vstupní pole

Dále jsem vytvořil GUI[i] komponentu *ControllableInput,* která přebírá funkcionalitu od třídy *jTextField.java* reprezentující vstupní pole a zároveň ji rozšiřuje o funkcionalitu definovanou novým rozhraním *Controllable*. Třída *ControllableInput* umožňuje použití přiřazeného validátoru, který při každé změně vstupní hodnoty provede její validaci. Pokud není validátor přiřazen, má třída svůj vlastní implicitní validátor, který defaultně vrací hodnotu TRUE – hodnota je v pořádku.

```
1.  public class ControllableInput extends JTextField implements Controllable {
2.      private InputValidator validator;
3.      private final ControllableInput self;
4.      public ControllableInput(String name){
5.          this.setName(name);
6.          this.self = this;
7.          validator = new InputValidator(this) {
8.                      @Override
9.                      public boolean _validate(ControllableInput textField) {
10.                         return true;
11.                     }
12.             };
13.         this.getDocument().addDocumentListener(new DocumentListener() {
14.                     @Override
15.                     public void removeUpdate(DocumentEvent e) { validator.validate();}
16.                     @Override
17.                     public void insertUpdate(DocumentEvent e) {validator.validate();}
18.                     @Override
19.                     public void changedUpdate(DocumentEvent e) {validator.validate(); }
20.             });
21.     }
22.     @Override
23.     public String getValue() { return this.getText();}
24.
25.      @Override
26.      public void setValidator(InputValidator validator) { this.validator = validator;}
27.
28.      @Override
29.      public boolean isCorrect() {return validator.validate(this);}
30. }
```

Kód 6: Třída reprezentující samokontrolující prvek

i GUI – grafické uživatelské rozhraní, které umožní ovládat počítače pomocí grafických komponent.

Pro použití v aplikaci stačí mít tedy připraveno několik validátorů, které ošetří vstupní hodnoty, se kterými se v aplikaci pracuje (*EmailValidator*, *NumberValidator*, *EmptyStringValidator*,...) a následně tyto validátory přiřadit konkrétním vstupním polím. Při zpracování formuláře se následně zavolá na vstupním poli metoda *isCorrect()*, která vrací *bool* hodnotu informující o správném vyplnění.

Pro vytvoření nového validátoru stačí vytvořit novou třídu dědící od třídy *InputValidator* (ukázka č.7), a implementovat metodu _validate(), která provede kontrolu vstupní hodnoty. Pokud je třeba prvek v případě chyby dekorovat jiným způsobem, než červeným orámováním, stačí přepsat metodu *setError()*. Ukázka emailového validátoru je v ukázce č. 8.

```
1.  public abstract class InputValidator {
2.      protected final ControllableInput component;
3.
4.      public InputValidator(ControllableInput component) { this.component = component; }
5.
6.      public boolean validate() {
7.          boolean isValid = _validate(component);
8.          inform(isValid);
9.          return isValid;
10.     }
11.     protected abstract boolean _validate(ControllableInput component);
12.
13.     public void inform(boolean value) {
14.         if (value) {
15.             resetError();
16.         } else {
17.             setError();
18.         }
19.     }
20.     public void setError() {
21.         component.setBorder(BorderFactory.createLineBorder(Color.RED));
22.     }
23.     public void resetError() {
24.         component.setBorder(BorderFactory.createLineBorder(Color.BLACK));
25.     }
26. }
```

Kód 7: Obecný validátor

```
1.  public class EmailValidator extends InputValidator {
2.
3.      public EmailValidator(ControllableInput component) {
4.          super(component);
5.      }
6.
7.      @Override
8.      protected boolean _validate() {
9.          return StringUtils.isValidEmail(component.getText());
10.     }
11. }
```

Kód 8: Ukázka emailového validátoru

2.4 Testování

Při každém buildování aplikace docházelo ke spuštění celé sady původních testů, které ověřují funkčnost aplikace. Dále jsou v elektronických přílohách akceptační testy, které ověřují základní funkcionalitu celé aplikace.

2.5 Shrnutí

Po úpravě lze klávesnici registrovat pomocí jednoho řádku. Konfigurační soubory jsou nyní různě pojmenované (*ClientCofing* a *ServerConfig*) a pro získání hodnoty se používá obecná metoda *getProperty(key)*. Pro validaci lze využít samokontrolující prvek (ControllableInput), který po přiřazení validátoru sám informuje uživatele o špatně zadané vstupní hodnotě.

3 Vyhodnocení výsledků usability testování

Během implementace a tvorby uživatelského prostředí, není vývojář schopen odhadnout chování reálných uživatelů. Abychom během vývoje ověřili, zda je používání aplikace pro uživatele srozumitelné, je pořeba podrobit aplikaci testování použitelnosti. Toto testování například ukáže, jak uživatelé reagují na jednotlivé prvky v uživatelském prostředí nebo zda jim některé úkony (vytvoření účtu, placení účtu, …) nedělají potíže.

Ve spolupráci s Bohuslavem Koukalem proběhlo testování aplikace v usability labu[ii] [7]. Výsledkem tohoto testování je zpracovaná výstupní zpráva nalezených problémů a jejich stupeň závažnosti, uvedená v tabulce č. 1. Každý nález má své číslo, popis problému, prioritu, číslo účastníka testu u kterých se daný problém objevil a návrh řešení od autora tabulky.

V této kapitole se budu věnovat analýze nalezených problémů, návrhu jejich řešení a následné implementaci.

V následující kapitole - 3.1 budu analyzovat jednotlivé problémy a budu předkládat různé možnosti řešení. V další v kapitole 3.2 předložím vybrané finální řešení, která dále budu realizovat.

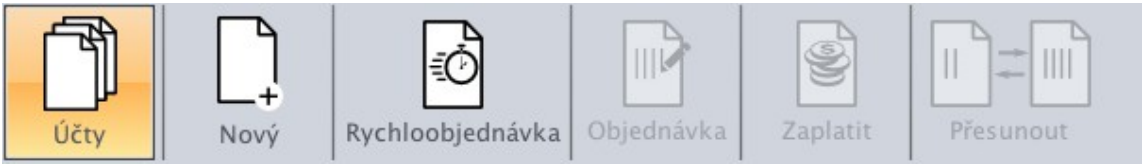
ii Testovací místnost, ve které probíhá testovaní interakce aplikace s uživatelem. Zaznamenaný průběh testování se zpětně vyhodnocuje a výsledkem je zpráva o nalezených problémech.

Prob. č.	Popis problému	Priorita	Úč.	Řešení
1.	Při kliknutí na rychloobjednávku se vytváří nové účty, což je problém při kliknutí vedle nebo při náhodném proklikávání při seznamování se s aplikací.	Vysoká	1	Prázdnou rychloobjednávku neuložit.
2.	Při přidávání nového účtu je první osoba a ne stůl. To účastníky mate, nevšimnou si, že se dá stůl přidat nebo osobu vyplní vždy stejnou a vznikne pak problém s duplicitními názvy.	Střední	1, 2, 4	Dát jako první stůl a ne osobu (pokud bude osoba zachována).
3.	Chyba při opakovaném prokliknutí do menu (kořalky).	Vysoká	1	Analyzovat a opravit chybu.
4.	Uživatel si není jistý, zda objednávku uložil správně.	Střední	1, 3	Deaktivovat tlačítko objednat, pokud je objednávka prázdná.
5.	Při kliknutí na Rychloobjednávku se skočí na Objednávku, což uživatel považuje za chybu.	Vysoká	1, 4	Při kliknutí na Rychloobjednávku zvýraznit toto tlačítko a ne tlačítko Objednávka.
6.	V dialogu, kde se nic nevybírá, se tlačítko jmenuje zaplatit vybrané. Uživatel neví, co to znamená.	Nízká	1, 4	Přejmenovat tlačítko na Zaplatit.
7.	Při placení účtu se otáčí pořadí položek.	Nízká	1	Pořadí položek neměnit.
8.	Při přesouvání mezi účty je potřeba nejdřív vytvořit nový účet, pak se přepnout na existující, pak dát přesunout a vybrat nově vytvořený.	Střední	1, 2, 3, 4	V poli Cílový účet nabídnout uživateli Nový účet.
9.	Nejde zaplatit víc účtů dohromady, je potřeba je nejdřív sloučit.	Vysoká	1, 2, 4	V dialogu zaplatit umožnit přidat k zaplacení více účtů nejlépe podle workflow participanta 1 (video 8:40).
11.	Editace položek není v Pokladně aktivní, dokud se nevybere nějaký existující účet.	Střední	1	Učinit ji aktivní i pokud není vybrán žádný účet.
12.	Při Editaci položek se nedá do menu dostat dvojklikem, tlačítko na otevření menu je daleko a je nevýrazné.	Střední	1, 2, 4	Vymyslet způsob prokliku, který bude intuitivnější, popř. kontextovější (nabídne se v místě kliknutí a ne na druhé straně obrazovky).
13.	Když uživatel neuloží objednávku a překlikne někam jinam, celá objednávka se bez varování smaže.	Vysoká	2, 3	Varovat uživatele, zda opravdu chce objednávku smazat.
14.	Uživatel sám od sebe nezaklikne Uzavřít účet, pokud zůstane prázdný. Zaplacené účty mu pak zůstávají otevřené.	Střední	2, 4	Pokud se neplatí všechno, checkbox neukazovat. Pokud se platí všechno, pak formulovat otázku opačně – Nechat tento účet otevřený.

Tabulka 1: Výstupní zpráva o nalezených problémech

3.1 Analýza nalezených problémů

3.1.1 Vytvoření rychloúčtu



Obr. 3: Hlavní menu aplikace

V hlavním menu aplikace se nachází položka „Rychloobjednávka“. Po kliknutí na toto tlačítko se vytvoří nový účet s předvyplněným názvem ve formátu HH:mm:ss d/m (např.: 12:23:44 7/2) a přejde se na obrazovku s objednávkou. Tlačítko je určené pro rychlé vytvoření účtu a objednávky v případě, kdy host přijde na bar a chce si objednat z nabídky a rovnou zaplatit. Při každém stisku tohoto tlačítka dochází k vytvoření nového účtu a tedy při opakovaném stisku či překliku se uživateli vytvářejí a hromadí nové prázdné účty bez jakéhokoli upozornění.

Dále není tlačítko správně pojmenováno. Toto tlačítko vytvoří účet s předvyplněným názvem a tedy se jedná o rychlé vytvoření účtu, nikoliv objednávky. Tlačítko by spíše mělo být pojmenováno „rychloúčet“.

Návrhy řešení:

1. Smazání prázdného účtu při odchodu z obrazovky Objednávka

Představme si situaci, kdy se zákazník rozhodne, že si ke stolu nakonec sedne a nejedná se tedy o rychloúčet. Účet by stačil přejmenovat, popřípadě doplnit další infomace jako je stůl či poznámka, ale účet by se smazal, protože obsluha prozatím nic neobjednala. Proto je toto řešení nevhodné.

2. Použití rozbalovacího menu

Možným řešením, jak předejít opakovanému nebo nechtěnému stisknutí, je schovat možnosti pro vytvoření nového účtu do rozbalovacího menu. To by znamenalo, že v hlavním menu zůstane tlačítko „nový účet“ a po stisku tohoto tlačítka se uživateli zobrazí tlačítko „standardní účet“ a „rychloúčet“.

Schováním možností do rozbalovacího menu, zamezíme nechtěnému, či opakovanému kliknutí, ale vznikne nám problém s poskakováním ostatních položek menu. Vedle tlačítka se nacházejí další kontrolní prvky, a to tlačítko pro objednávku, platbu účtu a přesunutí položek mezi účty. Při každém zobrazení možností pro tvorbu účtu se rozbalovací menu zobrazí vpravo/vlevo a dojde k odskočení ostatních prvků.

Položky menu by měly být stále na stejném místě, aby uživatelé nebyli zmatení ze změny umístění položek. Proto je tato možnost také nevhodná [8].

3. Odebrání tlačítka „rychloúčet“ z menu

Jak již bylo řečeno výše, normální účet a rychloúčet se liší pouze předvyplněným názvem, a proto by se mohla možnost vytvoření rychloúčtu přesunout na obrazovku pro vytvoření standardního účtu. Na této obrazovce by se uživatel rozhodl, jakým způsobem bude dále pokračovat.
V tomto případě existuje několik možných scénářů:

1. Na obrazovce vznikne vedle původního tlačítka „Vytvořit účet“ nové tlačítko - „Vytvořit rychloúčet“
2. Stávající tlačítko „vytvořit účet“ bude mít v případě nevyplněného názvu popisek „Vytvořit rychloúčet“ a v případě vyplněného názvu se popisek změní na „Vytvořit účet“ a naopak.
3. Stávající název tlačítka zůstane a vnitřní logika se zachová sama podle vyplněného, popřípadě nevyplněného názvu účtu.

3.1.2 Vytvoření nového účtu

Na obrazovzce s formulářem pro vytvoření nového účtu existuje pouze jedno povinné pole a to je název účtu. Ostatní pole jako je osoba, stůl a poznámka jsou nepovinné údaje, které slouží pouze pro upřesnění informace o účtu.

Pole osoba umožňuje vybrat například „štamgasta“ či zaměstnance restaurace, který může čerpat benefity (platit kreditem). Tato funkce však není momentálně podporována, a proto je tato informace zbytečná stejně jako informace „sleva“.

Návrh řešení:

1. Přeorganizovat pořadí vstupních polí dle důležitosti a implementované funkcionality v tomto pořadí: *Název, Stůl, Osoba, Poznámka , Sleva*

2. Změnit pořadí vstupních polí dle návrhu 1 a dále odstranit pole „osoba“ a „sleva“ dokud nebude implementována funkcionalita pro tyto možnosti.

3.1.3 Chyba s prokliknutím do menu

Tato chyba byla odstraněna v průběhu práce na projektu v předmětu A7B36SI2, kdy paralelně probíhala práce na projektu a testování v usability labu. Tento problém byl odstraněn v rámci opravy jiných chyb.

3.1.4 Trvale aktivní tlačítko „Objednat“

Na obrazovce s objednávkou pod panelem souhrnu nové objednávky, je umístěno tlačítko „Objednat“, kterým obsluha potvrzuje připsání zvolených položek na účet. Toto tlačítko je aktivní i po úspěšné objednávce, kdy nejsou vybrány žádné další položky.

Návrh řešení:

1. V okamžiku, kdy nejsou zvoleny žádné nové položky v detailu nové objednávky, bude tlačítko neaktivní.

3.1.5 Označení rychloobjednávky jako objednávka

V kapitole [3.1.1] popisuji rozdíl mezi rychloúčtem (chybně označeným jako „rychloobjednávka“) a normálním účtem. Po kliknutí se automaticky vytvoří účet a následuje přechod na obrazovku s objednávkou, aby obsluha mohla objednat požadované položky. Proto je v menu aktivně označená položka „objednávka“.

Návrh řešení:

1. Protože v kapitole [3.1.1] navrhuji schovat tlačítko „rychloobjednávka“, nebude nadále tento prvek trvale viditelný, a proto nebude docházet ke zmatečnému označení v hlavním menu.

3.1.6 Tlačítko „zaplatit vybrané“

Při placení uživatel vybere položky z účtu, které chce zákazník zaplatit a potvrdí platbu. Uživateli se zobrazí dialog s celkovým souhrnem platby, kde je umístěno tlačítko s popiskem „Zaplatit vybrané“. Tento text je matoucí, protože v souhrnném dialogu se již žádný výběr neprovádí - v tomto souhrnu jsou již vybrané položky z předchozího kroku. Toto tlačítko má pouze nešťastně zvolený popis, protože po stisku tlačítka dojde k zaplacení položek.

Návrh řešení:

1. Přepsat text tlačítka na „Zaplatit“.

3.1.7 Rotace pořadí položek

Klikne-li uživatel opakovaně na záložku „Zaplatit“, pokaždé uvidí zpřeházené pořadí položek v detailu účtu. Na uživatele tento jev může působit tak, že se mění stav účtu, aniž by on sám provedl nějakou akci.

*Návrh ř*ešení:

1. Při vkládání položek do detailu účtu se budou řadit podle zvoleného atributu (př. ID).

3.1.8 Přesouvání položek mezi účty

Představme si situaci, kdy si zákazník přeje rozdělit účet na dva. Obsluha nemůže přímo zvolit možnost přesunout na jiný účet, protože cílový účet prozatím neexistuje. Musí tedy vytvořit nový účet, poté se musí přepnout na přehled všech účtů, vybrat zdrojový účet a teprve poté může přejít na obrazovku pro přesun položek.

Pro zjednodušení celé situace by bylo vhodné, aby aplikace umožňovala vytvořit nový účet přímo na obrazovce pro přesun položek.

Návrh řešení:

1. Přidat možnost vytvořit účet na obrazovku s přesunem položek a dále pro větší přehlednost přidat přehled aktuálních položek na cílovém účtu.

3.1.9 Platba více účtů dohromady

Tento problém není relevantní. Každý účet je samostatná jednotka, která musí být zaplacena samostatně. Existuje-li nějaký důvod, proč by měly být dva účty placeny dohromady, může obsluha využít přesunutí položek z jednoho účtu na druhý, popřípadě vytvořit účet nový a na něj přesunout požadované položky.

3.1.10 Nastavení menu není aktivní

Neaktivní možnost „editace menu“ je chyba, která vznikla v důsledku změny aplikace v předmětu A7B36SI2, kdy existovalo několik editačních možností, které byly vázány na zvolený účet. Při zneaktivnění těchto možností byla omylem zakázána i možnost pro editaci menu.

Návrh řešení:

1. Možnost nastavení menu bude trvale aktivní, protože není závislá na zvoleném účtu.

3.1.11 Chybějící možnost dvojkliku

Pokud uživatel edituje menu – nabídku položek, nemůže se prokliknout do jednolivých podkategorií pomocí dvojkliku. Uživatel musí označit příslušnou kategorii a v levém horním rohu pomocí tlačítka „Zobrazit menu“ vstoupit do dané podkategorie. Protože barva tlačíka je totožná s pozadím aplikace a navíc není trvale aktivní pokud se nevybere buňka s podkategorií, je tlačítko velice lehce přehlídnutelné.

Návrh řešení:

1. Implementovat vstup do kategorie pomocí dvojkliku
2. Zvýraznit tlačítko pro vstup do kategorie a přesunout ho nad matici položek.

3.1.12 Smazání objednávky

Nepotvrzená objednávka se při přepnutí do jiné nabídky bez varování vymaže a uživatel musí následně celou objednávku vytvořit opakovaně. V okamžiku, kdy obsluha vyřizuje velkou objednávku, hrozí, že při překliknutí přijde o mezistav obejdnávky. Aplikace by měla umožňovat, aby se obsluha mohla vrátit k rozpracované objednávce.

Návrh řešení:

1. Zobrazit upozornění o nepotvrzené objednávce v případě přechodu na jinou obrazovku aplikace.

3.1.13 Nezavírání zaplacených účtů

Při platbě účtu si uživatel může zvolit, zda účet chce ponechat otevřený, či ho po zaplacení uzavřít. V drtivé většině případů se účet po zaplacení zavírá, proto by měla být výchozí hodnota nastavena na „Uzavřít účet“. V případě, kdy se jedná o jiný případ, si obsluha sama nastaví, aby se účet neuzavíral.

Návrh řešení:

1. Nastavit výchozí hodnotu na „Zavřít účet po zaplacení“

3.2 Výstup analýzy

Provedením analýzy jsme zjistili, které opravy či úpravy je nutné podniknout. Návrh řešení byl u každého nalezeného problému diskutován se zadavatelem projektu a u problémů, kde bylo identitifikováno více možností řešení, jsme se zadavatelem projektu vybrali nejvhodnější řešení. Konečné zadání úkolů:

1. Zadání - [3.1.1] Tlačítko „rychloobjenávka“ nebude nadále v hlavním menu aplikace. Možnost vytvořit „rychloobjednávku“ se přesune na obrazovku pro vytvoření standardního účtu, kde se bude měnit text tlačítka, dle vyplněného názvu. Dále dojde k sjednocení terminologie. „Rychloobjednávka“ se přejmenuje na „Rychloúčet“.

2. Zadání - [3.1.2] Pořadí vstupních polí se změní tak, aby název účtu byl na prvním místě. Pole pro výběr osoby a slevy se odstraní, protože nyní neexistuje využití pro tyto informace.

3. Zadání - [3.1.4] Pokud nejsou vybrány žádné položky pro přidání na účet, bude tlačtítko „objednat“ neaktivní.

4. Zadání - [3.1.5] Problém bude vyřešen bodem [1] této podkapitoly. Tlačítko nadále nebude v hlavním menu aplikace.

5. Zadání - [3.1.6] Tlačítko bude přejmenováno na „Zaplatit“.

6. Zadání - [3.1.7] List položek bude seřazen podle ID tak, aby nedocházelo k rotaci položek.

7. Zadání - [3.1.8] Na obrazovce s přesunem položek mezi jednotlivými účty bude tlačítko pro vytvoření nového účtu. Po kliknutí na toto tlačítko se zobrazí vstupní pole pro název nového účtu. Dále pro větší přehlednost bude na obrazovce zobrazen detail cílového účtu, aby obsluha věděla, které položky se na tomto účtu již nacházejí.

8. Zadání - [3.1.10] Možnost „nastavení menu položek“ v hlavním menu bude stále aktivní.

9. Zadání - [3.1.11] Bude implementována možnost dvojkliku pro vstup do menu.

10. Zadání - [3.1.12] Aplikace uživatele upozorní o rozpracované objednávce při odchodu na jinou obrazovku.

11. Zadání - [3.1.13] Výchozí hodnota pro uzavírání účtu při placení bude nastavana na „Uzavřít, pokud je účet prázdný“.

3.3 Implementace oprav a vylepšení

V této kapitole uvedu pouze významné změny a implementace nové funkcionality. Problémy, u kterých jsem řešil drobné změny, například odmazání nebo změnu textu, nebudu uvádět.

3.3.1 Odstranění tlačítka - rychloobjednávka

Vstupnímu poli pro název účtu jsem přiřadil listener, který kontroluje zadanou hodnotu tohoto pole. Pokud je vstupní pole prázdné, má tlačítko pro vytvoření účtu hodnotu „Vytvořit rychloúčet“ a v opačném případě „Vytvořit účet“. Ukázka implementace je uvedena v ukázce č. 9.

```
1.  jTextFieldName.getDocument().addDocumentListener(new DocumentListener() {
2.
3.                  @Override
4.                  public void removeUpdate(DocumentEvent e) {
5.                      check(jTextFieldName.getText());
6.                  }
7.
8.                  @Override
9.                  public void insertUpdate(DocumentEvent e) {
10.                     check(jTextFieldName.getText());
11.                 }
12.
13.                 @Override
14.                 public void changedUpdate(DocumentEvent e) {
15.                     check(jTextFieldName.getText());
16.                 }
17.
18.                 private void check(String str){
19.                 if(StringUtils.isNullOrEmpty(str)){
20.                         createAccountBtn.setText(CREATE_QUICK_ORDER);
21.                     }else{
22.                         createAccountBtn.setText(CREATE_BUTTON_LABEL);
23.                     }
24.             }
25.             });
```

Kód 9: Kontrola hodnoty vstupního pole

3.3.2 Trvale aktivní tlačítko „objednat“

Detail objednávky je měněn z několika různých míst – smazání objednávky, odstranění jednotlivých položek, přidání nové položky nebo při konečném potvrzení objednávky tlačítkem „objednat“. Při každé této akci dochází k překreslení komponenty pro detail účtu. Abych nemusel přidávat požadovanou funkcionalitu do cízí komponenty, vytvořil jsem vlastní Listener, který bude vyvolán v metodě *repaint()* komponenty detailu účtu, náhled v ukázce č. 10.

```
1.  class OrderPanel extends Jpanel{
2.   ...
3.   @Override
4.     public void repaint() {
5.        super.repaint();
6.        Object [] listeners = this.listenerList.getListenerList();
7.        for(int i = 0; i < listeners.length; i = i+2){
8.             if(listeners[i] == RepaintListener.class){
9.                  ((RepaintListener)listeners[i+1]).onRepaint(null);
10.            }
11.       }
12.   }
```

Kód 10: Volání vlastního listeneru v metodě reapint() komponenty OrderPanel

Následně ve formuláři pro objednávku registruji konkrétní listener pro kontrolu vyplněné objednávky, který bude aktivovat tlačítko pro potvrzení objednávky, ukázka č.11.

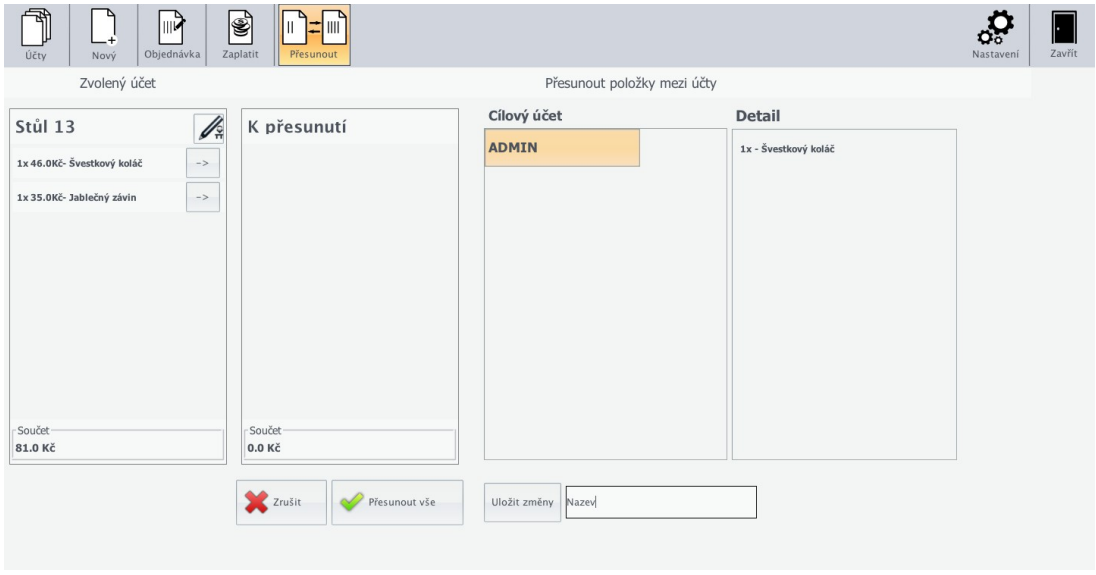
```
1.  OrderPanel.addCustomEventListener(new RepaintListener() {
2.          @Override
3.          public void onRepaint(Event e) {
4.
5.                  if(orderPanel.ordersPanel.getOrders().isEmpty()){
6.                          jButtonOrder.setEnabled(false);
7.                  }else{
8.                          jButtonOrder.setEnabled(true);
9.                  }
10.         }
11. });
```

Kód 11: Registrace vlastní akce při překreslení formuláře

3.3.3 Vytvoření nového účtu u přesunu položek

Na obrazovce pro přesun položek mezi účty byla přidána možnost vytvořit nový účet. Po vyplnění názvu se nový účet přidá na první místo v seznamu cílových účtů a automaticky se označí. Obsluze stačí zvolit položky pro přesun a potvrdit.

Jako další vylepšení jsem přidal na pravou stranu obrazovky náhled detailu cílového účtu. Obsluha nyní vidí, které položky jsou již na cílovém účtu. V předchozí verzi se náhled nezobrazoval a po provedení přesunu nebyla zpětná vazba o přesunutí. Nyní obsluha vidí, které položky se přesunuly nebo přidaly. Výsledný vzhled obrazovky můžeme vidět na obrázku č.4



Obr. 4: Obrazovka pro přesun položek mezi účty

3.3.4 Implementace dvojkliku

Stávající řešení nepočítalo s využitím dvojliku pro vstup do menu i když je hlavní modul přizpůsoben dotykovému ovládání. Editor menu pracoval pouze s označením dané buňky a následné otevření menu bylo provedeno špatně viditelným tlačítkem, které bylo schované v levém rohu.

Provedl jsem rozšíření metody *menuClicked()* pro umožnění dvojkliku. Deklaroval jsem třídní proměnnou pro počítání kliknutí na položku menu.

Protože editor umožňuje jedno kliknutí pro získání informací o buňce, je potřeba zajistit, aby se počítadlo samo vyresetovalo, neklikne-li uživatel podruhé. Pro tuto potřebu jsem definoval spouštěč, který po *500ms* vyresetuje počítadlo kliknutí. Náhled implementace je v ukázce č. 12.

```
1.  @Override
2.  public void menuClicked(long menuId, String menuName, int x, int y)
3.  {
4.          // ...
5.          menuClicked++;
6.          if(menuClicked == 2 )  {   // ... akce() return; }
7.          Timer t = new Timer("DoubleClickTimer");
8.          t.schedule(new TimerTask() {
9.              @Override
10.             public void run() {
11.                 menuClicked = 0;
12.             }
13.         }, 500);

14. }
```

Kód 12: Implementace dvojkliku

3.3.5 Upozornění o rozpracované objednávce

Původní myšlenka byla taková, že aplikace upozorní uživatele při odchodu na jinou obrazovku, pokud má rozpracovanou objednávku. Nepodařilo se mi ale úspěšně kontrolovat a ohlídat okamžik, kdy hlavní rám vyměňuje jednotlivé obrazovky. Každá obrazovka (nový účet, přehled účtů, objednávka, …) je reprezentovaná pomocí třídy *JPanel*, která nemá události typu *onShow*, *onClose*, aj. pro ohlídání, kdy je nový panel zobrazen. Využil jsem tedy interface *AncetorListner*, který definuje pro moje účely důležité následující dvě metody.

Metody *ancestorAdded(event)* a *ancestorRemoved(event),* které reagují na to, pokud zdroj nebo jeho předci jsou (ne)zviditelněny metodou *setVisibile(bool)* nebo pokud je komponenta přidána do hierarchie komponent.

Použití metody ancestorRemoved(event) by bylo ideální řešení, jak při odchodu na jinou obrazovku upozornit uživatele o rozpracované objednávce, avšak tato metoda je zavolána až poté, co je odstraněn původní panel s objednávkou a nelze tedy zamezit zobrazení další zvolené obrazovky uživatelem. Pro konečné řešení jsem tedy použil

metodu anestorAdded(event), kdy při opětovném návratu na obrazovku s objednávkou je kontrolována historie objednávek a v případě nalezení rozpracované objednávky se zobrazí dialog, zda se má tato objednávka použít (ukázka č. 13). Pamět pro rozpracované objednávky je reprezentovaná hashovací tabulkou, která si pamatuje objednávky pro konkrétní účty.

```
26.  this.addAncestorListener(new AbstractAncestorListener() {
27.
28.     @Override
29.     public void ancestorAdded(AncestorEvent event) {
30.       if(LeftPanelController.getOrdersHistory().get(PokladnaViewController.getSelectedAccountId()) != null){
31.            String[] options = {"Ano","Ne"};
32.             int reply = JOptionPane.showConfirmDialog(null, "Máte rozpracovanou objednávku, chcete ji
           použít?","Rozpracovaná objednávka" ,JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE);
33.             if(reply == JOptionPane.YES_OPTION){
34.                  SwingUtilities.invokeLater(new Thread(){
35.                        @Override
36.                        public void run() {
37.                          try {
38.                              sleep(100);
39.                              orderPanel.ordersPanel.removeOrders();
40.                              for(OrderLine line : LeftPanelController.getOrdersHistory()
41.                                    .get(PokladnaViewController.getSelectedAccountId()))
42.                           {
43.                                 orderPanel.ordersPanel.addOrder(line.getCount(), line.getName(), line.getPrice(), line.getId());
44.                                 orderPanel.ordersPanel.setButtonText("-");
45.                                 orderPanel.ordersPanel.setButtonVisibility(true);
46.                                 orderPanel.repaint();
47.                             }
48.                           } catch (InterruptedException e) {
49.                                 e.printStackTrace();
50.                            }
51.                           }
52.                  });
53.               }
54.              else{
55.                 LeftPanelController.getOrdersHistory().remove(PokladnaViewController.getSelectedAccountId());
56.              }
57.       }
58.     }
59.  });
```

Kód 13: Zobrazení dialogu o rozpracované objednávce, při změně obrazovky

3.4 Testování

V tuto chvíli by bylo vhodné provést další testování v usability labu, avšak další testování je nad rámec této práce. Proto v elektronických přílohách najdeme akceptační testy, které mají za úkol prověřit hlavní funkcionalitu aplikace. Dále můžeme vidět testování funkčnosti na natočených videotutoriálech, které předvádějí funkce aplikace a jejich scénáře byly z velké části inspirovány právě akceptačními testy.

3.5 Shrnutí

Podařilo se opravit všechny nalezené probémy a přidat novou funkcionalitu:

- pamět pro rozpracované objednávky
- vytvoření nového účtu na obrazovce pro přesun položek
- detail účtu na obrazovce pro přesun položek
- vstup do menu pomocí dvojkliku

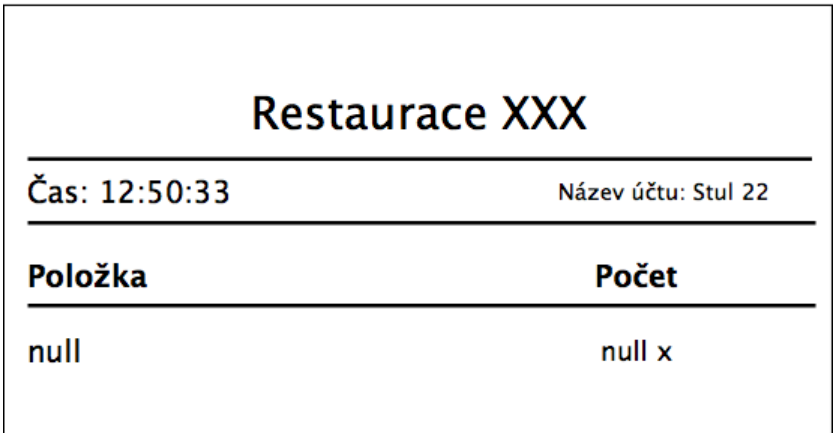
4 Tisk nových objednávek

Jedním z úkolů nad rámec zadání této práce bylo zkontrolovat funkčnost tisku účtenek a rozšířit o tisk nových objednávek. Tuto novou funkci budou zaměstnanci restaurace využívat pro lepší koordinaci práce. V případě větší objednávky si obsluha může vytisknout souhrn objednávky a při přípravě pokrmů, nápojů a jiných položek si může postupně odškrtávat již připravené položky a kontrolovat, zda na nic nezapomněla.

4.1 Analýza tisku

Pro úspěšný tisk je potřeba mít připravené tiskové šablony, které definují výsledný vzhled vytištěného dokumentu. Tyto šablony se následně musí distribuovat s aplikací, aby byly k dispozici při tisku.

Pro vytváření a tisk šablon se v aplikaci využívá řešení od JasperSoft [9]. Poskytovatel této knihovny na svých stránkách nabízí editor šablon jaspersoft iReport Designer [10], který slouží pro návrh a export šablon do požadovaného formátu. Podle následujícího návrhu (obrázek č. 5) budu připravovat šablonu pro tisk.



Obr. 5: Návrh šablony pro tisk nové objednávky

4.2 Implementace tisku objednávek

Pomocí výše zmíněného editoru jsem vytvořil tiskovou šablonu *order.jasper* a umístil do projektové složky *templates/*. Do formuláře pro objednávku *(CteateOrderForm.java)* jsem přidal pod detail nové objednávky tlačítko „Tisk objednávky“, které vyvolá tiskovou úlohu. (ukázka č. 14)

```
1.  private void printOrderBtnActionPerformed(java.awt.event.ActionEvent e){
2.      try {
3.
4.          List<OrderBeen> beens = new LinkedList<OrderBeen>();
5.          for(OrderLine line : orderPanel.ordersPanel.getOrders()){
6.              OrderBeen o = new OrderBeen();
7.              o.setCount(line.getCount());
8.              o.setName(line.getName());
9.              beens.add(o);
10.         }
11.
12.         SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
13.         Map<String,Object> parameters = new HashMap<String,Object>();
14.
15.         parameters.put("introText", "Objednávka k účtu");
16.         parameters.put("accountName", PokladnaViewController.getSelectedAccountName());
17.         parameters.put("printTime", sdf.format(new Date()));
18.
19.         Printer.printFromListToPrinter(parameters, beens, new File("templates/order.jasper"));
20.
21.         } catch (JRException e1) {
22.     PokladnaViewController.showErrorMsg("Nepodařilo se vytisknout objednávku.");
23.         }
24.  }
```

Kód 14: Tisk nové objednávky

Nastavení tisku

Po instalaci tiskárny jsem provedl její kalibraci přes dialog v OS Windows a tisk zkušební stránky. Dále jsem tuto tiskárnu nastavil jako výchozí [11]. Po provedení tohoto nastavení nevzniká žádné zpoždění tisku v důsledku hledání tiskárny. Problém se neobjevil ani v okamžiku, kdy jsem přepojil tiskárnu do jiného USB rozhraní. Vedoucím práce jsem byl upozorněn na to, že se vyskytovaly problémy s tiskem, resp. se spožděním tisku v případě, přepojení tiskárny do jiného USB rozhraní než přes které byla nainstalována.

4.3 Shrnutí

Původní tisk účtenek funguje a je implementována nová funkce pro tisk nových objednávek. Je důležité, aby připojená tiskárna byla zvolená jako výchozí pro konkrétní počítač, aby nedocházelo ke zpoždění tisku.

5 Online registrace

V této kapitole se budu zabývat rozšířením způsobu registrace aplikace. Cílem této práce je umožnit uživateli registraci, aniž by musel vyplňovat registrační formulář na webových stránkách produktu. Toto rozšíření je dalším krokem k tomu, aby se uživatel cítil komfortněji a nemusel kvůli registraci přecházet na internetové stránky produktu a tam vyplňovat formulář, který může vyplnit prostřednicvím aplikace. Ideální řešení je obrazovka, kde uživatel vyplní registrační email, své jméno a následně obdrží licenční klíč na uvedený email, který vloží do aplikace pro potvrzení registrace.

5.1 Analýza registrace

Stávající řešení počítá s offline registrací produktu (stanice není připojena do sítě s přístupem na internet), kde jsou pro úspěšné řešení nutné následující kroky. Uživatel po vypršení licence musí přejít na registrační obrazovku, následně zkopírovat licenční kód vygenerovaný aplikací, přejít na stránky produktu, zde vyplnit registrační formulář (vložit vygenerovaný licenční kód), následně vyzvednout licenční klíč v emailové schránce a jako poslední krok musí vložit zaslaný klíč do aplikace.

Pokud bude pracovní stanice připojena k internetové síti, může se celý proces zjednodušit tím, že aplikace nabídne uživateli možnost online registrace. Uživatel vyplní email a jméno a po přijetí registračního klíče emailem vloží tento klíč do aplikace.

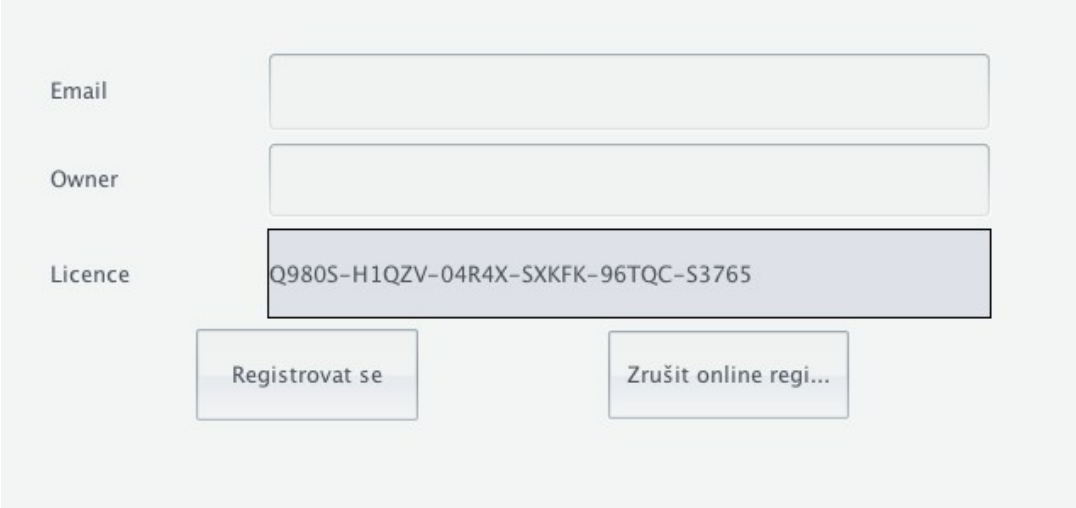
5.2 Implementace online registrace

Aby bylo možné provést registraci na serveru, vytvořil jsem v rámci dalšího bodu této práce webové stránky obsahující jednoduchý skript, který má na starosti online registraci. Online registrace bude dostupná na adrese *<doména>/online.php* a bude přijímat pouze HTTP POST [12] požadavky. Tato stránka vrací odpověď ve formátu JSON a to „{"status": STATUS, "error": TEXT-CHYBY}“ [13] (ukázka č. 15).

V aplikaci jsem vytvořil registrační třídu *RegistrationService,* která má na starosti komunikaci se serverem. První veřejnou metodou je *connectionAvailable(),* která vrací bool hodnotu, zda je, či není online registrace k dispozici. Metoda testuje, připojení k internetu tím, že zkusí vytvořit spojení se serverem, na kterém se provádí registrace. Pokud navázání spojení se serverem proběhne v pořádku, objeví se uživateli možnost přejít k online registraci. Následně uživatel vyplní jméno a email a potvrdí formulář, který můžeme vidět na obrázku č. 6. S těmito informacemi pracuje druhá veřejná metoda *register(args)*, která vytvoří HTTP POST požadavek, zpracuje odpověď a pokud není vrácený požadavek úspěšný, vyhodí metoda výjimku se zprávou ze serveru. Ukázka metod je znázorněna na ukázce kódu č. 16. Pokud online registrace proběhne v pořádku, aplikace zobrazí uživateli vstupní pole pro vložení licenčního kódu a po následném vložení je registrace úspěšně dokončena.

```
1.  $licenseKey = "";
2.  $emailSend = false;
3.  $info = "";
4.
5.  class Response{
6.      public  $status;
7.      public  $error;
8.      function  __construct($stat, $err){
9.          $this->status = $stat;
10.         $this->error = $err;
11.     }
12. };
13. if ($_SERVER['REQUEST_METHOD'] === 'POST') {
14.         $controller = new CreateLicenseController($_POST['holder'], $_POST['email'],
    $_POST['licenseCode']);
15.         $isValid = $controller -> validate();
16.         if ($isValid == "ok") {
17.             $licenseKey = $controller -> createLicenseKey();
18.         if($licenseKey!=""){
19.             $controller->sendEmail();
20.             $r = new Response(200,"");
21.             echo json_encode($r);
22.          }
23.         } else {
24.         $r = new Response(400,$isValid);
25.         $json = json_encode($r);
26.         echo $json;
27.         }
28. }
```

Kód 15: PHP script pro zpracování online registrace



Obr. 6: Formulář pro online registraci

```
1.      public boolean connectionAvailable() {
2.          try {
3.              URL url = new URL(URL_REGISTRATION);
4.              URLConnection connection = url.openConnection();
5.              connection.connect();
6.              return true;
7.          } catch (MalformedURLException ex) {
8.              return false;
9.          } catch (IOException ex) {
10.             return false;
11.         }
12.     }

13.     public void register(Controllable... components) throws IOException,JSONException {

14.         HttpClient client = HttpClients.createDefault();
15.         HttpPost post = _preparePost(components);
16.         HttpResponse response = client.execute(post);
17.         BufferedReader bfr = new BufferedReader(new
    InputStreamReader(response.getEntity().getContent()));
18.         String tmp = null;
19.         StringBuilder dom = new StringBuilder();
20.         while ((tmp = bfr.readLine()) != null) {
21.             dom.append(tmp);
22.         }
23.         String errors = _parseResponse(dom.toString());
24.         if (!StringUtils.isNullOrEmpty(errors)) {
25.             throw new IllegalArgumentException(errors);
26.         }
27.     }
```

Kód 16: Ukázka metod pro online registraci v aplikaci

5.3 Testování

Pro metodu zpracovávající přijatá data ve formátu. JSON jsem vytvořil ve třídě *RegistrationServiceTest* metodu, která testuje správné parsování. Metoda testuje žádnou odpověď, úspěšnou a neúspěšnou odpověď, špatný status kód a špatný formát. Dále přikládám v příloze akceptační test, který ověřuje funkčnost.

5.4 Shrnutí

Vedle klasické offline registrace nabídne aplikace v případě připojení pracovní stanice do sítě s přístupem na internet možnost registrovat aplikaci online. Tato možnost redukuje počet nutných kroků k registraci na tyto:

1. Vyplnění formuláře v aplikaci
2. *Vyzvednutí licenčního kódu v emailu*
3. *Vložení kódu do aplikace.*

6 Spuštění pomocí jednoho souboru

Cílem této kapitoly je upravit aplikaci tak, aby byla spustitelná jedním souborem. Uživatel by se měl při používání aplikace cítit pohodlně, a proto bychom ho neměli zatěžovat se spouštěním dvou souborů a navíc ještě v určitém pořadí. Prozatím se aplikace musí spouštět dvěma soubory. Prvním souborem se spustí server a následně dojde ke spuštění klienta druhým souborem. Ideální řešení je spuštění serveru a klienta pomocí jednoho souboru.

6.1 Analýza

Aplikace je rozdělena do několika samostatných modulů (Server, pokladna, skladový modul a modul pro plánování směn). Každý modul má své vlastní externí soubory (ikony, obrázky) používané v aplikaci. Všechny tyto soubory bude nutné umístit jednotným způsobem tak, aby při spuštění aplikace z jiného kontextu nedocházelo k chybám v důsledku nenalezení těchto souborů.

Dále bude potřeba zajistit, aby byl server spuštěn před klientem, jinak nebude možné přihlášení a uživatel bude muset čekat, dokud server nebude plně nastartován.

Dalším úkolem bude oprava zamrznutého úvodního dialogu se stavem načítání serveru. Při startu serveru zamrzne úvodní obrazovka (obrázek č. 7) a je zobrazena po celou dobu načítání (cca 30s), což působí, jako by se zasekla celá aplikace. V aplikaci je připraven dialog s aktivním prvkem (rotující progressbar), který informuje o načítaní, avšak ten se nezobrazí.

Obr. 7: Úvodní obrazovka při startu serveru

Kvůli zamrznuté úvodní obrazovce může nastat situace, že se uživatel pokusí o nové spuštění aplikace, protože původně spuštěná instance nereaguje. Nové spuštění ovšem selže, protože původní aplikace ve skutečnosti stále pracuje a nabíhající sever Tomcat [14] již blokuje port, ke kterému se nová instance snaží neúspěšně připojit. Pro zamezení opakovaného spuštění bude potřeba najít mechanizmus, který bude informovat o již spuštěné aplikaci a nedovolí vytvoření další instance.

6.2 Implementace

6.2.1 Zamrznutá úvodní obrazovka

Ve třídě MainFrame v serverovém modulu aplikace dochází ke startu aplikace pomocí metody *startServer(),* která má na starosti spuštění serveru Tomcat a překreslování GUI serveru.

Protože metoda spouští dlouhotrvající proces (spuštění serveru Tomcat) a mezitím dochází k překreslování obrazovky se stavem načítání, vytvořil jsem samostatné vlákno pro tuto akci, aby nedocházelo k blokaci dalších procesů. Dále jsem upravil tělo metody *checkServerState(),* která také pracuje s GUI aplikace tak, že jsem využil metodu *SwingUtilities.invokeLater (Runnable)* [15], která asynchronně uloží akci do fronty událostí pro změnu zobrazení (ukázka kódu č. 17).

```
1.  public void startServer(){
2.      this.stateLabel.setIcon(LOADING_ICON);
3.      this.stateLabel.setText(server_loading);
4.      LOADING_ICON.setImageObserver(this.stateLabel);
5.      this.stateLabel.repaint();
6.      this.repaint();
7.      Thread init = new Thread(){
8.              @Override
9.              public void run(){
10.                 controller.initialiseServer();
11.                 checkServerState();
12.             }
13.     };
14.     init.start();
15.   }

16. public void checkServerState(){
17. SwingUtilities.invokeLater(new Runnable(){
18.         @Override
19.         public void run() {
20.             if(controller.isServerRunning()){
21.           stateLabel.setText(server_running);
22.           stateLabel.setIcon(RUNNING_ICON);
23.           startServerButton.setEnabled(false);
24.         }else if(controller.isErrorFlagSet()){
25.           stateLabel.setText((server_error));
26.           stateLabel.setIcon(ERROR_ICON);
27.         }else{
28.           stateLabel.setText(server_not_running);
29.           stateLabel.setIcon(null);
30.         }
31.         repaint();
32.         }
33.     });
34.   }
```

Kód 17: Upravené metody, pro zobrazení dialogu o načítání

Nyní po startu serveru okamžitě naběhne obrazovka informující o startování serveru. Uživateli je zobrazen text „Spuštění...“ a rotující progressbar, který dává jasné znamení, že aplikace pracuje (obrázek č. 8).

Obr. 8: Obrazovka při startu serveru

6.2.2 Spuštění aplikace pomocí jednoho souboru

Z předchozího bodu víme, že spuštění Tomcatu dochází v serverovém modulu (CashBobServer) a je tedy nutné umístit spuštění klienta právě sem, těsně po spuštění Tomcatu. Pokud bychom tak neučinili, uživateli by se zobrazila obrazovka pro přihlášení ještě předtím, než by server přijímal HTTP požadavky, a proto by se uživatel nemohl přihlásit a musel by čekat, až server bude plně připraven.

Samotné nastavení nebylo obtížné. Do souboru *pom.xml* (ukázka č. 18) jsem definoval závislost na modul CashBob – (klientská část aplikace).

```
1.      <dependency>
2.        <groupId>cz.cvut.fel.restauracefel</groupId>
3.        <artifactId>CashBob</artifactId>
4.        <version>1.0</version>
5.      </dependency>
```

Kód 18: Definování závislosti na klientskou část

Po definování této závislosti nic nebrání přístupu ke klientským třídám a k následnému spuštění klienta. Do připravené metody z předešlého bodu - *startServer()* jsem přidal spuštění klienta opět pomocí *SwingUtilities.invokeLater* (kód č. 19).

```
1.  public void startServer(){
2.      this.stateLabel.setIcon(LOADING_ICON);
3.      this.stateLabel.setText(server_loading);
4.      LOADING_ICON.setImageObserver(this.stateLabel);
5.      this.stateLabel.repaint();
6.      this.repaint();
7.      //this.showMessageDialog("startupWarning", "InformationMessageTitle");
8.      Thread init = new Thread(){
9.          @Override
10.         public void run(){
11.             controller.initialiseServer();
12.             checkServerState();
13.             SwingUtilities.invokeLater(new Runnable(){
14.                 @Override
15.                 public void run() {
16.                     cz.cvut.fel.restauracefel.restauracefel.Controller.Controller.getInstance().run();
17.                 // Pokladna module starts after tomcat init
18.                 }
19.             });
20.         }
21.     };
22.     init.start();
23. }
```

Kód 19: Spuštění klienta v serverovém modulu

Špatně umístěné resource soubory

Při pokusu o spuštění aplikace se však objevila chyba kvůli špatně umístěným externím souborům aplikace. Zejména v modulu pokladna se používaly absolutní adresy pro ikony k tlačítkům a ve chvíli spuštění z jiného kontextu nebyly soubory nalezeny. Ukázku špatně používaných cest můžeme vidět v ukázce kódu č. 20. Tento modul počítal, že vedle soubrou *.jar* existuje složka *images* s obrázky.

```
1. public JButton addTypeWorkShift = new JButton(new ImageIcon("images/add-type-
   workshift.png"));
2. public JButton addTemplate = new JButton(new ImageIcon("images/add-template.png"));
3. public JButton listOfTemplates = new JButton(new ImageIcon("images/seznam-
   sablon.png"));
4. public JButton planOfShifts = new JButton(new ImageIcon("images/plan-of-
   workshifts.png"));
5. public JButton statistics = new JButton(new ImageIcon("images/statistics.png"));
6. public JButton information = new JButton(new ImageIcon("images/information.png"));
7. public JButton overviewShift = new JButton(new ImageIcon("images/edit-leader.png"));
8. public JButton overviewEmpShift = new JButton(new ImageIcon("images/edit-
   employees.png"));
9. public JButton close = new JButton(new ImageIcon("images/exit.png"));
```

Kód 20 Ukázka špatné adresace externích souborů

Veškeré obrázky a soubory se proto musí stát součástí spustitelného souboru a musí být umístěny v resource balíčkách (resource balíčky jsou složky, které jsou nastaveny v classpath aplikace a které jsou zahrnuty při sestavování aplikace). Výhodou tohoto umístění je, že po sestavení aplikace jsou tyto resource složky součástí jednoho

spustitelného souboru, a proto nemusíme nadále spolu s tímto souborem distribuovat další složku s obrázky.Prošel jsem tedy celou aplikaci a upravil jsem načítání obrázků a ikon tak, aby se načítaly pomocí metody *getResource(name)* [16], která hledá daný soubor v resource složkách pro danou třídu.Po přesunutí všech externích souborů do správných složek nebránilo nic spuštění aplikace.

6.2.3 Omezení počtu spuštěných instancí aplikace

Pro omezení počtu spuštěných aplikací jsem zvolil kontrolu pomocí zamykacího souboru, který bude informovat o spuštěné instanci. Při startu nové instance vznikne soubor, kterému se bude v pravidelném intervalu měnit čas modifikace. Díky poslední známé aktivitě aplikace nejdříve zkrontroluje dobu, před kterou byla poslední spuštěná instance aktivní. V případě, kdy rozdíl aktuálního času a času poslední modifikace je menší než určená hranice, může aplikace předpokládat, že již nějaká instance běží a tedy novou instanci nespustí. V opačném případě, bude-li rozdíl času vyšší než určený limit, bude se k situaci přistupovat, jako by žádná aplikace neběžela a dojde ke spuštění.

Další důležitou vlastností zamykacího souboru je, aby při pádu aplikace tento soubor přestal existovat. Kdyby soubor nadále existoval, uživateli by se nepovedlo spustit aplikaci do té doby, než uplyne čas potřebný k tomu, aby byl při novém spuštění aplikace rozdíl aktuálního času a času poslední aktivity vetší než zvolený interval.

Vytvořil jsem třídu s názvem *InstanceHook* (ukázka č. 22), která se bude starat o zamykací soubor. Spuštění samotné instance se bude řídit následujícím scénářem:

1. Kontrola, zda může dojít ke spuštění aplikace
2. Vytvoření zamykacího souboru
3. Spuštění aplikace
4. Spuštění časovače pro změnu času modifikace

Dále je tedy nutné, aby při uzavření aplikace došlo ke smazání souboru. K tomu jsem využil metodu *addShutdownHook(Thread)* ze třídy *Runtime* [17], která registruje nový virtualní stroj, který je spuštěn v okamžiku vypnutí aplikace a slouží k provedení akcí před vypnutím aplikace. Vypnutí může být vyvoláno metodou System.exit(), nebo ukončením aplikace operačním systémem (např. ctrl + c). Výslednou podobu celého spuštění aplikace můžeme vidět v ukázce č. 21.

```
1.  public static void main(String[] args) throws IllegalAccessException {
2.          final InstanceHook starter = new InstanceHook();
3.          if(starter.canStart())
4.          {
5.              starter.start();
6.              Runtime.getRuntime().addShutdownHook(new Thread() {
7.                  public void run() { starter.finish();}
8.              });
9.              Controller controller = Controller.getInstance();
10.             controller.run();
11.             starter.getTimer().start();
12.         }
13.     }
```

Kód 21: Výsledná metoda Main, pro spuštění aplikace

```
1.  public class InstanceHook {
2.      private final String FILENAME = "config/starter.ini";
3.      private final Long INTERVAL = 60000L;
4.      private final Integer REFRESH_INTERVAL = 3000;
5.
6.      public void InstanceHook() {}
7.
8.      /*** Delets locked file*/
9.      public void finish() {
10.         File f = new File(FILENAME);
11.         if(f.exists()){
12.             f.delete();
13.         }
14.     }
15.     /** Checks if the file exists and refreshing interval is not over the limit.  @return */
16.     public boolean canStart() {
17.         File init = new File(FILENAME);
18.         Date now = new Date();
19.         if (init.exists()) {
20.             Long lastModifed = init.lastModified();
21.             if (lastModifed < (now.getTime() - INTERVAL)) {
22.                 return true;
23.             } else {
24.                 return false;
25.             }
26.         } else {
27.             return true;
28.         }
29.     }
30.     /* Creates file or refreshes modified date.  @throws IllegalAccessException  */
31.     public void start() throws IllegalAccessException {
32.         if(!canStart()){
33.             throw new IllegalAccessException("Application already running.");
34.         }
35.
36.         File f = new File(FILENAME);
37.         if (f.exists()) {
38.             f.setLastModified(new Date().getTime());
39.             return;
40.         }
41.         try {
42.             f.createNewFile();
43.         } catch (IOException e) {
44.             e.printStackTrace();
45.         }
46.     }
47.     /*  Refreshes modified date. @throws FileNotFoundException*/
48.     public void update() throws FileNotFoundException {
49.         File init = new File(FILENAME);
50.         if (!init.exists()) {
51.             throw new FileNotFoundException();
52.         }
53.         init.setLastModified(new Date().getTime());
54.     }
55.     /* Refreshes modified date.  @return */
56.     public Timer getTimer() {
57.         Timer t = new Timer(REFRESH_INTERVAL, new ActionListener() {
58.             @Override
59.             public void actionPerformed(ActionEvent e) {
60.                 try {
61.                     update();
62.                 } catch (FileNotFoundException e1) {
63.                     e1.printStackTrace();
64.                 }
65.             }
66.         }) {
67.         };
68.         return t;
69.     }
70. }
```

Kód 22: Třída pracující se zamykacím souborem

6.3 Testování

Spuštění aplikace bylo vyzkoušeno na zařízeních se systémem Windows (XP, Windows 7 a Windows 8.1), OSX 10.10.2 a Ubuntu 14.04. Při každém spuštění se objevila úvodní obrazovka s progressbarem a po startu serveru byla spuštěna klientská část.

Ukázku spuštění lze také vidět ve videu na kanále YouTube [20], kde je ukázáno, jak nainstalovat a spustit aplikaci.

Dále jsem provedl statické testování pomocí programu PMD [18]. Kompletní výsledky testů jsou přiložené v elektronické příloze. Soubory *static-test-pokladna.html a static-test-cashbob.html* obsahují kompletní seznam nalezených chyb v modulech cashbob a pokladna. Cílem této bakalářské práce nebylo odstranit veškeré chyby, které se v projektu vyskytovaly ještě před mým zapojením do projektu, ale mým úkolem bylo ověřit, zda neexistují závažné chyby, které by bránily dalšímu používání aplikace. Dále bylo nutné zkontrolovat, zda nevznikly mým zásahem nové, závažné chyby. Ve výsledcích se objevuje několik chyb, které se týkají nově přidaného kódu, ale tyto chyby však nevyhodnocuji jako závažné. Například jedna z nalezených chyb poukazuje na skutečnost, že by vlastnosti třídy měly být umístěny hned za deklarací třídy, avšak při tvorbě GUI samo vývojové prostředí – Netbeans [19] umisťuje proměnné na konec třídy. Dalším příkladem chyby je, že by se programátor měl vyhnout jednopísmenému pojmenování proměnných, ale stejně jako v předešlém případě některé bloky kódu, jako např. bloky try-catch, jsou generovány standardně vývojovým prostředím a proměnné jsou jednopísmené. Výše uvedené chyby ukazují závažnost nalezených chyb.

6.4 Shrnutí

Je připravena uživatelsky přívětivější varianta spuštění aplikace pomocí jednoho souboru. Aplikace při startu informuje uživatele o svém stavu a uživatel tedy vidí, že aplikace pracuje. Také se dále podařilo zamezit několikanásobnému spuštění.

7 Videonávody

Cílem této kapitoly je natočit krátká videa, která pomůžou uživateli se seznámením s aplikací. Tato videa mají ukázat nejdůležitější funkcionalitu aplikace, tedy používání pokladního modulu. Natočená videa budou mít dvě funkce. První funkcí je samozřejmně návod použití. Uživatel si zvolí video, které ho zajímá a zároveň se může přesunout na požadovanou stopáž. Tímto způsobem se rychle dostane k požadované informaci, oproti psaného manuálu, u kterého by musel pročítáním najít hledanou informaci. Druhou funkcí je reklama produktu. Uživatel si před stáhnutím a zakoupením produktu podívá, jak aplikace vypadá a co umožňuje.

7.1 Analýza

Videa bude potřeba natočit, sestříhat a umístit na veřejně dostupné místo, aby byly přistupné budoucím uživatelům. Jako scénaře videí můžou být použity upravené akceptační testy pro hlavní funkcionalitu.

7.2 Řešení

Na serveru YouTube [20] jsem založil kánál – CashBob návody, který obsahuje 7 nahrávek o průměrné délce 1,5 minuty. Tato videa uživatele naučí základnímu používání pokladního modulu:

1. Instalace - video s návodem, jak stáhnout, rozbalit a spustit aplikaci. Video obsahuje také návod, jak nainstalovat javu.
2. Vytvoření účtů – video popisuje základní práci s účty. Vytvoření klasického účtu, rychloúčtu a účtu s přiřazeným stolem.
3. Objednávky – video uživatele naučí, jak snadno a rychle přidat novou položku na účet.
4. Placení účtu – v tomto videu se uživatel naučí, jak zaplatit celý účet nebo pouze vybrané položky.
5. Přesun položek mezi účty – video ukazuje důležitou funkcionalitu pro přesun položek mezi jednotlivými účty. Uživatel se naučí, jak přesunout položky na již existující účet nebo na prozatím neexistující.
6. Rozpracované objednávky – v tomto videu se uživatel naučí používat paměť pro rozpracované objednávky.
7. Editace menu – Toto video ukáže, jak si uživatel může přizpůsobit nabídku položek, od vytvoření nové kategorie, přiřazení položky až po změnu barvy jednotlivého políčka a změny názvu.

7.3 Shrnutí

Na serveru YouTube [21] jsou připravené videonávody, které usnadní uživatelům práci s aplikací.. Do budoucna doporučuji, aby po přidání důležité funkcionality vzniklo další video ukazující práci s aplikací.

8 Webové stránky

Nedílnou součástí nabízeného produktu je webová prezentace, která seznámí budoucí uživatele s produktem. Webová stránka je místo, kde uživatel nalezne všechny potřebné infomace, počínaje samotnými informacemi o produktu, ukázkami aplikace, kontakty a v neposlední řadě informacemi, kde a jak může uživatel aplikaci získat. Proto je jedním z cílů této práce vytvořit webové stránky.

8.1 Analýza webových stránek

Na stránkách by měl uživatel najít ty nejdůležitější informace, jako jsou informace o samotném produktu, kontakty a odkazy na další místa, kde lze nalézt další informace.

Protože produkt není ještě k dispozi, instalační balíček nebude prozatím umístěn na stránkách, ale bude pouze uvedena informace o blížícím se datu zvěřejnění (léto 2015). Ze stejného důvodu nebude na stránkách registrační formulář, přes který se bude provádět registrace produktu po uplynutí 30 dnů.

Výsledná stránka tedy bude obsahovat informace o nejdůležitejších vlastnostech produktu (pokladní modul, dotykové ovládání, editace menu a přesun položek mezi účty) doplněné o ilustrační obrázky. Dále bude obsahovat informace o získání aplikce a další důležitou sekcí budou informace o připravených návodech, a to videoukázky na YoutTube a stručný manuál. Poslední a nejdůležitější částí budou kontakty. Uživatelé se dozví, že nás mohou sledovat na twitteru, stát se fanouškem na Faceboooku nebo sledovat již výše zmíněný YouTube kanál.

Součástí původní webové prezentace je PHP skript pro vygenerování licenčního kódu. Tento skript zůstane a využijiho při zpracování registračního formuláře.

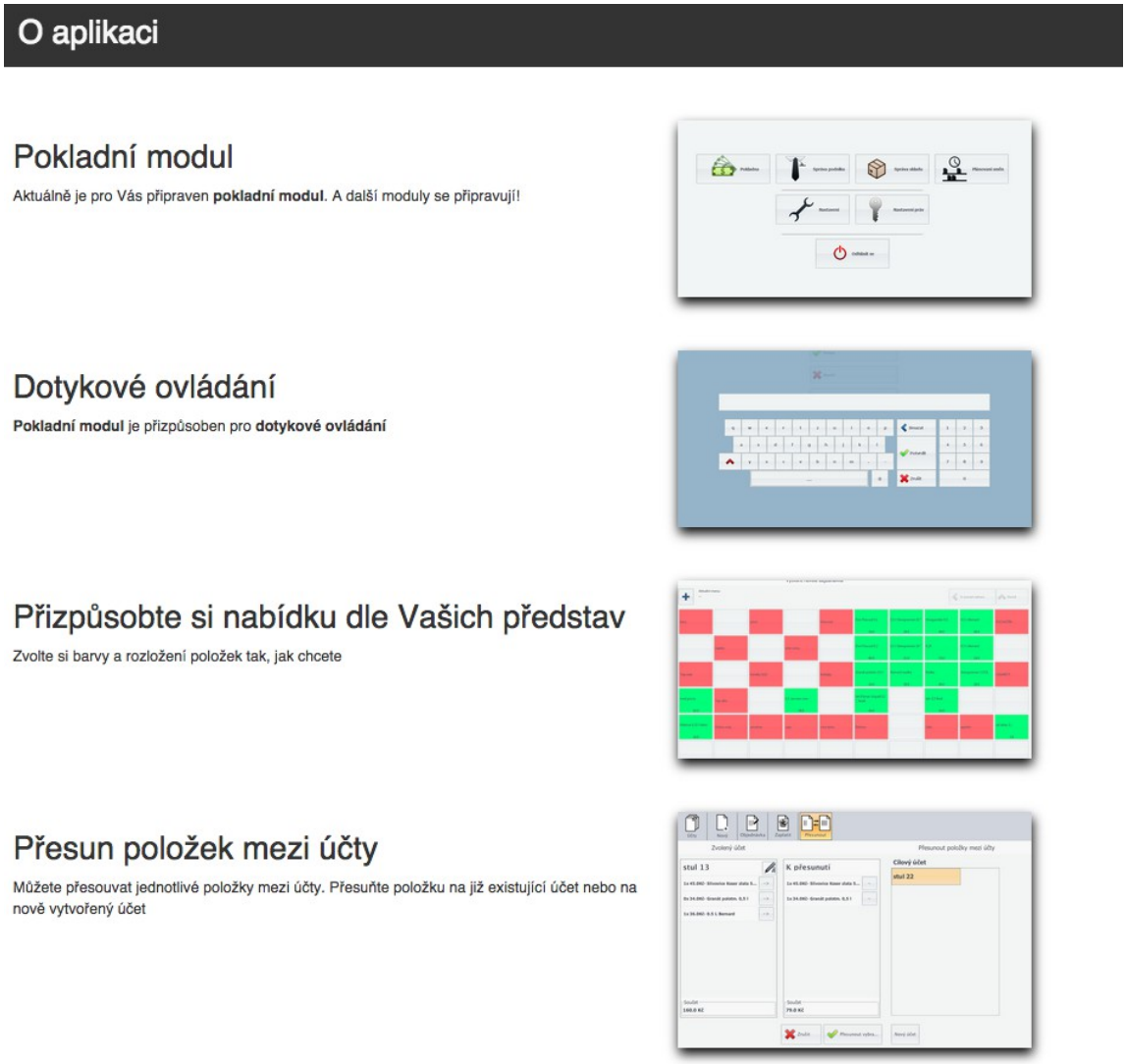
8.2 Implementace webových stránek

Pro implementaci webových stránek jsem použil následující technologii:

- HTML, HTML5 [22]
- PHP [23]
- jQuery – javascriptová knihovna [24]
- Twitter Bootstrap 3.5 – HTML, CSS framework [25]

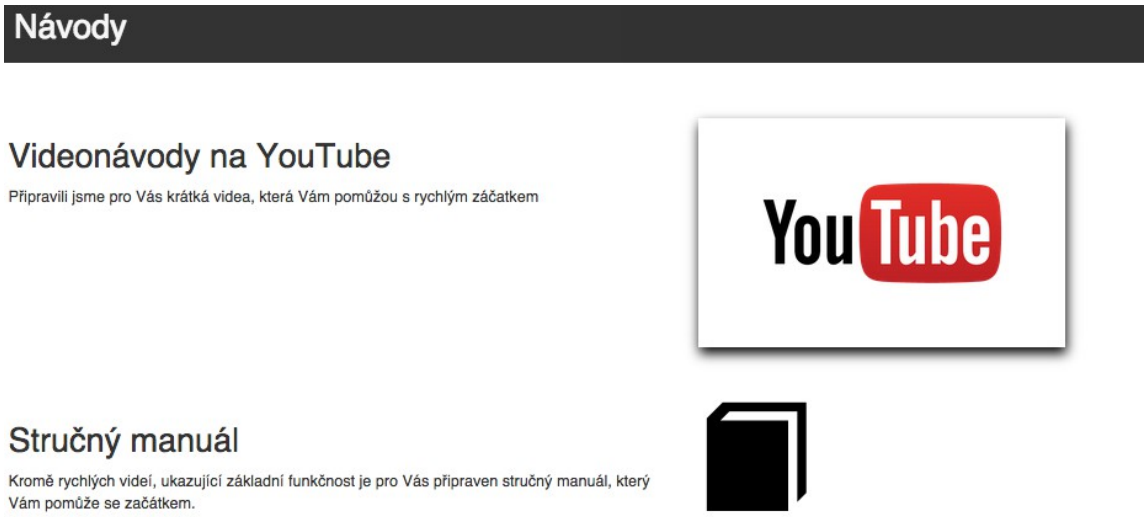
Stránky byly rozšířeny novou funkcionalitou a to možností online registrace. Toto rozšíření jsem popisoval již v kapitole č. 5 - Online registrace, a proto nebudu znovu popisovat detail implementace.

Výsledný vzhled jednotlivých částí webových stránek lze vidět na následujících obrázcích. Stránky jsem nasadil na zadavatelem pronajatý webhosting a jsou přístupné na adrese http://www.cashbob.cz.



Obr. 9: Informace o produktu na webových stránkách

Obr. 10: Infomace o stáhnutí aplikace na webových stránkách



Obr. 11: Odkazy na návody

Obr. 12: Odkazy na sociální sítě

Vytvořil jsem účet Google Analytics – webová analýza [26], abychom získali přehled o návštěvnosti a chování uživatelů. Tyto informace se mohou později využít pro případné změny před uvedením produktu na trh. Pro měření dat stačí umístit do stránky vygenerovaný javascriptový kód (ukázka č. 23).

```
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
      (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
      m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
  })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

  ga('create', 'UA-XXXXXX', 'auto');
  ga('send', 'pageview');
```

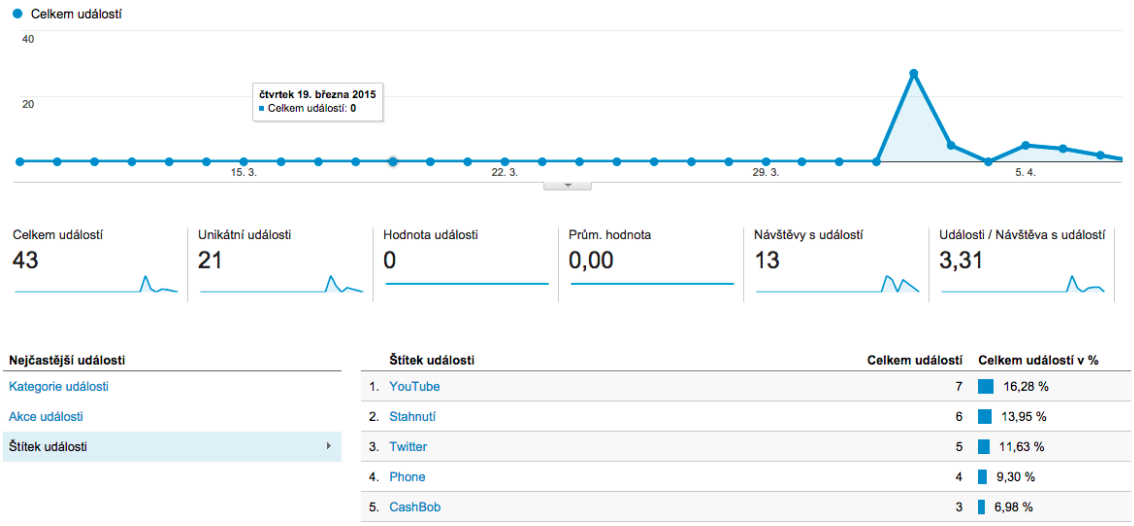
Kód 23: Měřící kód Google Analytics

Pro hlubší analýzu jsem zajistil měření událostí. Tyto informace poslouží například k přehledu, kolik lidí přešlo na YouTube kanál, Twitter či si prohlídli manuál aplikace. Toto měření zajistíme opět javascriptovou událostí, definovanou dle dokumentace [27] – názvem kategorie a vyvolané události, štítkem a hodnotou. Všechny mnou definované události vznikají při kliknutí na element. Příklad použití události je uvedený v ukázce č. 24. Tato událost odesílá na server informaci o kliknutí na položku „Návody“ v hlavním menu.

```
<a href="#tutorial" onclick="ga('Menu','click','Návody');">Návody</a>
```

Kód 24: Událost při kliknutí na odkaz

Obr. 13: Panel s aktuálními informacemi



Obr. 14: Přehled vyvolaných událostí uživateli na stránce

8.3 Shrnutí

Byla vytvořena a schválena nová varianta webových stránek. Dále byl na webu umístěn měřící skript pro sledování návštěvnosti. Ukázky aplikace Google Analytics můžeme vidět na obrázcích č. 13 a 14.

9 Vytvoření distribuční databáze

Dalším cílem je vytvoření distribuční databáze se základním množstvím ukázkových dat vytvořených dle zvolené tématiky. Ukázková data slouží k simulaci reálného prostředí, ve kterém se aplikace používá. Uživateli se tak po instalaci aplikace naskytne možnost přímo ji vyzkoušet, aniž by musel sám vytvářet testovací data.

Vytvořením jednoduché struktury ukázkových dat zajístíme, aby uživatel nebyl zmatený z velkého množství položek při prvním spuštění. Uživateli se vytvoří ukázková databáze s několika položkami a bude pro něj snadnější orientovat se v nové aplikaci.

9.1 Analýza

Se zadavatelem projektu jsme zvolili tématiku malé kavárny. Proto bude v nabídce předpřipraveno několik káv a koláčů. V projektu se používá technologie FlyWay [28] - databázový migrační nástroj pro vytvoření požadovaného stavu databáze. Dále aplikace pracuje s H2 databází [29]. Pro vytvoření ukázkové databáze bude nutné vytvořit SQL skript, který vymaže testovací data a vytvoří požadovaný stav.

9.2 Implementace

Protože aplikace Cashbob pracuje s H2 databází , využil jsem webovou konzoli (prohlížení tabulek, psaní SQL dotazů, ...) pro práci s databází. Vymazal jsem veškerá původní data a přes pokladní modul jsem vytvořil nově požadovaná data. Následně jsem použil příkaz *SCRIPT TO* '<název_souboru>', který vytvoří SQL dump [30] (textový soubor s SQL příkazy, který po spuštění vytvoří stav databáze ve chvíli vygenerování tohoto souboru). Z tohoto souboru jsem vybral požadované řádky a vytvořil jsem v pořadí 8 migrační skript v aplikaci s názvem *V8__production.sql* [31].

V okamžiku, kdy se aplikace spouští a neexistuje databáze, migrační nástroj FlyWay spustí jednotlivé skripty v pořadí *V1, V2, ...Vn* [32].

Při sestavování aplikace však neprošly testy, protože se při testech používají původní testovací data. Proto bylo nutné rozdělit migrační skripty pro testovací a produkční databázi. Vytvořil jsem složky (*test,migration*) a v konfiguračních souborech (*applicationContext.xml* a *testApplicationContext.xml*) jsem zadal cestu k migračním skriptům (ukázka č. 25).

```
<bean id="flyway" class="com.googlecode.flyway.core.Flyway" init-method="migrate">
        <property name="locations" value="classpath:db/migration/"/>
 </bean>
```

Kód 25: Přidání umístění skriptů v konfiguračním souboru applicationContext.xml

9.3 Shrnutí

Je vytvořená distribuční databáze, která uživateli vytvoří ukázková data pro malou kavárnu.

10 Reklama produktu

Důležitou součástí projektu je propagace produktu. Je nutné vytvořit prostor, kde budoucí uživatelé najdou potřebné informace a podporu v případě technických problémů či nejasností. V kapitole č. 8 jsem vytvořil a publikoval webové stránky, které mají odkaz na tři sociální sítě, které budeme používat pro propagaci – YouTube [20], Twitter [33] a Facebook [34].

10.1 Návrh

Prozatím se neplánuje použití finančních prostředků pro reklamu, dokud nebude aplikace mít několik stáhnutí a několik zpětných vazeb od uživatelů. Proto si prozatím vystačíme se sociálními sítěmi, které jsou zdarma. Prvním kanálem bude YoutTube, na kterém budou umístěny videotutoriály (kapitola č. 7). Jako druhé místo pro komunikaci bude sloužit Twitter účet , přes který budeme publikovat novinky a zprávy a uživatelé budou moci s námi řešit problémy, které je pálí. Posledním místem bude Facebook stránka, kde lze vkládat obrázky, videa a také přidávat komentáře. Všechny tyto kanály jsou zdarma a jsou masově používané lidmi.

10.2 Řešení

Webové stránky jsou k dispozici na adrese www.cashbob.cz a byly připraveny v rámci kapitoly č. 8. Kanál s videoturiály jsem popisoval v kapitole č. 7. Zbývá tedy vytvořit twitter účet a facebook stránku. Twitter účet byl založen bez problémů s uživatelským jménem *cashbob_soft* a je přístupný na adrese [35]. Facebook stránku nelze vytvořit samostatně pro nějaký produkt, jako profil na twitteru, ale je potřeba aby stránku založila osoba s existujícím účtem na facebooku. Prozatím jsem vytvořil stránku pod svým uživatelským profilem a později bude nutné přeřadit spravování stránky na osobu pověřenou zadavatelem projektu. Facebook stránka je k dispozici na adrese [36].

10.3 Shrnutí

Byly vytvořeny základní kanály na sociálních sití pro reklamu, komunikaci a podporu uživatelům. Aplikaci Cashbob najdeme na YouTube, Facebooku a Twitteru.

11 Licenční dialog

Při prvním spuštění aplikace by se měl uživatel seznámit s licenčním ujednáním [37] poskytovaného softwaru a potvrdit, že souhlasí s podmínkami užívání aplikace. Licenční ujednání je soupis pravidel a povinností, které umožňují aplikaci používat či distribuovat. Uživatel potvrzením licenčních podmínek dává najevo, že s danými podmínkami užívání souhlasí. Proto je dalším cílem této práce zobrazit uživateli při prvním spuštění aplikace licenční dialog.

11.1 Analýza

Pro návrh znění licence jsem se inspiroval licencí aplikace Seznam Instalátor [38] od společnosti Seznam.cz [39].
I přes nalezenou informaci v článku [40], který pojednává o právním významu zřeknutí se odpovědnosti a ve kterém je uvedeno: „*Je možné konstatovat, že ustanovení, jimiž se poskytovatel licence „zříká“ odpovědnosti za vady software a odpovědnosti za škodu, by měla být podle českého práva v každém jednotlivém případě neplatná pro rozpor se zákonem. Konkrétně se jedná o rozpor s kogentním ustanovením § 574 odst. 2 občanského zákoníku, jež stanoví, že "dohoda, kterou se někdo vzdává práv, jež mohou v budoucnosti teprve vzniknout, je neplatná"*“, jsem použil větu o zřeknutí se odpovědnosti, protože i tak velká fima, jako Seznam.cz používá tento obrat, uvedený v následující citaci: „*Seznam.cz rovněž nenese žádnou odpovědnost za škodu způsobenou použitím softwarového produktu sInstalator, ani za jakékoliv jiné škody nepřímo spojené s užitím tohoto softwarového produktu.*“ [39]

Licenci bude nutné zobrazit uživateli při prvním spuštění aplikace. Samotné zobrazení licence proběhne pomocí dialogu, který se po spuštění aplikace zobrazí. Pokud uživatel nebude se zněním licence souhlasit (zvolí tlačítko „Nesouhlasím“ nebo dialog zruší) nebude uživatel schopen aplikaci nadále používat a aplikace se ukončí.

Copyright (c) 2011, CACTOO SOFTWARE, Czech Republic
Všechna práva vyhrazena
Toto ujednání uzavíráte jako uživatel softwarového produktu Cashbob (dále jen „Uživatel“)
se společností Cactoo Software (http://cactoosoftware.com/) (dále jen "Výrobce").
Tento software je majetkem firmy Cactoo Software (http://cactoosoftware.com/).
S odsouhlasením této licence souhlasíte s následujícími body:

- *Software CashBob je poskytován bez záruky jakéhokoliv druhu,protože je poskytován ZDARMA.*
- *Za vzniklé škody užíváním této aplikace a za samotné chyby aplikace nenese výrobce žádnou odpovědnost a nelze tedy žádným způsobem vymáhat jakékoliv odškodnění.*
- *Aplikaci používáte na vlastní zopovědnost.*
- *Je zakázána dekompilace nebo jiný způsob získání zdrojových kódů aplikace.*
- *Je zakázána další distribuce nebo modifikace zdrojových kódů.*
- *Pro používání aplikace na jeden rok, je nutná registrace.*
- *Uživatel je povinen zálohovat důležitá data počítače.*

Pokud nesouhlasíte s touto licencí nemůžete aplikaci Cashbob nadále používat.

Text 1: Licenční ujednání aplikace CashBob

11.2 Implementace

Vytvořil jsem třídu *EulaForm*, která reprezentuje dialogové okno, které se zobrazí uprostřed monitoru v modálním zobrazení (dokud uživatel tento dialog nezavře nelze pracovat s jinými okny aplikace). Znění licence se načte ze souboru *LICENSE*, umístěného v kořenové složce aplikace. Součástí dialogu jsou tlačítka pro vyjádření souhlasu a nesouhlasu s licencí. Pokud dojde k odsouhlasení licence, do konfiguračního souboru se uloží informace o souhlasu a při dalším načtení nedojde opakovaně k zobrazení. V opačném případě se aplikace ukončí.

11.3 Testování

Součástí elektronických příloh je akceptační test, který testuje správné zobrazení licenčního dialogu.

11.4 Shrnutí

Ujednotilo se znění licence a byl implementován licenční dialog, který se zobrazí při prvním spuštění aplikace.

12 Manuál aplikace

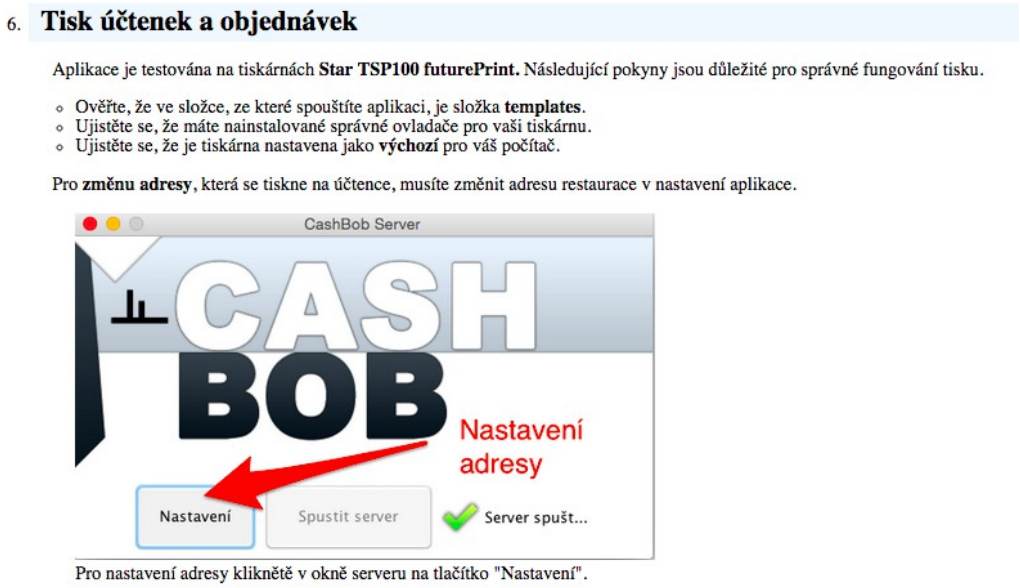
Dalším cílem této práce je vytvořit uživatelský manuál k aplikaci. V kapitole č. 7 - Videotoriály, jsem však vytvořil krátká videa ukazující nejdůležitější funkce pokladního modulu. Proto jsme se se zadavatelem projektu domluvili, že tento manuál bude pouze popisovat funkcionalitu, která nebyla zahrnuta ve videonávodech, aby nevznikala duplicitní práce.

12.1 Analýza

Do manuálu bude důležité zahrnout návod pro modul „Správa podniku“, který slouží pro editaci zaměstnanců, kategorií a položek menu. Dále se musí umístit do tohoto manuálu postup pro registraci, jak tisknout a jak změnit informace na účtence. Pro snadnou údržbu a editaci manuálu jsem zvolil html formát. Pro lepší srozumitelnost manuálu budou součástí textu názorné obrázky, které budou rozšiřovat text. Výsledný manuál bude součástí webových stránek.

12.2 Implementace

Manuál má formát jednoduché statické html stránky. Obrázky jsou vloženy do stránky ve formátu BASE64 (převod binárního souboru do prostého textu) [41] tak, aby byl celý manuál při stažení pouze jeden soubor. Ukázku části manuálu můžeme vidět na následujícím obrázku.



Obr. 15: Ukázka z manuálu, nastavení adresy

12.3 Shrnutí

Byl vytvořen manuál, který doplňuje videonávody na kanále YoutTube a je k dispozici na stránkách produktu. Uživatel v manuálu najde následující kapitoly: *Stáhnutí a instalace aplikace, spuštění aplikace, přihlášení, správa uživatelů, správa položek menu, registrace a tisk účtenek.*

13 Obrazovka se statistikami prodeje zboží a účtů

V samotném závěru práce jsme spolu se zadavatelem projektu identifikovali důležitou funkcionalitu, kterou aplikace postrádá. Uživatel nemůže žádným způsobem kontrolovat a procházet již uzavřené účty, a tedy nemůže provádět základní operace s účty, jako je zpětná kontrola položek účtů, nebo kontrola celkové utržené částky za prodané zboží. Úkolem je tedy implementovat obrazovku se správou účtů, na které uživatel najde informace o účtech a prodaných položkách.

13.1 Analýza

S vedoucím práce jsme se dohodli, že budeme postupovat formou prototypování, implementujeme základní funkcionalitu a postupně ji budeme rozširovat a upravovat. Sepsali jsme základní funkční požadavky a načrtli hrubý nákres vzhledu obrazovky. Pro počáteční verzi jsme identifikovali následující požadavky:

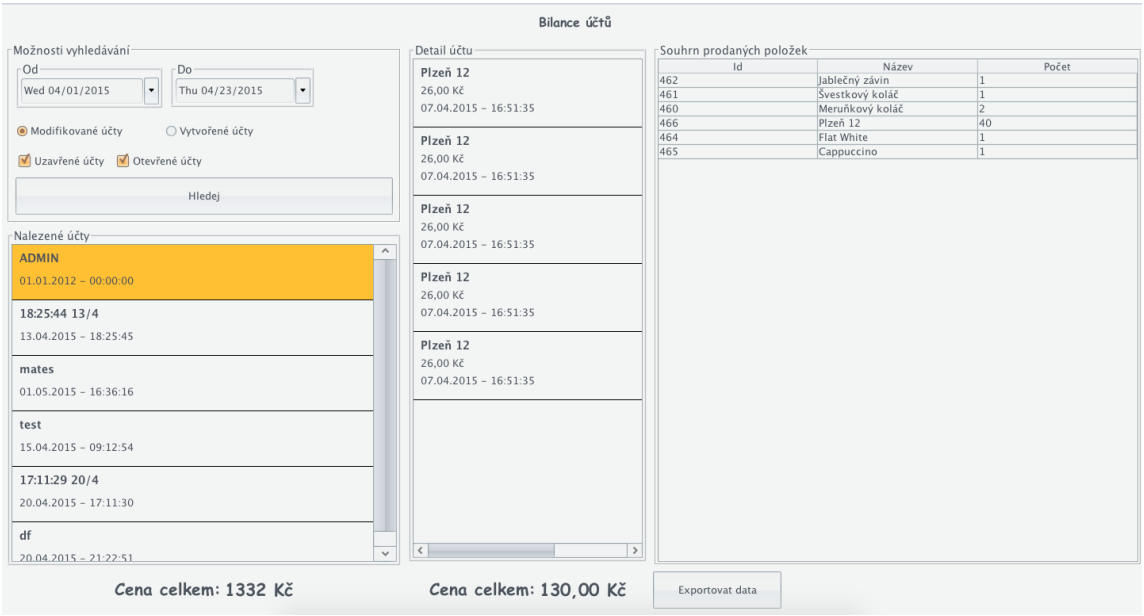
1. Hledání účtů pomocí časového intervalu:
   1.1. Podle datumu vytvoření účtu.
   1.2. Podle datumu poslední modifikace.
   1.3. Podle stavu účtu (otevřený, uzavřený).
2. Zobrazení detailu vybraného účtu.
3. Zobrazení údajů o všech prodaných položkách ve zvoleném časovém období.
4. Export všech předchozích infomací do tabulky (např. CSV, XLS).

13.2 Řešení

Vzhled obrazovky lze vidět na obrázku č. 16. V levé horní části se nachází panel s možnostmi pro vyhledávání. Uživatel zde nalezne dvě vstupní pole pro zadání datumu (od, do), dále upřesnění vyhledávacího dotazu pomocí přepínače a zaškrtávacích polí. Uživatel vybere, zda hledá účty podle jejich datumu vytvoření, nebo hledá účty, se kterými se naposled pracovalo v uvedeném časovém období. Posledním krokem je upřesnění dotazu, zda se hledají pouze otevřené účty, zavřené účty, nebo oba stavy zároveň.

Pod vyhledávacím panelem se nachází seznam nalezených účtů. Po vybrání požadovaného účtu se napravo od seznamu zobrazí detail účtu, kde obsluha nalezne veškeré objednané položky definované *názvem, cenou* a *časem objednání*. Pod seznamem účtů je zobrazena celková cena všech položek na všech účtech ve zvoleném období a pod detailem účtu se nachází celková cena všech položek na zvoleném účtu.

V levé části obrazovky nalezne uživatel souhrn prodaných položek za zvolené období. Tyto údaje lze použít pro zpětnou kontrolu množství prodaného zboží či například pro vyhodnocení, které zboží se nejvíce prodává. Pod touto tabulkou se nachází tlačíko pro export výsledných dat do tabulkového procesoru. Uživateli se otevře dialogové okno a po výběru umístění dojde k exportu do formátu *xls* (Formát Microsoft Excel).



Obr. 16: Obrazovka s bilancí účtů

Tvorba uživatelského prostředí nebyla nijak náročná. Finální verzi jsem vytvořil v editoru uživatelského prostředí. Pro získání dat bylo třeba vytvořit dotazy do databáze, pomocí kterých se získají požadovaná data. Dotaz pro získání účtů podle datumu vytvoření je velice jednoduchý – ve klauzuli *WHERE* se píší podmínky přímo na sloupce tabulky. Dotaz pro získání účtů, dle poslední modifikace, je nepatrně složitější. Zde se nejprve musí najít položky, které byly přidány v požadovaném čase a poté získat účty, pomocí reference do tabulky účtů, protože každá objednávka má informaci, ke kterému účtu patří (cizí klíč).

Pro vyplnění tabulky o přehledu prodaných položek se musí získat součet jednotlivých položek produktů. Aktuálně tyto data získávám průchodem přes již získané účty a položky, avšak tento princip není příliš vhodný, protože tím získávám asymptotickou složitost $O(n) = n^2$ (jeden cyklus uvnitř druhého). Pro použití alogiritmu s touto složitostí jsem se rozhodl z několika důvodů.

Protože byla tato kapitola nad rámec zadání této práce a nezbývalo mnoho času na její dokončení, upřednostnil jsem celkovou funkčnost před lepším řešením. Lepším řešením by byl databázový dotaz, který by vracel množinu uspořádaných trojic (*název produktu, počet, celková cena*), avšak toto vylepšení můžeme implementovat v dalších verzích programu, až budeme mít první zpětné vazby od uživatelů.

Dalším důvodem bylo špatně zobrazené datum napříč aplikací. S tímto problémem se do této doby nikdo nesetkal, protože se v aplikaci na žádném místě nepracovalo s datumem. Zobrazení datumu ve správném formátu je však na obrazovce s přehledem účtů klíčovým požadavkem pro správné používání přehledů, a proto jsem věnoval čas odstranění této chyby.

Pro odstranění této závady nakonec stačilo vytvořit adaptér transformující datum do požadovaného formátu. Problém spočíval ve špatném převodu datumu v okamžiku vytvoření objektu zaslaného serverem v podobě XML. Metoda, která měla tuto transformaci na starosti, vytvořila namísto instance klasického gregoriánského kalendáře

instanci julianského kalendáře, která způsobila nesprávné zobrazení datumu. Pro komunikaci mezi serverem a klientem se využívala technologie JAXB [42]. Řešení spočívá v přidání anotace (*@XmlJavaTypeAdapter(DateAdapter.class)*) k atributu reprezentujícího datum v třídě, která se transformuje do XML a zpět.

Posledním významným rozšířením je export dat do XLS souboru. Pro vytvoření souboru jsem využil knihovnu http://jexcelapi.sourceforge.net/. Export probíhá způsobem, kdy se do buněk tabulky pomocí souřadnic (i-tý řádek, j-tý sloupec) zapisuje požadovaná hodnota. Ukázku vytvoření jedné stránky - s přehledem účtů, vidíme v ukázce č. 26. Jedná se o jednoduchý iterační cyklus přes dané účty.

```
1.  private void fillAccountSheet(WritableSheet sheet, List<AccountDTO> accounts)
2.                      throws RowsExceededException, WriteException {
3.    String[] columns = new String[] { "id", "Název", "Vytvořeno","Poznámka", "Zaplaceno" };
4.    createTextCell(sheet,0,0,String.format("Souhrn - %s - %s", sdf.format(from),sdf.format(to)));
5.    for (int i = 0; i < columns.length; ++i) {
6.      createTextCell(sheet, i, 1, columns[i]);
7.    }
8.    for (int i = 0; i < accounts.size(); ++i) {
9.        createTextCell(sheet, 0, i + 2, accounts.get(i).getId().toString());
10.       createTextCell(sheet, 1, i + 2, accounts.get(i).getName());
11.       createTextCell(sheet, 2, i + 2, sdf.format(accounts.get(i).getCreatedate()));
12.       createTextCell(sheet, 3, i + 2, accounts.get(i).getNote());
13.       createTextCell(sheet, 4, i + 2, accounts.get(i).isOpened() ? "NE": "ANO");
14.   }
15. }
```

Kód 26: Tvorba stránky s přehledem vytvořených účtů

13.3 Testování

Jak jsem již uvedl v předchozí kapitole, na kompletní vyřešení tohoto úkolu včetně testů nezbyl čas. Prozatím jsou vytvořeny testy pro exportování XLS souboru. Třída XLSExportServiceTest testuje funkcionalitu vytvoření souboru. První test kontroluje, zda lze vytvořit požadovaný soubor a druhá metoda testuje, zda se korektně ukládají hodnoty na požadované pozice.

Doporučením pro další vývoj je dopsat testy na SQL dotazy, zda korektně vrací požadovaná data. Jedná se tedy o výčet účtů v požadovaném rozmezí časů a dotaz pro celkovou cenu všech položek na všech účtech.

13.4 Shrnutí

Rozšířením aplikace o přehled účtů a položek jsme poskytli uživateli silný nástroj pro zpětnou kontrolu. Uživatel může kontrolovat peníze a prodané položky. Díky exportu do souboru si může uživatel vytvořit grafy a pracovat s informacemi podle svého uvážení.

14 Závěr

Všechny zadané cíle této práce byly úspěšně splněny a podařilo a výsledkem je tedy funkční systém pro restaurace, který je připraven pro nasezení. Dále se podařilo práci rozšiřit o několik úkolů, které nebyly v původním zadání. V následujících odstavcích stručně srhnu veškerou práci na projektu.

Prvním důležitým úkolem bylo vyhodnotit a zpracovat výsledek testování použitelnosti. Podařilo se opravit všechny nalezené chyby a vylepšit funkčnost aplikace. Za zmínku stojí zejména paměť pro rozpracované objednávky a možnost vytvoření nového účtu přímo na obrazovce s přesunem položek mezi účty.

Další bodem této práce byla úprava kódu pro lepší použitelnost a budoucí rozšíření aplikace. Došlo ke změně používání systémové klávesnice, která usnadní přiřazení klávesnice ke vstupnímu prvku bez duplicitního kódu při každé registraci. Jednotlivé konfigurační soubory aplikace mají nyní jednoznačná jména. Pro validaci vstupních hodnot byla vytvořena komponenta, která uživateli oznamuje správné vyplnění hodnoty.

Také spouštění aplikace prošlo změnou. Aplikace se nyní spouští jedním souborem. Po startu serverové části dojde k automatickému spuštění klienta.

Uživateli je umožněno provést registraci produktu přímo v aplikaci. Nemusí přecházet na web produktu a tam vyplnit registrační údaje, ale aplikace uživateli nabídne formulář pro vyplnění údajů a následně uživateli na vyplněný email zašle licenční klíč, který vloží do registračního formuláře v aplikaci.

Na adrese www.cashbob.cz jsou umístěny propagační stránky produktu, kde uživatel najde nejdůležitější infomace o produktu. Dále byly vytvořeny profily na sociálních sítích Twitter, Facebook, které budou sloužit pro reklamu a podporu uživatelům.

Pro lepší začátek a seznámení s aplikací byly vytvořeny videonávody, které ukazují důležité vlastnosti aplikace - *Instalace, vytvoření účtu, vytvoření objednávky, placení účtu, přesun položek mezi účty, paměť pro rozpracované objednávky* a *editace menu.* Jako doplňující návod vznikl manuál aplikace, který provádí uživatele dalšími částmi aplikace a je k dispozici na stránkách produktu ve formě html stránek.

Nad rámec zadání přibyla do aplikace možnost tisku nových objednávek - obsluha restaurace si může vytisknout novou objednávku a odškrtávat si připravené věci. Dále byla vytvořena ukázková databáze, která při prvním spuštění vytvoří ukázková data malé kavárny. V neposlední řadě jsem implementoval zobrazení licenčního ujednání.

Nad rámec zadání jsem připravil obrazovku s přehledem prodaných položek. Uživatel bude mít možnost hledat účty dle datumu vytvoření nebo poslední změny, uvidí celkové součty cen všech účtů i jednotlivé detaily. Také zde nalezne statistiku prodávaných položek za zvolené období. Všechny tyto hodnoty lze vyexportovat do souboru.

15 Obsah přiloženého CD

CD /:

- **src/**.....Adresář s textem bakalářské práce
  - *bp.pdf*.....Text bakalářské práce ve formátu PDF
  - *bp.odt*.....Text bakalářské práce ve formátu ODT
- **tests/**.....Adresář s testy
  - *akceptacni-testy.pdf*.....Akceptační testy ve formátu PDF
  - s*tatic-test-pokladna.html*.....Výsledky statického testování č. 1
  - *static-test-cashbob.html*.....Výsledky statického testování č. 2

[1] Cashbob, pokladní modul, www.cashbob.cz [online]
[2] Jak psát hezký kód. www.zdrojak.cz, 13.4.2010. http://www.zdrojak.cz/clanky/jak-psat-hezky-kod-i/ [Online, cit. 13.05.2015.].
[3] Procházka Filip. Čistý projekt a spokojený zákazník. filip-prochazka.com., https://filip-prochazka.com/blog/cisty-projekt-a-spokojeny-zakaznik [Online, cit. 13.05.2015].
[4] Delegation pattern. http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/delegation.html, [Online, cit. 15.04.2015]
[5] Konfigurační soubor: Wikipedia, 17.02.2014, http://cs.wikipedia.org/wiki/Konfigura%C4%8Dn%C3%AD_soubor, [Online, cit. 10.01.2015]
[6] Abstract Methods and Classes. https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html, [Online, cit. 02.02.2015]
[7] Usability Lab at CTU in Prague http://ulab.cz/, [Online, cit. 15.02.2015]
[8] Heuristická analýza. http://human-computer-interaction.webnode.cz/testovani-a-hodnoceni-rozhrani/metody-testovani/heuristicka-analyza/, [Online, cit. 15.04.2015]
[9] Jaspersoft Business Intelligence, http://www.jaspersoft.com/ [Online, cit.05.03.2015]
[10] Ireport Dessigner. https://community.jaspersoft.com/project/ireport-designer, [Online, cit. 05.03.2015]
[11] Change your default printer. http://windows.microsoft.com/en-us/windows/change-default-printer#1TC=windows-7, [Online, cit. 05.05.2015]
[12] HTTP/1.1: Method Definitions, http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html, [Online, cit. 21.03.2015]
[13] JSON Ttutorial, http://tomcat.apache.org/, [Online, cit. 22.03.2015]
[14] Apache Tomcat, [Online, cit. 14.03.2015]
[15] SwingUtilities (Java Platform SE 7). http://docs.oracle.com/javase/7/docs/api/javax/swing/SwingUtilities.html, [Online, cit. 15.02.2015]
[16] Class (Java Platform SE 7) http://docs.oracle.com/javase/7/docs/api/java/lang/Class.html#getResource(java.lang.String), [Online, cit. 15.02.2015]
[17] Runtime (Java Platform SE 7) http://docs.oracle.com/javase/7/docs/api/java/lang/Runtime.html#addShutdownHook(java.lang.Thread) [Online, cit. 15.02.2015]
[18] PMD. http://pmd.sourceforge.net/, [Online, cit. 21.02.2015]
[19] Welcome to NetBeans, https://netbeans.org/, [Online, cit. 03.01.2015]
[20] YouTube, https://www.youtube.com/
[21] CashBob návody, https://www.youtube.com/playlist?list=PL0bK_89680mtsXe4Fk_fODM-bowhOaeEA
[22] HTML Tutorial, http://www.w3schools.com/html/, [Online, cit. 28.01.2015]
[23] PHP: Hypertext Preprocessor, http://php.net/
[24] JQuery, https://jquery.com/
[25] Bootstrap · The world's most popular mobile-first and responsive front-end framework, http://getbootstrap.com/
[26] Google Analytics, http://www.google.com/analytics/
[27] Event Tracking - Web Tracking (ga.js) https://developers.google.com/analytics/devguides/collection/gajs/eventTrackerGuide, [Online, cit. 02.04.2015]
[28] FlyWay. http://flywaydb.org/, [Online, cit. 05.04.2015]
[29] H2 Database Engine. http://www.h2database.com/html/main.html [Online, cit. 05.04.2015]

[30] SQL Dump. http://www.postgresql.org/docs/9.1/static/backup-dump.html [Online, cit. 05.04.2015]
[31] Sql-based migrations. http://flywaydb.org/documentation/migration/sql.html [Online, cit. 05.04.2015]
[32] Migrate. http://flywaydb.org/documentation/command/migrate.html, [Online, cit. 05.04.2015]
[33] Twitter, https://www.twitter.com/
[34] Facebbok, https://www.facebook.com/
[35] Cashbob (@cashbob_soft), https://twitter.com/cashbob_soft
[36] CashBob, https://www.facebook.com/pages/CashBob/348550025349925
[37] Softwarové licence: úvod pro obyčejné lidi. http://www.zdrojak.cz/clanky/softwarove-licence-uvod-pro-obycejne-lidi/ [Online, cit. 07.04.2015]
[38] Instalátor – Seznam instalátor, http://software.seznam.cz/instalator, [Online, cit. 05.04.2015]
[39] Licenční ujednání k užití Seznam Instalátoru. http://software.seznam.cz/podminky-sluzby [Online, cit. 05.04.2015]
[40] Odpovědnost za vady software. http://www.e-advokacie.cz/cs/clanky/odpovednost-za-vady-software [Online, cit. 09.04.2015]
[41] Binary-to-text encoding. https://en.wikipedia.org/wiki/Binary-to-text_encoding [Online, cit. 11.04.2015]
[42] Java Architecture for XML Binding, http://www.oracle.com/technetwork/articles/javase/index-140168.html, [Online, cit. 15.05.2015]