VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

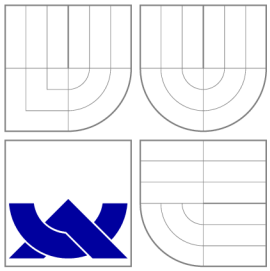
SYSTÉM REGISTRAČNÍ POKLADNY S PODPOROU DOTYKOVÝCH ZAŘÍZENÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

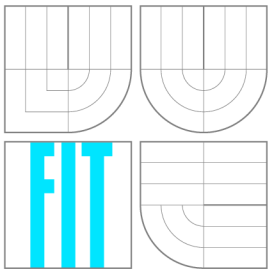
AUTOR PRÁCE
AUTHOR

MAREK SKOPAL

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM REGISTRAČNÍ POKLADNY S PODPOROU DOTYKOVÝCH ZAŘÍZENÍ

CASH DESK SYSTEM WITH A TACTILE DEVICE SUPPORT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MAREK SKOPAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2013

Abstrakt

Cílem této práce bylo navrhnout a implementovat systém registrační pokladny s použitím pro restaurační zařízení s optimalizací pro různá dotyková zařízení. Zadavatelem práce je společnost Draci Brno a vytvořený systém odpovídá jejich specifikaci. Při návrhu bylo cíleno na přehledné a efektivní rozhraní. Při implementaci rozhraní byly použity technologie HTML5 a CSS3 vytvářející responzivní web design. Aplikační část je vytvořena v PHP s Nette Frameworkem. Dále práce popisuje implementaci tiskového klienta v jazyce C#, který umožňuje tisk daňových dokladů pomocí termotiskárny.

Abstract

The goal of this work was to design and implement cash desk system for restaurants. This system should have various optimizations for touch devices. The submitter of this project was company Draci Brno and the designed system corresponds to their specifications, which mostly consist of requirements for clear and effective interface. The HTML5 and CSS3 technologies were used for implementation of interface, which led to creation of responsive web design. The application part of the system is created in PHP with Nette Framework. The next part of the work focuses on implementation of the print client in C#, which allows the print of tax receipts with use of thermal printer.

Klíčová slova

Pokladní systém, dotyková zařízení, HTML5, CSS3, responzivní web design, PHP, Nette Framework

Keywords

Cash Desk System, Touch devices, HTML5, CSS3, responsive web design, PHP, Nette Framework

Citace

Marek Skopal: Systém registrační pokladny s podporou dotykových zařízení, bakalářská práce, Brno, FIT VUT v Brně, 2013

Systém registrační pokladny s podporou dotykových zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Burgeta, Ph.D.

.....
Marek Skopal
14. května 2013

Poděkování

Děkuji vedoucímu práce, Ing. Radku Burgetovi Ph.D., za poskytnuté konzultace a odbornou pomoc při tvorbě práce. Dále bych rád poděkoval vedení společnosti Draci Brno za umožnění vzniku této práce.

© Marek Skopal, 2013.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **3**

2 Pokladní systémy **4**
2.1 Dotykové obrazovky 4
2.2 Termotiskárna 5

3 Návrh systému **6**
3.1 Analýza českého trhu 6
3.2 Návrh grafického rozhraní 6
3.2.1 Rozhraní pokladního systému 7
3.2.2 Rozhraní skladního systému 8
3.3 Návrh funkcí 10
3.3.1 Původní specifikace zadavatele 10
3.3.2 Uživatelské role 10

4 Implementace **12**
4.1 Použité technologie 12
4.1.1 PHP 12
4.1.2 Nette Framework 13
4.1.3 HTML5 13
4.1.4 CSS3 13
4.1.5 AJAX a JQUERY 13
4.2 Struktura aplikace 14
4.2.1 Model-View-Presenter 14
4.2.2 Konfigurační soubor 14
4.3 Implementace pokladny 15
4.3.1 Responzivní web design 15
4.3.2 Datové operace 16
4.3.3 Překreslování view 17
4.4 Implementace skladu 18
4.4.1 Stránkování 18
4.4.2 Validace formulářů 19
4.4.3 Zprávy a chybová hlášení 19
4.5 Autorizace uživatelů 20
4.6 Databáze 20
4.6.1 Optimalizace 21
4.7 Implementace tisku 22
4.7.1 Popis XML komunikace 22

4.7.2 Implementace tiskového klienta 24
4.8 Export / import 24

5 Testování a nasazení **26**
5.1 Vyjádření klienta 26

6 Závěr **28**

A Obsah DVD **30**

B ER diagram a Use Case diagram **31**

Kapitola 1

Úvod

S prodejem zboží nebo služeb se setkáváme snad v každém oboru lidské činnosti. Provozovatelé těchto prodejen potřebují o veškerých prodejích evidovat záznamy, aby mohli vést účetnictví a provádět inventury, k čemuž stále častěji využívají systémů registračních pokladen. Tyto se snaží většinu úkonů s tím spojených zjednodušit. Podobných systémů je implementována celá řada a tvorba dalšího by se nemusela jevit jako opodstatněná. V poslední době však dochází k velkému rozšíření mobilních dotykových zařízení a cílem této práce je navrhnout a implementovat systém registrační pokladny optimalizovaný pro použití na celé řadě těchto zařízení, který na trhu s pokladními systémy chybí.

Druhá kapitola definuje, co to pokladní systémy jsou a jaký je jejich historický vývoj. Dále popisuje technologie dotykových obrazovek, pro které je navržený systém optimalizován, a technologii termotiskáren, které slouží pro tisk účtenek a daňových dokladů pro zákazníka. Třetí kapitola se věnuje návrhu systému, kde je nejprve provedena analýza českého trhu, ze které jsou zjištěny typické funkce a definovány někdy až zásadní nedostatky nabízených pokladních systémů. Společně se specifikací externího zadavatele Draci Brno a na základě předchozího průzkumu je poté navrženo grafické rozhraní a funkce systému pokladny a skladového systému, se kterým pokladna spolupracuje. Následující kapitola se věnuje samotné implementaci jako webové aplikace postavené na nejnovějších technologiích na straně klienta HTML5 a CSS3, doplněných responzivním designem zajišťujícím podporu různých velikostí obrazovek. Na straně serveru je využito jazyka PHP a implementováno za pomocí Nette Frameworku. Dále je zde popsána implementace aplikace tiskového klienta, který slouží pro komunikaci s termotiskárnou pro platformu Microsoft Windows. Pátá kapitola se zabývá testováním vytvořeného systému a popisem zavedení u externího zadavatele. Závěrečná kapitola zhodnocuje dosažené výsledky a uvažuje nad dalším vývojem aplikace.

Kapitola 2

Pokladní systémy

Z historického hlediska systém pokladního terminálu [1] je klávesnicový terminál, spolupracující s mechanickou zásuvkou na fyzické uložení hotovosti a tiskárnou, která tiskne účtenky, často doplněnou displejem informujícím zákazníka o kupovaných položkách, respektive jejich cenách. Funkcí pokladního systému je automatizovat výpočet finální ceny kupovaných položek a výpočet jejich daně. Měl by se umět snadno přizpůsobit změnám cen a dani položek.

S těmito převážně hardwarovými pokladnami řízenými mikrokontrolerem se v poslední době setkáváme stále méně. Jsou nahrazovány řešeními postavenými na klasických PC architekturách a využívajícími běžné operační systémy. Navíc s rozšířením dotykových displejů, již mnohdy absentují použití hardwarové klávesnice a veškerý vstup se tak děje pomocí dotykového rozhraní. Z tohoto hlediska je zřejmé, že klíčové se tu stávají pokladní aplikace, které vytvářejí mezivrstvu mezi uživatelem a operačním systémem pokladny.

Pokladní systémy jsou tedy aplikace umožňující prodávajícímu personálu provádět činnosti v provozovně spojené s transakcí, ukládáním a předáváním peněžních prostředků za zboží či služby. Pokladní systém sjednocuje všechny tyto činnosti a vytváří evidenci pro hotovostní i bezhotovostní platby. Časté u pokladního systému bývá rozdělení uživatelských rolí, kdy prodávající personál má omezené možnosti prováděných činností. Většinou má k dispozici jen funkce spojené s prodejem zboží. Management má k dispozici další nástroje sloužící ke skladové evidenci a inventurám, nástroje umožňující kontrolovat personál, vydávat a přijímat faktury, zobrazovat různé statistiky a jiné funkce spojené s administračními úkony.

Pokladních systému je celá řada druhů, lišících se vzhledem ke specializaci na konkrétní obor činnosti, se kterým jsou spojené (např. obchod s potravinami, restaurace, hotely, rozvážkové služby aj.). V této práci se budu téměř bez výjimek zabývat návrhem a implementací systému s využitím pro restaurační zařízení.

2.1 Dotykové obrazovky

Dotyková obrazovka [11] je označení pro elektronický displej, který dokáže detekovat dotek a identifikovat jeho umístění na zobrazované ploše. Dotek je zpravidla prováděn prsty, nebo některým ze speciálních zařízení, jako je stylus. Některé tyto displeje jsou vybaveny vrstvou, která je nejen schopna registrovat tlak, ale může také pracovat v opačném směru a poskytovat dotykovou odezvu. Dotykové obrazovky se liší technologií použitou pro snímání doteku a v praxi jsou nejpoužívanější tyto:

Rezistivní Pracuje na principu dvou vrstev, kdy flexibilní vrstva, potažená transparentním oxidem kovu, je stlačována dotykem, a přichází do kontaktu s druhou vrstvou, která je pokryta vodiči. Tato technologie není přímo závislá na dotýkající se předmět, ale pro identifikaci doteku je třeba vyvinout určitý tlak. Rezistivní vrstva navíc zhoršuje čitelnost obrazu na displeji.

Infračervené záření Tvoří jej matice LED vytvářející infračervené záření zleva doprava a shora dolů, kde je snímáno detektorem záření. Při doteku se paprsek světla přeruší a dojde tak k identifikaci odpovídající souřadnici na obrazovce. Výhodou je, že dotyk může být proveden libovolným větším předmětem (stylus je nevhodný). Hlavní nevýhodou je, že diody jsou umístěné nepatrně nad vrstvou obrazovky, a k identifikaci doteku tak dochází ještě před samotným dotekem na plochu obrazovky. Může zde docházet k nechtěným identifikacím způsobených cizími tělesy, např. znečištění prachem.

Povrchová akustická vlna (SAW) Zde jsou přenášeny ultrazvukové akustické vlny po povrchu displeje. Při doteku prstem je část vlny absorbována, což umožňuje lokalizaci souřadnice. Tato technologie umožňuje ovládání v rukavicích, ale obrazovky jsou velmi náchylné na znečištění.

Kapacitní Displej je potažen vrstvou transparentních vodičů, které registrují dotek jako přenos elektrostatického náboje lidského těla, který je měřitelný jako změna kapacity. Tato technologie vyžaduje ovládání prsty, případně speciálním kapacitním stylusem. Umožňuje nejkvalitnější zobrazení a není snadno ovlivnitelná znečištěním ani náchylná na nevyžádané identifikace doteku.

2.2 Termotiskárna

Termální tiskárny [2] se od jiných typů tiskáren liší v použití tepla na inkoust, který je součástí papíru, na který tiskárna tiskne. Termální tiskárna se skládá z hlavy tiskárny, roličky termopapíru a posuvného mechanismu s gumovým válečkem. Některé tiskárny jsou doplněné řezacím mechanismem, usnadňujícím utržení papíru. Zdroj tepla v termotiskárně je elektricky zasílán do tiskové hlavy. Jakmile tisková hlava zahřeje požadovanou řádku textu, je papír posunut dopředu gumovým válečkem, kde se po doteku s hlavou zobrazí text na papíře. Tato operace je velmi rychlá a termální tiskárny dovedou tisknout rychlostí 6–12 palců za sekundu. Barva v pásce používaná v termální tiskárně není na bázi běžného inkoustu, ale na bázi vosku nebo pryskyřice. Vytištěný text tak nemá velkou životnost a asi dva roky po jeho vytisknutí text zcela mizí. Většina tiskáren tiskne pouze v černé barvě, případně v několika odstínech šedi, existují ale i varianty umožňující barevný tisk. Termotiskárny jsou používané především jako součást pokladních terminálů, kde nahradily pomalejší a hlučnější jehličkové tiskárny.

Kapitola 3

Návrh systému

Navrhovaný pokladní systém byl zadán jako zakázka společností Draci Brno (dále jen zadavatel). Návrh tedy probíhal na základě specifikace zadavatele a následných konzultací. Pro seznámení se s běžnými funkcemi pokladních systémů určených pro restaurační provoz jsem provedl analýzu programů nabízených na českém trhu. Na základě těchto poznatků jsem navrhl grafické rozhraní a funkce systému.

3.1 Analýza českého trhu

Na českém trhu se vyskytuje několik pokladních systémů sloužících k provozování restaurace a implementace dalšího by se tak nemusela jevit rozumně. Zadavatel práce Draci Brno však měl specifické požadavky na funkce, které konkurenční řešení nenabízejí. Navíc téměř ve všech případech trpí nabízené aplikace nejrůznějšími, z uživatelského i profesionálního pohledu zřejmými chybami. Nejzásadněji se jeví chyby v uživatelském rozhraní. Pokud systém obsahuje rozhraní použitelné pro dotykové ovládání (několik výrobců dotyková ovládání zcela opomíjí), mají většinou z ergonomického hlediska nevhodně umístěné prvky. Prvků je nadbytečné množství a mnohdy není zřejmá jejich funkce. Toto je dáno možností přílišného uživatelského nastavení obsahu prvků, kde si uživatel může volit jejich velikost, barvu, umístění a někdy do nich vkládat obrázky, nebo dělat jiné úpravy (viz. obrázek 3.1). V celkovém pohledu se tak i zasvěcený personál může v systému lehce ztratit, čímž jistě klesá produktivita práce. Všechny zkoumané nabízené systémy jsou nativní aplikace pro Microsoft Windows a neumožňují tak použití na přenosných zařízeních využívajících jiný než tento operační systém

3.2 Návrh grafického rozhraní

Po předcházející analýze a na základě specifikace zadavatele jsem se rozhodl systém rozdělit do dvou na sobě nezávislých rozhraní. V prvním pokladním rozhraní, které bude sloužit pro účely práce s pokladnou, budou zobrazeny položky, jejich kategorie a otevřené účty. Toto rozhraní bude optimalizováno pro ovládání na dotykovém zařízení. Druhé rozhraní pro funkce skladu bude obsahovat převážně prvky zobrazené v tabulkách se vkládáním a editací dat pomocí formulářů. Pro jeho spíše kancelářské použití by optimalizace pro dotykové zařízení nebyla vhodná.

Obrázek 3.1: Program QPOS a program Bar & Restaurant

3.2.1 Rozhraní pokladního systému

Z analýzy existujících aplikací vyplývá, že požadované pokladní rozhraní musí být maximálně jednoduché na použití i pro netechnicky založeného uživatele (nikoliv však na úkor efektivity často prováděných úkonů) a minimalistické. Z jeho prvků musí být na první pohled jasná jejich funkce a až na nějakou výjimku nebude možné jakkoliv měnit jejich tvar, barvu, nebo pozici. Design a velikost prvků grafického rozhraní jsou přizpůsobeny ovládání dotykem pomocí prstů na obrazovkách kapacitních dotykových displejů a infračervených dotykových displejů. Mezi těmito zařízeními je velký rozdíl ve velikosti (7″ tablet vs. 27″ monitor) a při návrhu pokladního rozhraní tak bylo třeba reflektovat velké rozdíly ve velikosti, rozlišení, i poměru stran obrazovek, kde některá zařízení se používají v režimu nastojato – portrait a některá naležato – landscape.

Kategorie a položky

Navržené rozhraní by se dalo rozdělit na dvě části. V levé části zabírající asi 70% šíře obrazovky jsou tlačítka s názvy kategorií, která se po stisku změní na tlačítka s názvy položek dané kategorie, doplněná tlačítkem pro návrat na výběr kategorií. Stiskem těchto tlačítek je provedeno přidání jednoho kusu dané položky do aktuálně otevřeného účtu. Ve spodní části těchto tlačítek je uvedena informace o ceně položky a počtu zbývajících kusů na skladě. Důležitou a jedinou vlastností tlačítek je jejich barva, která zřetelně oznamuje

stav (nikoliv přesný počet) položky na skladě: zelená – pokud je zboží dostatek, žlutá – pokud dochází, červená – pokud není skladem. Tyto informace jsou však pouze orientační a zboží lze prodávat, i když se podle systému skladem nenachází. Nemuselo totiž ještě dojít k jeho naskladnění do systému. Pod těmito tlačítky je pás rychlých kategorií, které přepnou z právě zvolené kategorie rovnou do kategorie jiné. Kategorie tlačítek nemají potomky v podobě jiných kategorií z důvodu přehlednosti a odhadovaného počtu prodávaného zboží udávaného zadavatelem. Počet kategorií je tak limitován na 35, počet položek na kategorii na 34. Systém samozřejmě umožňuje evidovat libovolný počet položek, ale více než stanovují tyto limity, nebude zobrazeno.

Účty

Ve druhé, pravé části navrženého rozhraní, zabírající zbylých 30% celkové šíře obrazovky jsou informace o otevřených účtech a operacích s nimi. Je-li otevřen nějaký účet, je v horní části zobrazeno tlačítko pro zobrazení seznamu otevřených účtů, které se po jeho aktivaci změní na tlačítko pro vytvoření nového účtu. Při vytvoření nového účtu je rovnou změněn pohled z výběru mezi účty do otevíraného účtu. Ve spodní části je vždy zobrazeno tlačítko pro vytištění účtu, které provede uzavření a tisk posledního otevřeného účtu. Mezi těmito tlačítky jsou informace o otevřeném účtu – jeho číselná identifikace, a pokud je přiřazen nějakému vlastníkovi zobrazí se jméno jeho vlastníka. Vlevo vedle této informace je tlačítko pro zrušení celého účtu a vpravo možnost přiřazení účtu novému nebo jinému vlastníkovi. Pod tímto se nalézají informace (název a cena) o jednotlivých položkách na účtech doplněných tlačítkem pro jejich smazaní. Pod nimi je celkový součet ceny pro daný účet. Celá tato část je při více položkách na účtu než umožnují možnosti zobrazujícího zařízení scrollovatelná.

Stav pokladny, příjem a výdej hotovosti

V průběhu návrhu byla zadavatelem přidána nutnost funkce evidence hotovosti v pokladně. Aby nedošlo ke změně současného rozložení, má pravá část obrazovky přepínatelný obsah. K přepnutí dochází po stisku tlačítka umístěného mezi pravou a levou částí. Kromě číselné representace aktuální hotovosti v pokladně a denní tržby jsou tu tlačítka pro výběr nebo vklad hotovosti. Při jejich aktivaci se zobrazí formulářový input pro vložení přijímané nebo vybírané sumy a číselná klávesnice pro pohodlné zadání na dotykovém zařízení.

Kreditní uživatelé

Při vyvolání změny vlastníka účtu je přes celou obrazovku vykreslen seznam evidovaných kreditních uživatelů. Jejich aktuální stav kreditů je doplněn jednoduchým formulářem pro přidání nového uživatele. Je-li účet nějakému z nich přiřazen, jsou ceny položek změněny na informaci o kreditech a suma na finální změnu kreditů po uzavření účtu. Seznam uživatelů je scrollovatelný. Samotné kredity jsou zobrazeny jako běžná položka v kategorii kredity.

3.2.2 Rozhraní skladního systému

Pro činnost pokladního systému je potřeba naplnit systém daty, zároveň je dobré mít o prodejích dobrý přehled. Rozhraní skladní části nebylo optimalizováno pro použití s dotykovými zařízeními, protože rozsáhlé tabulky jsou na malých obrazovkách špatně čitelné a vyplňování množství formulářů na virtuální klávesnici je značně nepohodlné. Předpokládá

se navíc, že tyto převážně manažerské operace bude provádět jiná osoba z kanceláře. Rozhraní skladního systému je tedy zcela jiné. Upřednostnil jsem uspořádání prvků typické pro webovou aplikaci před rozhraním běžným u desktopových aplikací. Na pravé straně obrazovky je umístěno menu se záložkami jednotlivých pohledů a zbylá část je věnována aktuálně zobrazenému pohledu. Navržené pohledy jsou následující:

Skladové zásoby Zobrazuje stav na skladě pro jednotlivé položky za určité časové období (po dnech) případně jen pro určitou kategorii zboží. Dává informaci o naskladněných a prodaných kusech a nákladech s naskladněním spojenými. Při výběru nějaké položky umožňuje zobrazit podrobný pohyb na skladě formou grafu. Položky, které jsou podpoložkou jiné, se tu nezobrazují a stav skladu je zobrazován jen pro položku jim nadřazenou.

Import dat Obsahuje formulář pro nahrání souboru z tabulkového procesoru k úpravě stavu skladu. Zároveň umožňuje vygenerovat šablonu tohoto souboru pro aktuální stav skladu.

Seznam účtenek Zobrazuje souhrnné informace o prodaných účtenkách a jejich vliv na stav kreditů a hotovosti v kase za určité období, případně pro konkrétního kreditního uživatele. Je zde tlačítko pro stornování účtenky a tlačítko pro její opětovné vytištění. Při výběru konkrétní účtenky se zobrazí detaily jejího obsahu. Generování šablony lze omezit jen na konkrétní kategorii položek.

Seznam položek Zobrazuje seznam všech položek v systému, umožňuje vyhledání podle názvu položky, její ceny, nebo její kategorie. Jsou zde nástroje pro deaktivaci položky, kde položka zmizí z kasy i ze skladu, její znevidtelnění, kde není vidět jen na pokladně, případně její odstranění ze systému. Pro každou položku vzniká možnost její úpravy, která provede přesměrování na formulář úprav.

Přidat položku Obsahuje formulář pro přidání položky s inputy pro její název, kategorii, nadřazenou položku, nákupní a prodejní cenou, sazbou DPH a případně s jejím slovním popisem.

Správa kategorií Zobrazuje stejný pohled jako Seznam položek, ale tentokrát pro kategorie. Navíc obsahuje formulář pro přidání nové kategorie, který zde pro svoji jednoduchost není na samostatném pohledu.

Stavy pokladny Tento pohled rozšiřuje informace ze seznamu účtenek o záznamy výběrů a příjmů hotovosti a zobrazuje tak kompletní pohyby hotovosti v kase za určité období.

Účetnictví export Pro měsíce, u nichž došlo k nějakým operacím, umožňuje tento pohled export účetnictví do tabulkového procesoru, zvlášť pro náklady a zvlášť pro příjmy.

Seznam kreditních uživatelů Zobrazuje seznam kreditních uživatelů a pro každého z nich přidává nástroje pro smazání, deaktivace a exportu všech položek, které za své kredity nakoupil. Kreditní uživatelé mohou nabývat různých slev a musejí tak mít možnost být odděleni od zbytku účetnictví.

3.3 Návrh funkcí

Některé funkce jsem musel přizpůsobit specifikaci zadavatele, naopak některé funkce očekávané v restauračním pokladním systému byly pro jejich nevyužití opomenuty. Postupem času bylo v rámci konzultací se zadavatelem pozměněno původní zadání v absenci evidence účtů ke konkrétním stolům a s možností přesouvat a slučovat účty. Bylo tak učiněno z důvodu, že provozovna umožňuje pouze objednávku u baru a tyto funkce by se nepoužívaly. Naopak byla přidána funkce pro vkládání přijatých a vydaných faktur a funkce evidence fyzického stavu hotovosti v pokladně, aby byla jasně viditelná denní tržba a byly jednoduché denní inventury.

3.3.1 Původní specifikace zadavatele

Po první konzultaci se zadavatelem se stanovily základní cíle a specifikace na požadavky funkcí systému, které bylo nutno implementovat. V závazné smlouvě byla specifikace následovná:

- Systém bude zhotoven jako webová aplikace s MySQL databází, bude mít možnost více spuštěných instancí pokladny i skladu současně.
- Pokladní systém a jeho uživatelské rozhraní musí být maximálně optimalizované pro dotykový display a různá rozlišení (AllInOne PC, tablet)
- Pokladní systém bude moci tisknout daňové doklady, objednávky do kuchyně a výdejové kupony pro výdej zboží mimo prostory restaurace. Tisk bude probíhat pomocí termotiskárnen.
- Pokladní systém bude moci spravovat účtenky jednotlivým osobám, nebo ke konkrétním stolům. Položky i celé účty bude moci vzájemně slučovat, či přesouvat.
- U každé položky na pokladně bude srozumitelně viditelný zbývající počet položek na skladě (i v případě nepočitatelného zboží).
- Pro potřeby hráčů a zaměstnanců bude součástí systému správa kreditů, které budou sloužit k placení na pokladně místo skutečné měny. Kredity bude možno předplácet i evidovat případný dluh.
- Skladní systém bude evidovat aktuální stav jednotlivých položek s okamžitou vazbou na každý prodaný kus na pokladně. Zároveň musí evidovat i neomezenou historii v nastavitelném rozsahu. V daném rozsahu musí být u každé položky srozumitelně vidět její celkový obrat, zisky, náklady, i případné ztráty.
- Naskladňování položek a exporty systému musí být z/do Microsoft Excel, případně jiného opensource formátu.

3.3.2 Uživatelské role

Jelikož byl systém rozdělen na pokladní a skladovou část, lze očekávat, že operace s nimi budou provádět uživatelé na různých pozicích a řadoví prodavači by tak neměli mít přístup k některým operacím spojeným s managementem prodejny. Zavedl jsem tedy tři uživatelské role – pokladník, manažer a administrátor. Pokladník může vykonávat všechny operace

s modulem pokladny, ale nemá přístup ke skladním funkcím. K těm má přístup manažer, který dědí práva pokladníka. Nejvyšší práva má administrátor, který dědí práva obou předchozích a může vytvářet nové uživatele, měnit jim oprávnění nebo je smazat či deaktivovat. V návrhu je počítáno s tím, že by se v budoucnu mohly uživatelské role nebo jejich oprávnění měnit. Správa uživatelů a jejich rolí je oddělena od zbytku aplikace.

Kapitola 4

Implementace

Tato kapitola je věnována implementaci jednotlivých modulů aplikace. Již ze specifikace zadavatele bylo nutné, aby byla aplikace implementována jako webová aplikace, přestože by implementace v některém z vyšších programovacích jazyků s kompilací do binárního kódu mohla být efektivnější. Z dostupných technologií jsem zvolil PHP s frameworkem Nette pro implementaci serverové části. Databáze byla zvolena MySQL. Pro klienta byl zvolen základ stránky v HTML5 doplněný zobrazením pomocí CSS3. Aplikační funkce klientské části jsou implementovány za pomocí JavaScriptu a jeho knihovny jQuery. Samostatná část implementace je věnována tisku za pomocí termotiskárny, který jsem implementoval v jazyce `C#`. Vzhledem k rozsáhlosti zde nebudu popisovat funkci každé implementované třídy, metody, nebo atributu. Jejich popis je dostupný v komentářích zdrojového kódu. Popisovat zde budu pouze obecné principy, zajímavé části nebo konstrukce.

4.1 Použité technologie

Při implementaci byly použité technologie běžné při vývoji webových aplikací. Konkrétní zástupci byli voleni se zaměřením na aktuální trendy v oblasti webdesignu, na dobrou podporu pro AJAX a samozřejmě i na čitelnosti kódu a pohodlí při programování.

4.1.1 PHP

PHP (rekurzivně PHP: Hypertext Preprocessor) je skriptovací jazyk sloužící k tvorbě webových aplikací, které jsou interpretované na serveru, a uživatel vidí jen jejich výsledek. Vytvořil jej roku 1995 Rasmus Lerdorf a v současnosti je produkován společností The PHP Group. Od začátku jeho vývoje nebylo cílem vytvořit zcela nový jazyk, ale maximalizovat jeho užitečnost pro uživatele, kteří mohou vytvářet silné aplikace i s minimem znalostí. Základní implementace totiž obsahuje více než 200 knihoven pro práci s databázemi, práci s textem, grafické operace, manipulaci s informacemi a dalšími funkcemi pro efektivní tvorbu dynamických webových stránek [4]. PHP je vyvíjeno jako opensource, je multiplatformní a jedná se o jeden z nejrozšířenějších programovacích jazyků současnosti. Pro částečně komplikované ladění v PHP psaných aplikacích a zjednodušení implementací často používaných konstrukcí je vyvíjeno mnoho PHP frameworků.

4.1.2 Nette Framework

Pro efektivnější programování nad PHP byl zvolen Nette Framework [8] původně vytvořený Davidem Grudlem a nyní rozvíjený organizací Nette Foundation. Nette jsem zvolil z důvodu dřívějších zkušeností s tímto frameworkem z jiných projektů, ale i pro jeho celou řadu výhod: Nette a jeho početné pluginy jsou opensource dostupný pod licencemi New BSD a GNU GPL. Framework umožňuje rozdělit aplikaci podle Model-View-Presenter návrhového vzoru a dále nutí používat OOP a dobré programovací návyky. Interpretovaný kód je v něm akcelerován pomocí cashe, což přispívá k efektivitě aplikace. Dále umožnuje částečnou typovou kontrolu, i když PHP je netypované. Pro testování a ladění aplikace disponuje celou řadou kvalitních ladících nástrojů. Přispívá k bezpečnosti aplikace a má velmi dobrou podporu pro AJAX, který byl při implementaci často využíván. V neposlední řadě má dobrou podporu komunity, která dovede poradit při problémech s jeho použitím.

4.1.3 HTML5

HTML5 [6] je stejně jako jeho předchůdci značkovací jazyk pro presentaci obsahu WWW stránek. Jeho specifikace je vytvářena konsorciem W3C a organizací WHATWG. Jeho současná specifikace vychází z jeho předchůdců HTML 4.01 and XHTML 1. Změny, které přináší, jsou dány hlavně změnou přístupu k webovým stránkám jako statickému obsahu na interaktivní multimediální obsah. Za tuto změnu pohledu přispěla velkou mírou technologie Adobe Flash ve spojení se službami jako je YouTube. HTML5 tedy přináší nové značky `<audio>` a `<video>` pro integraci přehrávačů hudby a videa, které tak přináší multimediální obsah pro mnohem širší škálu zařízení. Pro jasnější definici struktury stránky, které by měla přispět k lepší optimalizaci pro vyhledávače, přináší HTML5 značky `<header>` `<footer>` `<article>`. Na současných webech se vyskytuje velké množství formulářů a značka `<input>` tak byla rozšířena o atributy typu dates, times, email, url, search, number, range, tel a color, které mají zjednodušit vkládání a validování těchto hodnot. Důležitou novinou je také blokový element `<canvas>`, který vytváří programovatelné prostředí.

4.1.4 CSS3

CSS3 (Cascading Style Sheets 3) [12] je jazyk, který popisuje vizuální formát aplikací psaných v některém ze značkovacích jazyků. Jeho specifikace je také vytvářena konsorciem W3C. Specifikace CSS3 je na rozdíl od CSS2, která je vytvořena jako jeden velký celek popisující řadu rozdílných funkcí, rozdělena do několika modulů. Některé tyto moduly popisují vlastnosti CSS2 pro zachování zpětné komptability, jiné přidávají vlastnosti zcela nové. Modularizace umožnila nezávislý vývoj a specifikace CSS3 se tak postupně stále rozšiřuje a než bude finální, měla by přinést ještě okolo čtyřiceti nových modulů. Mezi hlavní novinky patří základní grafické transformace obrazu (rotace, zkosení, atd.), možnosti definování vlastních typů písma, vytváření barevných gradientů a zejména v této práci využitý modul Media Queries. Ten umožňuje adaptabilní vykreslování webových stránek podle různých aspektů jako je výška a šířka rozlišení prohlížeče, výška a šířka rozlišení zařízení a orientace zařízení na výšku nebo na šířku.

4.1.5 AJAX a JQUERY

Původní JavaScript přispěl v hojné míře k zvýšení dynamičnosti stránek, ale měl limity, co se týče získávání dat ze serveru za běhu na pozadí aplikace. AJAX (Asynchronous JavaScript

and XML) umožnil získávat části stránky, místo generování stránky celé. Je možné vytvářet stránky obsahující jen jedinou stránku, kde je na pozadí poslán http dotaz na určitou část stránky, která je přijata s odpovědí, a následně je pouze tato část stránky překreslena.

Jediný problém původního JavaScriptu byl, že různé prohlížeče nepodporují stejné funkce nebo neprovádí stejné funkce stejným způsobem, což vyžadovalo obtížné testování nad každým cíleným prohlížečem. Jako další se mohla jevit poměrně velká upovídanost kódu v JavaSriptu psaném, obzvlášť při vytváření optimalizací aplikací pro různé prohlížeče. Tyto nedostatky řeší opensource knihovna jQuery.

Knihovna jQuery [5] dokáže jednoduše adresovat HTML elementy různým způsobem, vytvářet dynamické stylování HTML elementů nebo měnit obsah jejich atributů. Dovede téměř libovolně manipulovat s DOM strukturou dokumentu a vytvářet základní animace. Nejdůležitější vlastností použitou v tomto projektu je jeho dobrá podpora práce s AJAX funkcemi. Nad jQuery je postavená spousta dalších knihoven (např. zde využitá jQuery-UI, která rozšiřuje možnosti animací a přidává podporu nejpoužívanějších widgetů).

4.2 Struktura aplikace

Vzhledem k tomu, že jsem využil Nette frameworku, má aplikace jasně danou strukturu tříd i adresářovou strukturu souborů stanovenou pravidly, které Nette doporučuje. Třídy aplikace jsou rozděleny do modulů a pro každý modul je aplikován návrhový vzor Model-View-Presenter.

4.2.1 Model-View-Presenter

Model-View-Presenter je návrhový vzor, který dělí vytvářené systémy na tři na sobě nezávislé části. Jeho implementace je velmi podobná známějšímu Model-View-Controler z něhož vychází. Rozdíly mezi nimi demonstruje obrázek 4.1.

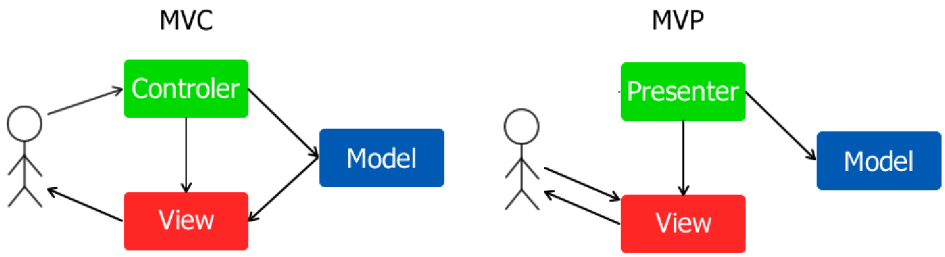
Model je datový a funkční základ celé aplikace. Veškerá data této aplikace jsou uložena v MySQL databázi a model zde slouží pro spojení s databází a provedení všech SQL dotazů, které aplikace vyžaduje. Stejně tak model obstarává funkce spojené s přihlášením a odhlášením uživatelů.

View je vrstva aplikace, která má na starost zobrazení výsledku požadavku. Zde representovaná jako šablona šablonovacího jazyka Latte. Kromě značek Latte definující podmínky, cykly a bloky získané pomocí AJAX, obsahují šablony jen části HTML5 kódu doplněného inline Javasriptem v místech, kde nebylo vhodné použít externích souborů.

Presenter je radič, který spojuje předcházející části dohromady. Presenter zpracovává požadavky od uživatele a následně upravuje data pomocí modelu nebo žádá o vykreslení dat prostřednictvím view.

4.2.2 Konfigurační soubor

Pro nastavení konfigurace slouží soubor `config.neon`. Soubor obsahuje připravené struktury pro připojení MySQL databáze a možnosti si nastavit vlastní řetězec soli používané u autorizace uživatelů. Ostatní nastavení slouží pro připojení modelů pomocí dependency injection a není je třeba jakkoliv měnit.



Obrázek 4.1: Rozdíl mezi MVC a MVP

4.3 Implementace pokladny

Registrační pokladna je implementována v modulu PosModule, který obsahuje jeden presenter `PosPresenter`, jeden model `PosQuery` a jedno view `default`. Aby byl zachován požadavek na optimalizaci pro dotyková rozhraní, použil jsem při implementaci responsivní design. Veškeré operace s daty a překreslování, vyjma stránky prvního renderu při obnovení zajišťuje AJAX.

Při renderování presenteru jsou z modelu získány kategorie, položky, otevřené účty, seznam kreditních uživatelů a stav hotovosti v kase. U položek, které jsou v poměru oproti položce jim nadřazené, je vypočítáno jejich množství na základě onoho poměru.

4.3.1 Responzivní web design

S masivním rozšířením mobilních zařízení – chytrých telefonů a tabletů, výrazně vzrostla penetrace návštěv webových stránek právě z těchto zařízení. Tato zařízení mají velmi omezené možnosti na šířku zobrazované oblasti, zejména pokud využíváme v režimu portrait – na výšku. Opakem jsou stále se rozšiřující obrazovky velkých úhlopříček u klasických monitorů, kdy lze vídat obrazovky s rozlišením šíře 2560 pixelů i vyšším. Většina webových stránek je však přizpůsobena na šířku stránky kolem 960 pixelů a jejich zobrazování na těchto zařízeních se stávají nevhodná.

Moderní prohlížeče samy dovedou obsah stránek přizpůsobovat možnostem daného zařízení, většinou jednoduchou změnou rozměrů všech prvků v určitém poměru. Toto řešení však není ideální, zejména u malých telefonů, kde se nevyhneme použití funkce zoom.

Mnohem efektivnějším řešením je Responsivní web design [3], kdy jedna a ta samá stránka je pomocí HTML5 v kombinaci s CSS3 zobrazena zcela různě podle zařízení, na kterém se zobrazuje tak, aby byla co nejvíce přizpůsobena možnostem daného zařízení. Jako jediná hlavní nevýhoda tohoto postupu se jeví nekompatibilita se starými webovými prohlížeči (Internet Explorer 6, aj.), jejichž rozšíření se naštěstí stále snižuje.

V mém případě jsem potřeboval zajistit, aby uspořádání prvků na stránce bylo stejné, ale jejich rozměry se lišily. Toho zle zajistit pomocí přepočítání velikostí prvků z pixelů na procentuální representace oproti jejich rodičovskému prvku. Původní návrh tak může být nadále statický – programy na tvorbu webových šablon (v mém případě Adobe Fireworks CS6) podporují pouze tvorbu v pixelech respektive palcích. Pro přepočet stačí použít následující vzorec:

$$x_{[\%]} = \frac{x_{[px]}}{parentx_{[px]}} * 100$$

Komplikace nastala pouze u výšky prvků, kdy pro projevení procentuální representace výšky, je třeba definovat procentuální výšku všem jeho prvkům nadřazeným včetně kořenového DOM elementu `<html>`.

Pro zamezení možnosti prohlížeče samovolně měnit rozměry všech prvků v určitém poměru, kde vznikaly drobné neshody ve velikostech prvků na různých zařízeních, bylo nutné nastavit meta tag viewport následovně:

```
<meta name="viewport"
    content="initial-scale=1.0, width=device-width">
```

Po tomto nastavení byl na všech zařízeních poměr prvků stejný, nicméně dobrá použitelnost navrženého designu byla jen v orientaci landscape. Aby byla použitelnost dobrá i v portrait režimu, bylo třeba boční část obsahující informace o položkách na otevřených účtech umístit jinam. Aby nedocházelo ke scrollování, bylo zvoleno řešení, kdy tento box vyjíždí z boku obrazovky po stisku tlačítka (viz. obrázek 4.5). Prohlížeč sám pozná, v jakém z režimů orientace se nachází a nastavením media atributu u link tagu pro načtení css souborů, jsem definoval jaké css je má použít.

```
<link rel="stylesheet" type="text/css"
    media="only screen and (orientation: landscape)"
    href="/css/screen_landscape.css" />
<link rel="stylesheet" type="text/css"
    media="only screen and (orientation: portrait)"
    href="/css/screen_portrait.css" />
```

Samotná animace vyjíždění se již netýká responsivního designu a byla zajištěna funkcemi knihovny jQuery-UI.

Posledním problémem byla velikost obrázků na pozadí tlačítek a písma v tlačítkách, která byla na malých obrazovkách příliš velká a přetékala přes okraje elementů. Řešením je @media atribut CSS souboru, který podobně jako v předchozím případě nastavuje kaskádové styly pro určité zobrazení, tentokráte uvnitř souboru jen pro konkrétní elementy a pro konkrétní maximální rozlišení. Příklad tlačítek akcí účtenky v portait reřimu:

```
@media only screen and (max-device-width: 480px) {
    #side-menu a{
        background-size: 110% auto;
    }
}
```

4.3.2 Datové operace

Při uživatelském požadavku změny dat v modelu jsou operace volány pomocí AJAX implementované pomocí jQuery metody `Get()`, která odkazuje na příslušný handler dané operace: `handleNewBill()` pro vytvoření účtu, `handleDeleteBill()` pro jeho smazání, `handleUpdateBill()` pro přidání položky k účtu, `handleDeleteItem()` pro odebrání položky z účtu a konečně nejdůležitější `handleCloseBill()` pro uzavření účtu.

Tato metoda slouží k uzavření účtenky a vytvoření daňového dokladu. Nejprve získá z databáze údaje o uzavírané účtence a jejich položkách, poté dojde k úpravě stavu účtenky na uzavřený a dostane údaj o datu a času jejího uzavření. Následně je pro každou položku naplňována kolekce tisknutých položek a počítána jejich celková suma. Pro každou položku je upraven její stav skladu, respektive pro položky, které jsou v poměru k jiné položce, je

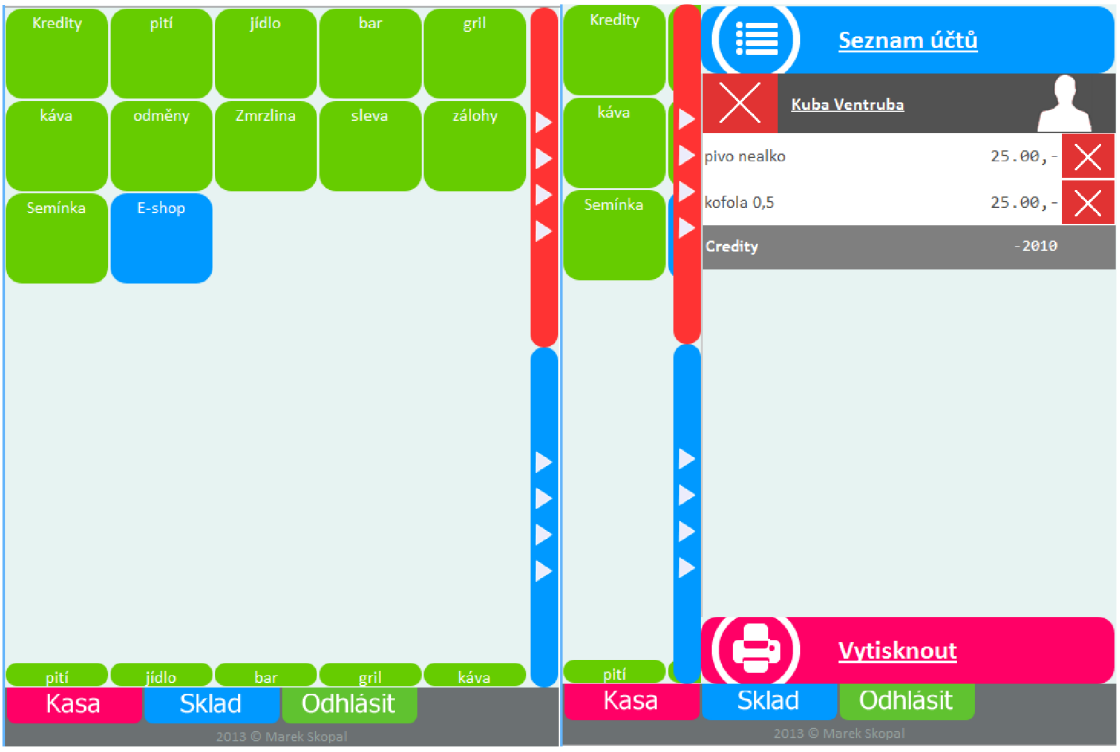
Obrázek 4.2: Aplikace pokladny v landscape režimu

upraven stav jen položky nadřazené v nastaveném poměru. Na základě toho, zdali byl k účtu přiřazen kreditní vlastník, je změněna hodnota výše kreditního zůstatku tohoto uživatele a upraven stav hotovosti v kase. U kategorií položek, kde je zvolen tisk výdejových účtenek, se připraví kolekce obsahující tyto kategorie a jejich položky. Poslední výpočet je pro celkové DPH rozdělené do skupin podle jeho sazby. Ze všech těchto údajů je nakonec pomocí latte šablony vygenerován XML soubor dle struktury 4.7.1 a uložen do složky `/bin`, odkud je ve výchozím stavu zpracováván tiskovým klientem.

4.3.3 Překreslování view

Překreslování view stránky není řešeno pomocí refreshe celé strany. Při vytvoření nového účtu, jeho uzavření nebo zrušení, jsou některé bloky překresleny pomocí AJAX. Jsou to bloky s identifikátorem `itemList` pro tlačítka s nabízenými položkami, blok `billItems` pro položky na otevřených účtech, `cashState` pro stav hotovosti a `creditList` reprezentující seznam kreditních uživatelů. O samotné překreslení se stará Nette Framework při zavolání metody `invalidateControl()` s atributem názvu překreslovaného bloku.

AJAXové dotazy trvají určitý čas (řádově desítky až stovky milisekund) a pro procházení kategorií a modifikaci položek na účtu se jevily jako pomalé. Zvolil jsem řešení, kdy jsou všechny kategorie s jejich položkami načteny při prvním renderu a jsou zneviditelněny pomocí CSS atributu `display: none;`. Zviditelnění požadované kategorie se děje pomocí Javasriptu voláním funkce `selectCategory()` a opětovné zneviditelnění funkcí `categoryList()`. Při stisku tlačítka požadované položky je do seznamu položek na aktuálním otevřeném účtu přidána položka funcí `addItem()`, která přímo vloží HTML kód řádku vykreslované tabulky a zavolá funkci pro přepočítání celkové zobrazované sumy. Paralelně s touto operací je však zavolán AJAXem hander `handleUpdateBill()` pro přidání položky do databáze. Přidávaná položka je vykreslena okamžitě už v průběhu jejího zpracování, aby mohl uživatel přidávat položky rychle, bez čekání na prováděné úpravy v databázi. Jediná



Obrázek 4.3: Aplikace pokladny v portrait režimu bez a s vysunutými informacemi o účtenkách

operace, která vyžaduje čekání. je uzavření účtenky. Při této operaci dochází ke kontrole, zdali neprobíhá nějaký paralelní AJAX dotaz, a až po jeho případném ukončení, proběhne operace uzavření účtu. O probíhajících dotazech je uživatel informován pomocí načítací animace.

4.4 Implementace skladu

Skladní část systému je implementována jako klasický informační systém, ve kterém figurují tabulky pro zobrazení dat jednotlivých pohledů a formulářové inputy pro vkládání a editaci dat v databázi. Na rozdíl od pokladního modulu, není modul skladu v jednom souboru po každé složce MVP, ale je rozdělen do více částí podle logické struktury na třídy pracující s položkami, kategoriemi, účtenkami, stavem skladu, stavem hotovosti, kreditními uživateli a účetnictvím.

4.4.1 Stránkování

Vyjma účetnictví, které obsahuje pouze export do Excelu, je obsahem každého view tabulka s položkami dané sekce. Aby se netvořily potencionálně nekonečné tabulky, jsou odstránkovány po zvoleném počtu (ve výchozím stavu 15) položek. Stránkování je implementováno jako instance třídy `VisualPaginator`, které je z databáze předán celkový počet řádků dané typu dotazu a počet položek na stránku. Na základě aktuální strany je vrácený offset předán jako parametr pro metodu zpracování samotného SQL dotazu, ze které jsou získány

jen řádky odpovídající dané stránce. Jsou tak rychle provedeny dotazy i v tabulkách které potenciálně budou obsahovat řády statisíců řádků.

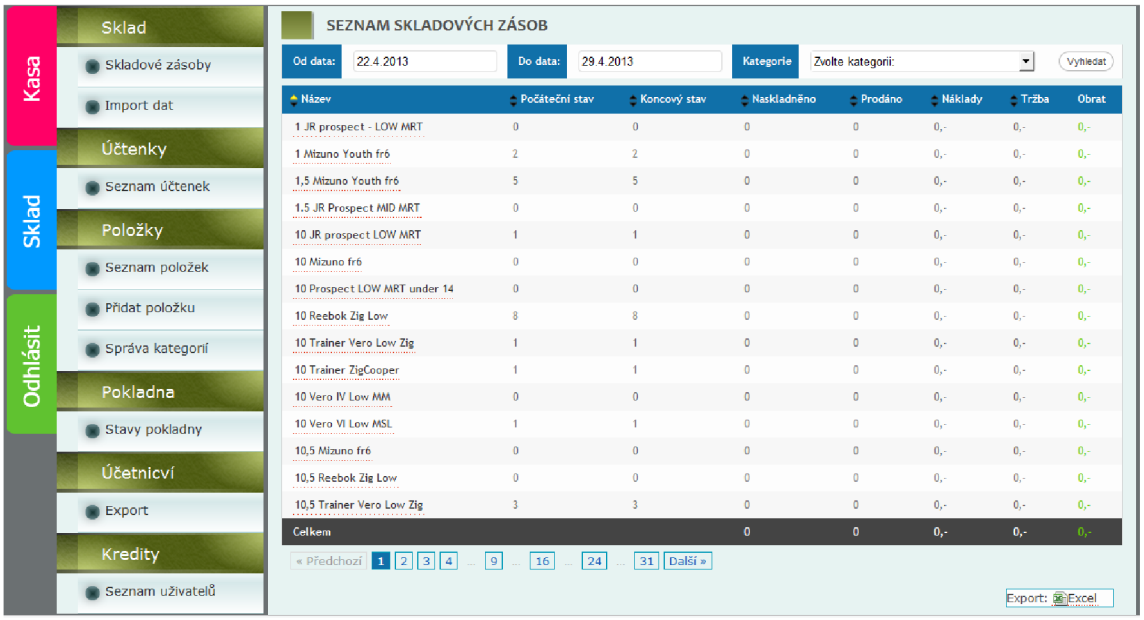
4.4.2 Validace formulářů

Vkládání a editace položek, kategorií, uživatelů atd. je řešena pomocí formulářů. Aby uživatel vyplnil všechny povinné pole formuláře, a aby nedocházelo k pokusům o vkládání dat jiných datových typů, než jaké jsou v návrhu databáze, měla by být provedena validace vložených dat.

Zde je aplikována live validace, kdy je uživatel informován o případných chybách ve vkládání ještě před samotným odesláním formuláře. Formulář jde odeslat teprve, pokud jsou všechna vyplněná data korektní. Validační pravidla jsou definována při vytváření instance formuláře metodou `addRule()` obsahující typ podmínky a případnou zprávu při jejím porušení. Při samotné validaci Nette využívá nových vlastností HTML5 pro definici typu `<input>` a část kontroly tak probíhá už na straně prohlížeče. Live validace je pak řešena addonem LiveValidation s doplněním definice zvýraznění neplatných validací do CSS.

4.4.3 Zprávy a chybová hlášení

Při provedení nějakých akcí je uživatel informován o jejich úspěchu nebo neúspěchu formou zprávy. Žlutá barva značí informativní, červená kritickou. Stejně tak je pomocí těchto zpráv uživatel vyzván k provedení potvrzení, je-li vyžadována zásadnější změna v systému např. odebrání položky. Tato potvrzení jsou implementována pomoci Nette addonu ConfirmationDialog, který zabezpečuje kritickou akci proti CSRF útoku a neumožnuje změnit parametry akce.



Obrázek 4.4: Aplikace skladového systému

4.5 Autorizace uživatelů

Pro přístup do systému je využito systému autentifikace pomocí uživatelského jména a hesla. Ta probíhá před všemi dalšími operacemi a je definována v abstraktní třídě `BasePresenter` od níž dědí všechny další presentery. Pokud není uživatel přihlášen, je přesměrován na přihlašovací formulář. Po vyplnění formuláře jsou všechny přihlašovací údaje předány modelu `UserAuthenticator` implementující rozhraní `\Nette\Security\IAuthenticator`. Při přihlášení je jméno převedeno na malé znaky a heslo posláno hashovací funkci. Poté je jméno a hash hesla porovnán s uživatelskými údaji v databázi. Hash je typu sha256. Pro zvýšení síly zabezpečení je zde použité dvojí nasolení. První sůl je konkatenována s přijatým heslem z formuláře. Sůl je uložena v databázi u konkrétního uživatele opět jako hash typu sha256 vytvořená při registraci uživatele z náhodně vygenerovaného čísla a měla by zabraňovat slovníkovým útokům. Druhá sůl je získaná z konfiguračního souboru aplikace. Tato sůl slouží jako klíč do hashovací funkce při vytváření celkové hashe a odstraňuje duplicity v databázi při stejných heslech. Uložením na dvou odlišných úložištích se navíc velmi znesnadňuje možnost zcizení.

Hashovací funkce vypadá obecně následovně:

```
hash_hmac('sha256', $credentials['password'] . $database->pass_salt,
    $config->salt);
```

Při úspěšném přihlášení vrací autentifikátor instanci třídy `\Nette\Security\Identity` o jejíž předávání se stará Nette Framework.

Autorizace uživatele probíhá přes instanci třídy `UserPermmision`, která pro jednotlivé uživatelské role definuje oprávnění, jak lze nakládat se zdroji. Zdroje jsou jednotlivé logické části systému (položky, účty, stav skladu atd.). Definovaná oprávnění jsou pro každý render porovnávány metodou `testUserPermission()` a při zobrazení ve view jsou části systému, ke kterým uživatel nemá přístup, zneviditelněny v šabloně Latte makrem `isInRole`.

4.6 Databáze

Pro uložení veškerých dat je využito MySQL databáze. Struktura databáze je rozdělena do devíti tabulek. Jelikož obsahuje tranzitivní vazby, je pouze v druhé normální formě. Aby nedošlo se záměnou s tabulkami z jiných aplikací, obsahují názvy tabulek prefix `mc_`.

mc_bills_confirm obsahuje informace o účtenkách – datum vytvoření účtenky, datum jejího uzavření, stav (0 – otevřená, 1 – zaplacená v hotovosti, 2 – zaplacená credit kartou), cizí klíč do seznamu kreditních vlastníků (nulový pokud vlastníka nemá) a evidenční číslo, aby se daly měnit série účtenek. Např. pro rok 2013 budou účtenky ve tvaru 201300001, 201300002 atd., ve výchozím nastavení však číslování odpovídá sloupci s primárním klíčem.

mc_bills_item obsahuje seznam položek na účtenkách – cizí klíč do účtenek, cizí klíč do seznamu položek a cizí klíč do cen položek, protože stejná položka může libovolně měnit svoji cenu a na účtence je potřeba evidovat cenu, kterou měla položka v době jejího prodání.

mc_cash obsahuje vklady a výběry hotovostí, jejich datum, výši, pojmenování, evidenční číslo a stav hotovosti v kase v době vložení řádku této tabulky. Tabulka tak neobsahuje veškeré pohyby hotovosti v pokladně, a pokud je následně nutno spočítat aktuální

stav pokladny nebo stav k nějakému datu, je z této tabulky vybrána hodnota s časem nejbližším před žádaným datem, a je k této hodnotě přičtena suma cen prodaných položek do cíleného data.

mc_categories obsahuje kategorie položek, jejich názvy, stav (0 – zrušeno, 1 – aktivní, 2 – deaktivováno, 3 – zneviditelněno) a příznaky pro zobrazení v rychlých volbách a tisk výdejového ústřižku.

mc_credit obsahuje seznam kreditních uživatelů, počet jejich kreditů a stav (1 – aktivní, 2 – deaktivováno)

mc_item obsahuje položky, jejich názvy, stav (viz. kategorie), odkaz na primární klíč nadřazené položky a jejich vzájemný poměr jako desetinné číslo (nulový pokud žádný nemá). Dále obsahuje počet kusů na skladě. Na základě množství kusů skladem systém informuje o docházení dané položky. Poslední sloupec udává pořadí položky při zobrazování (nulová pokud je zachováno výchozí nastavení a položka není upřednostněna před jinými).

mc_prices obsahuje seznam cen – jejich nákupní hodnotu, prodejní hodnotu a výši DPH a cizí klíč do tabulky položek.

mc_stock obsahuje všechny pohyby na skladě. Uchovává pro ně datum a čas jejich uskutečnění, množství kusů na skladě, počet naskladněných nebo prodaných položek a cizí klíč na jejich cenu, aby se dala spočítat výše nákladů v konkrétním čase.

mc_users obsahuje informace o uživatelích a jejich přihlašovacích údajích.

4.6.1 Optimalizace

V žádném ze svých předchozích projektů jsem nepracoval s SQL tabulkami čítajícími více než tisíce řádků. V návrhu databáze jsem se proto nevěnoval optimalizaci, což se později ukázalo jako zásadní problém. Zhruba po měsíci ostrého provozu zpozoroval klient citelné zpomalení systému. Po provedení základního profilingu, jsem zjistil, že zpomalení způsobují některé databázové dotazy, kde bylo nevhodně zvolené spojení více tabulek, nebo nebyly zavedeny indexy. Také jsem uvažoval změnu typu MySQL enginu InnoDB za starší MyISAM, ve kterém by SELECTy měly být rychlejší, ale jelikož systém pracuje s finanční hotovostí, musel jsem zachovat podporu transakcí, které MyISAM nepodporuje [9].

Hlavní chybou bylo, že při spojování dvou tabulek operací LEFT JOIN jsem špatně zvolil tabulku, ze které je prováděn SELECT, a tabulku, se kterou je spojována, byť výsledek dotazu by při opačném spojení byl zcela stejný. Exponenciálně jsem tak zvýšil složitost dotazu. V následujícím příkladu, kde byla navíc velmi nešikovně provedena sumarizace, vzrostla rychlost dotazu pro cca 8000 řádků tabulky z 9s na 80ms, což je více než o 10000%.

```
SELECT
SUM(count) AS count,
SUM(price) AS price,
SUM(credit) AS credit,
SUM(cash) AS cash FROM (
    SELECT
    COUNT(iid) AS count,
    SUM(IF(iid > 10,price_wt,0)) AS price,
```

```
    IF(cid,(SUM(IF(iid <= 10,price_wt,0)) -
        SUM(IF(iid > 10,price_wt,0))),0) AS credit,
    (IF(cid, SUM(IF(iid <= 10,price_wt,0)), SUM(price_wt))) AS cash
    FROM fh_bills_confirm NATURAL LEFT JOIN fh_bills_item
```

oproti optimalizovanému

```
SELECT
COUNT(iid) AS count,
SUM(IF(iid > 10,price_wt,0)) AS price,
SUM(IF(cid,(IF(iid <= 10,price_wt,0) -
        IF(iid > 10,price_wt,0)),0)) AS credit,
SUM(IF(cid, IF(iid <= 10,price_wt,0), price_wt)) AS cash
FROM fh_bills_item NATURAL LEFT JOIN fh_bills_confirm
```

Dále byly do některých tabulek zavedeny indexy, které také nemalou měrou přispěli k efektivitě SELECTů, např. v tabulce `mc_stock` byl přidán index typu BTREE pro sloupec `item_id`.

4.7 Implementace tisku

Po seznámení se se zadavatelem dodanou tiskárnou STAR TSP143U ECO a prostudování nabízené dokumentace pro vývojáře, jsem došel ke zjištění, že pro tisk je třeba použít dodávaného SDK, které je k dispozici pro několik platforem a programovacích jazyků. PHP však mezi nimi není, a protože klient bude aplikaci provozovat pod Windows, byl pro implementaci tisku zvolen jazyk `C#`.

Cílem bylo však umožnit do budoucna tisk pod libovolnou platformou, samozřejmě i podpora tiskáren jiných výrobců musela připadat v úvahu. Musel jsem tedy vytvořit rozhraní mezi PHP aplikací a aplikací starající se o tisk. Zvolen byl výstup do jazyka XML, který má velmi dobrou knihovní podporu u většiny vyšších programovacích jazyků a je dobře čitelný i člověkem.

4.7.1 Popis XML komunikace

Pro budoucí implementaci dalších tiskových klientů, pro jiné platformy nebo tiskárny jiného výrobce je třeba popsat použité XML konstrukce.

Kořenový element `<bill>` obsahuje povinně elementy:
`<bid>` – obsahuje číselný identifikátor účtenky
`<date>` – obsahuje datum účtenky ve tvaru DD.MM.YYYY
`<time>` – obsahuje časové razítko účtenky ve formátu HH:MM
`<items>` – element obsahující libovolný počet elementů `<item>`
`<item>` – obsahuje textový popis skupiny jednoho typu položek, obsahuje atribut `iid` – číselný identifikátor typu položky, `price` – suma vč. DPH, `quantity` – počet položek
`<taxes>` – element obsahující libovolný počet elementů `<tax>`
`<tax>` – obsahuje číselnou výši daně, má atributy `price_wt` – suma vč. DPH, `price_nt` – základ daně, `tax` – daň

Dále může obsahovat nepovinné elementy:
`<credits>` – element obsahuje povinně podentity `<name>` a `<credit>`
`<name>` – jméno kreditního uživatele (vlastníka účtenky)

`<credit>` – číselný zůstatek kreditů daného uživatele
`<issues>` – obsahuje libovolný počet elementů `<issue>`
`<issue>` – reprezentuje jeden výdejový ústřižek a obsahuje povinně elementy `<name>` a `<items>`
`<name>` – název výdeje
`<items>` – má stejný význam jako `<items>` v kořenovém elementu

Příklad XML souboru obsahujícího výdejový ústřižek:

```
<?xml version="1.0" encoding="UTF-8"?>
<bill>
    <bid>375</bid>
    <date>14.04.2013</date>
    <time>16:36</time>
    <credits>
        <name>Mika</name>
        <credit>390</credit>
    </credits>
    <items>
        <item iid="273" price="95.00" quantity="1">
                kureci nugety, hranolky velka
        </item>
        <item iid="274" price="75.00" quantity="1">
                kureci nugety, hranolky mala
        </item>
    </items>
    <sum>170.00</sum>
    <taxes>
        <tax price_wt="170.00" tax="29.50" price_nt="140.50">
                21
        </tax>
    </taxes>

    <issues>
        <issue>
            <name>jidlo</name>
            <items>
                <item quantity="1">
                        kureci nugety, hranolky velka
                </item>
                <item quantity="1">
                        kureci nugety, hranolky mala
                </item>
            </items>
        </issue>
    </issues>
</bill>
```

4.7.2 Implementace tiskového klienta

Mnou vytvořený tiskový klient nazvaný TermalPrinterDaemon je WinForms aplikace implementovaná v jazyce `C#` spouštějící se po startu Windows. O své existenci dává informaci jen pomocí icony v tray oznamovací oblasti, aby nijak nerušila uživatele při činnosti. Je však možné ji dvoj-clickem aktivovat do okenního režimu, kde umožňuje nastavení cílové tiskárny a tisk zkušební účtenky pro kontrolu správného zapojení tiskárny. Dále dává jednoduchou informaci o poslední vytištěné účtence. Aplikace je rozdělaná do dvou projektů – GUI aplikace a knihovny pro práci s tiskárnou.

GUI část definuje funkci tlačítek na zapnutí a vypnutí aplikace, tisk testovací účtenky pro kontrolu správného zapojení a selectboxu pro zvolení konkrétní výstupní tiskárny, pokud je v systému nainstalováno více tiskáren. Funkce aplikace je následující: Po inicializaci aplikace dochází k zavolání nekonečné smyčky pomocí `BackgroundWorker`, která je volaná v samostatném vláknu a po jednom průchodu na 10 milisekund pozastavena, aby docházelo k co nejmenšímu vytížení procesoru. Ve smyčce dochází k prozkoumání cílového adresáře a pro všechny XML dokumenty obsahující informaci o účtence zavolání knihovní části a jejich následné odstranění z adresáře. Při otevírání souboru je kontrolováno, zda do souboru právě nezapisuje PHP server. Při stisku tlačítek jsou vykonávány jejich obslužné rutiny bez přerušení hlavní činnosti, vyjma rutiny tlačítka zastavit.

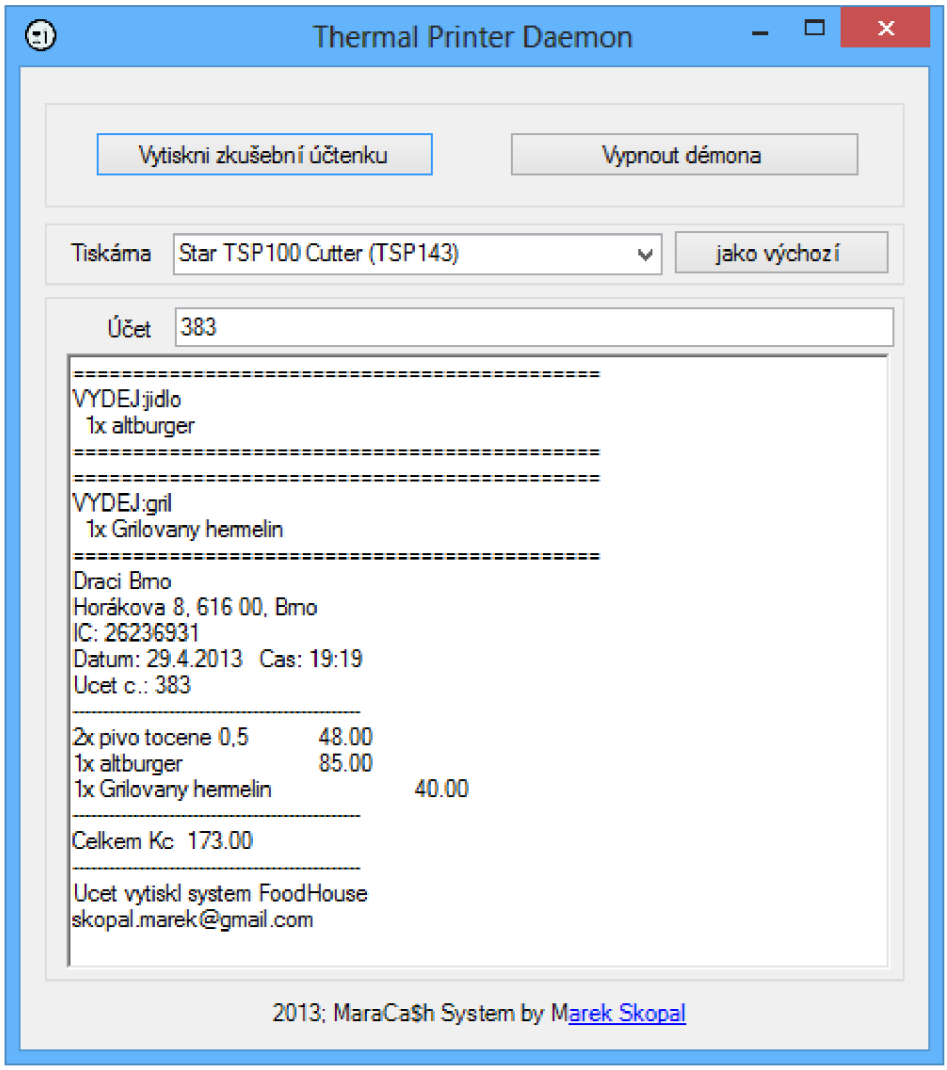
Knihovna pro práci s tiskárnou nejprve rozparsuje předaný XML soubor a naplní atributy instanci třídy `Bill`. Poté jsou zavolány statické metody třídy `PrinterText` pro vytvoření textového řetězce, který bude poslán tiskárně doplněný o instrukce tiskárny pro orientaci odstavců, nastavení zarážek a ovládání řezacího nože podle datasheetu [10] dodávaného výrobcem Star Micronics. Poté je poslaná funkci `SendStringToPrinter` třídy `RawPrinterHelper`, která zavolá operace samotného tisku. Třídu `RawPrinterHelper` jsem převzal z dodávaného SDK.

4.8 Export / import

Klientem velmi požadovaná funkce byl jednoduchý export a import dokumentů do tabulkového procesoru. Klient používá Microsoft Excel, ale já jsem chtěl zajistit podporu i u konkurenčních řešení. Původní snaha byla implementovat podporu pro Office Open XML. Pro jeho velké nároky na znalost specifikace a nenalezení dobré PHP knihovny pro tvorbu dokumentů v tomto formátu, byl zvolen starší formát Microsoft Office XML, který má také dobrou podporu v jiných kancelářských balících. Zvolený formát nicméně nepodporuje všechny funkce, které Excel nabízí, a pro některé funkce je limitován [7].

Soubor zvoleného formátu je klasický soubor v jazyce XML. Export je proto řešen vykreslením pomocí view a není zde využito žádné PHP knihovny pro vytváření XML dokumentů. Aby se neprovedlo zobrazení obsahu souboru v prohlížeči jako u běžné webové stránky, musel jsem ručně změnit HTTP hlavičku. Soubor je tak automaticky nabízen ke stáhnutí. Kromě datové části obsahující souhrnné informace o stavu skladu, tržbách nebo nákladech, je v souboru definován styl dokumentu. Stylovány jsou nadpisy tabulek a celkových sum. Pro zpřehlednění tabulky je vytvořeno i střídání typu lichých a sudých řádků.

Importování je prováděno jen v případě úpravy stavu skladu. XML soubor je uploadován pomocí formuláře a funkcí `simplexml_load_string()` je vytvořena instance objektu s jeho strukturou. Pro položky u nichž došlo ke změnám, je poté provedena změna v databázi.



Obrázek 4.5: Tiskový klient

Kapitola 5

Testování a nasazení

Během vývoje aplikace vznikala řada testů na funkce jednotlivých částí systému postupně, jak byly implementovány. Nejprve na základní funkce skladu, abych vytvořil množinu testovacích dat pro pokladnu. Ta byla implementována a testována až následně na to. Poté jsem provedl prezentaci zadavateli a po odsouhlasení došlo s nákupu hardware počítače, na němž by pokladna měla běžet, a také termotiskárny, kterou jsem větší část vývoje neměl k dispozici.

Od této chvíli probíhalo testování výhradně na cílovém stroji, kde bylo nejprve třeba vybrat a nakonfigurovat http server, PHP server a MySQL databázi. Klient si přál provozovat server pod Windows 7 a při výběru http serveru jsem dal přednost IIS7 proti Apache2. Komplikovanější byla volba použitého PHP serveru, kdy verze získané přímo z php.net byly i s použitím akcelerátoru velmi pomalé. Přestože nejsem zastánce all-in-one řešení, konečná volba padla na Zend Server 6.0.1 s PHP 5.3, který dosahoval o celý řád lepších výsledků.

Po otestování tiskového klienta byl stroj s aplikací vrácen zadavateli a ten prováděl týdenní intenzivní testování všech funkcí a upozorňoval na problémy či nedostatky. Po jejich odstranění se již systém začal používat v ostrém provozu.

5.1 Vyjádření klienta

Pro zhodnocení spolupráce při vývoji aplikace jsem zadavateli položil několik otázek. V zastoupení společnosti Draci Brno na ně odpověděl Bc. Jan Altrichter.

Jak hodnotíte postup a kvalitu konzultací během vývoje? Byly splněny všechny Vaše požadavky na specifikaci systému?

„Četnost a rozsah konzultací byli pravděpodobně odpovídající požadavkům. Bylo třeba zohlednit, že naše představa o pokladním systému byla velmi rámcová a hlavní iniciativa tak byla z vaší stany. Potěšilo nás, že jste reagoval na veškeré naše podměty a velkou měrou přispěl i vlastní invencí.“

Byli jste spokojeni s kvalitou testování? Jaká v něm byla Vaše role?

„Poté, co jste nám prezentoval funkce betaverze, jsme zakoupili hardware. Na dotykový all-in-one počítač jste naistaloval pokladní systém a připravil testovací data. Tyto testovací data jsme rozšířili o svoje a několik dní testovali na námi požadovanou funkčnost. Tento postup nám vyhovoval, protože jsme se mohli vyjádřit k funkčnosti ještě před dokončením aplikace. Při testování byli odhalené drobné chyby týkající se operací s kreditními účty, které jste obratem odstranil.“

Jaké byly Vaše dojmy z ostré verze?

„Systém odpovídal žádaným úpravám na funkčnost vyplývající z našeho testování. Zpočátku se aplikace zdála býti rychlá a spolehlivá, ale s postupem času začal být systém velmi pomalý a od určitého momentu skoro nepoužitelný. Poměrně rychle jste však zjistil příčinu a odstranil ji.“

Nyní jste s tedy produktem spokojení?

„Ano, nyní když se odladily počáteční chyby jsme se systémem poměrně spokojení. S přihlédnutím na to, jaké nám pokladní systém přináší výhody, bychom bez něj již nejspíše nedovedli fungovat.“

Máte nějaké návrhy na možná zlepšení, nebo přidání funkcí systému?

„Používáním jsme zjistili, že několik funkcí by bylo dobré přidat. Bylo by dobré, kdyby byla možnost zobrazit více položek v jedné kategorii než v současnosti. S aktuálním množstvím kreditních uživatelů bychom uvítali možnost vyhledávání v kreditních uživatelích a možnost jejich řazení.“

Mohu tedy očekávat další spolupráci?

„Ano, pokladní systém budeme od vás chtít dále rozšiřovat. Navíc díky velké spokojenosti s vaším přístupem a s kvalitou vytvořeného produktu zahájíme předběžnou domluvu na dalším produktu týkajícím se tvorby sportovního webového portálu.“

Kapitola 6

Závěr

Na základě specifikace klienta Draci Brno, analýzy trhu konkurenčních řešení a průběžných konzultacích při vývoji, jsem navrhl a implementoval systém registrační pokladny pro restaurační zařízení s podporou dotykových obrazovek různých zařízení. Dále byl implementován jednoduchý klient pro tisk pomocí termotiskárny na platformě Windows. Aplikace svojí multi-platformností a snadnou možností rozšíření pro potřeby konkrétního zákazníka vyplňuje místo na trhu s pokladními systémy. Klient Draci Brno systém používá při běžném denním provozu, a i přes jisté počáteční komplikace způsobené nedostatečnou optimalizací databázových dotazů a indexu databáze je se systémem spokojen a doufá v další spolupráci, jak na vývoji tohoto systému, tak na projektech jiných.

Aplikace se bude nadále rozvíjet nad rámec této bakalářské práce. Vznikne podpora tisku na dalších platformách (Android). V době psaní tohoto textu projevili o systém zájem již další dva zákazníci, kteří by také chtěli mít v systému některé funkce pro jejich specifické potřeby a aplikaci by tak bylo vhodné rozšířit o podporu zásuvných modulů, aby se dalo jednoduše vybírat mezi funkcemi, které konkrétní verze bude obsahovat.

Literatura

[1] Brobeck, W. M.; Givins, J. S.; Meads, P. F.; aj.: Point-of-sale system and apparatus [online]. http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=3946220.PN.&OS=PN/3946220&RS=PN/3946220, 06 1974.

[2] Burke, A.: How Do Thermal Printers Work [online]. http://satoprinter.danielcadams.com/how-do-thermal-printers-work/, 01 2013.

[3] Frain, B.: *Responsive Web Design with HTML5 and CSS3*. Addison-Wesley UK, 04 2012, iSBN 1849693188.

[4] Gilmore, W. J.: *Beginning PHP and MySQL: From Novice to Professional*. Apress, 2010, iSBN 978-1-4302-3114-1.

[5] Jenkov, J.: *jQuery Compressed*. Jenkov Aps, 02 2013.

[6] McLaughlin, B.: *What Is HTML5?* O'Reilly Media, 07 2011.

[7] Microsoft Corporation: Features and limitations of XML Spreadsheet format [online]. http://office.microsoft.com/en-us/excel-help/features-and-limitations-of-xml-spreadsheet-format-HP001073362.aspx, 2003.

[8] Nette Foundation: Nette Framework [online]. http://nette.org/cs/, 2013.

[9] Schneider, R. D.: *MySQL Oficiální průvodce tvorbou, správou a laděním databází*. Grada Publishing a.s., 2006, iSBN 80-247-1516-3.

[10] Star Micronics: Star Line Mode Command Specifications [online]. http://www.starmicronics.com/support/Mannualfolder/starline_cm_rev1.15_en.pdf, 04 2013.

[11] Studio 7.5; Zwick, C.; Schmitz, B.: *Designing for Small Screens: Mobile Phones, Smart Phones, PDAs, Pocket PCs, Navigation Systems, MP3 Players, Game Consoles (Required Reading Range)*. Academia, 2006, iSBN 978-2940373079.

[12] Weyl, E.: *What's New in CSS3*. O'Reilly Media, 09 2012.

Příloha A

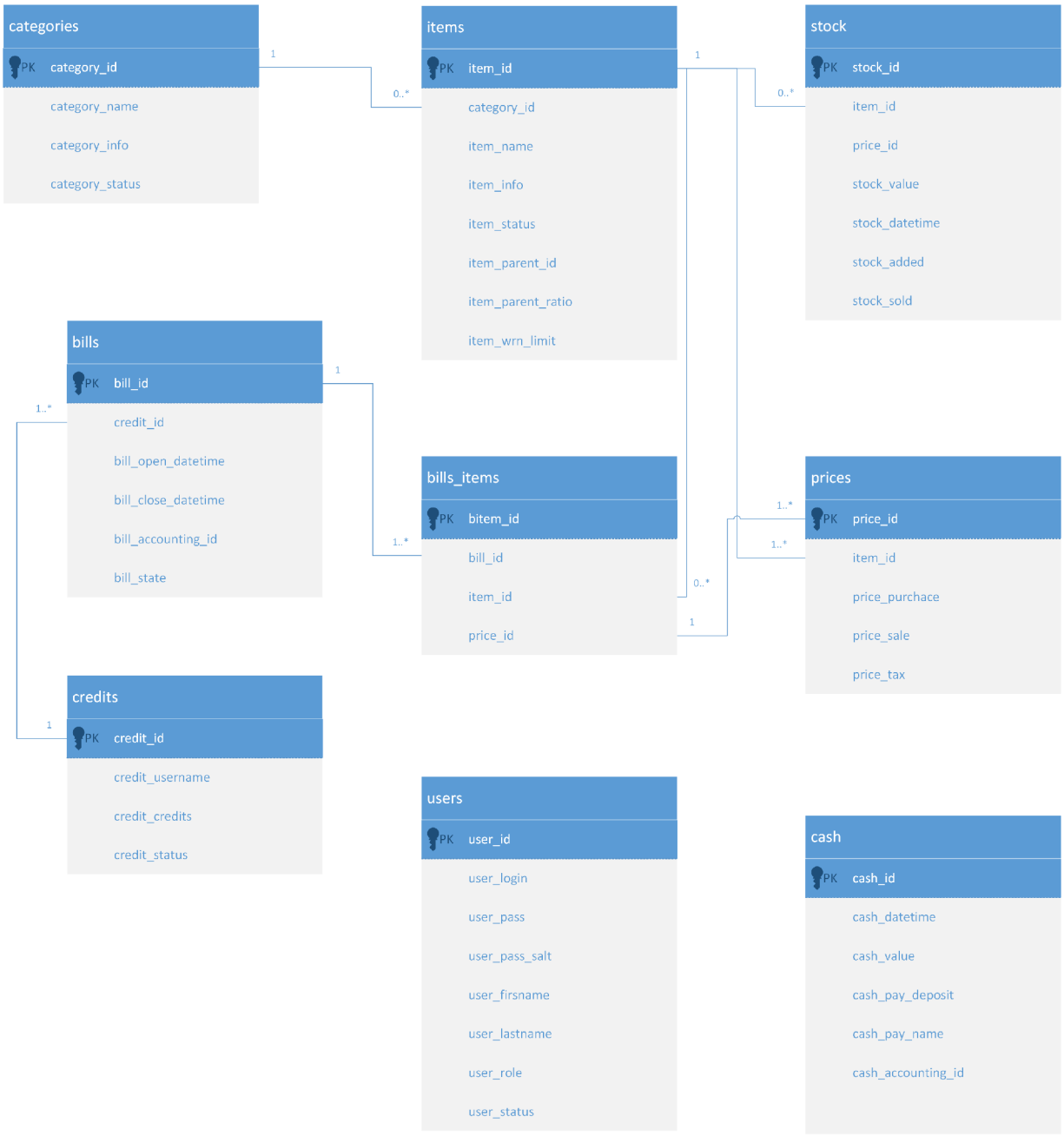
Obsah DVD

Na přiloženém DVD nosiči se nachází:

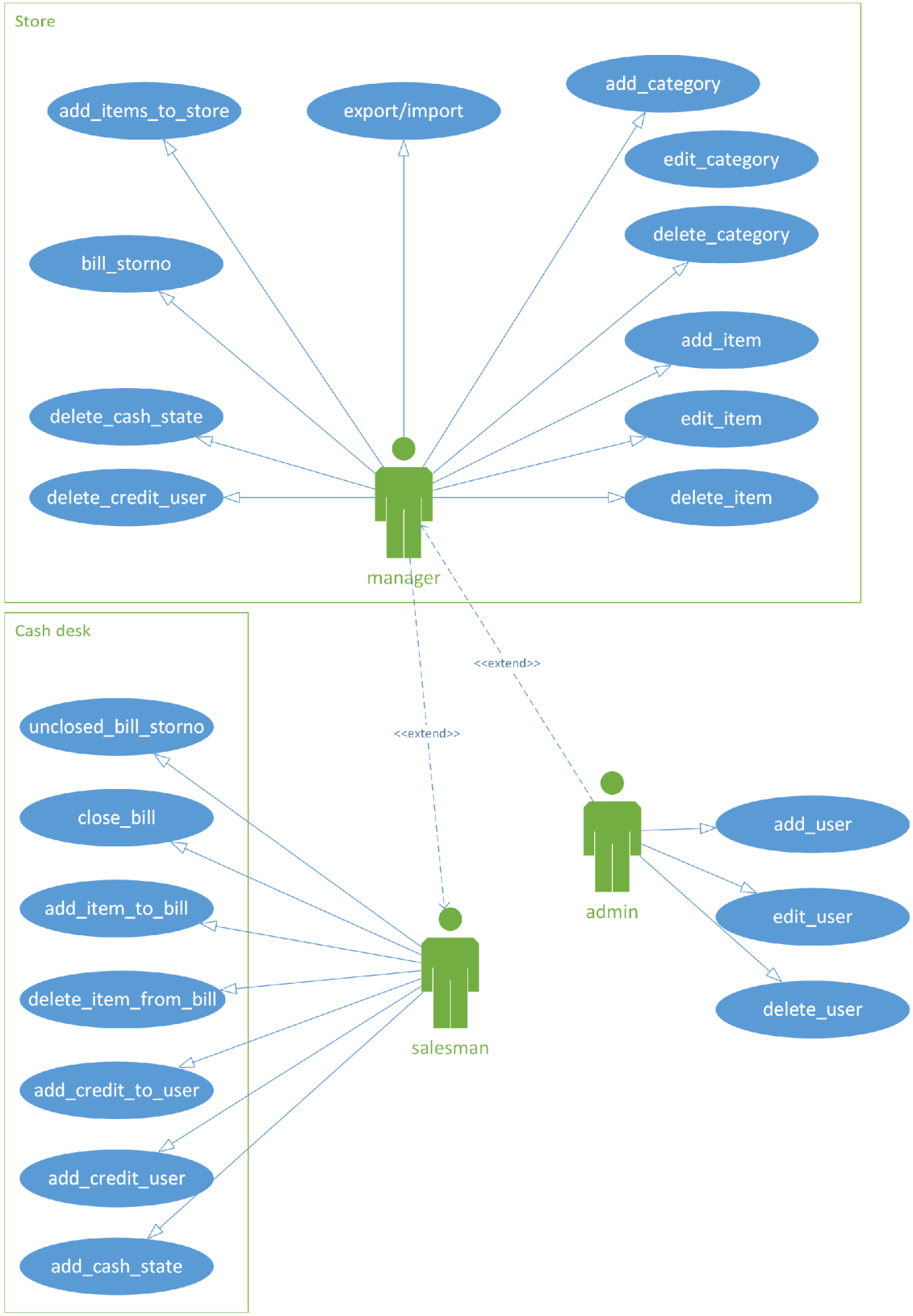
- text bakalářské práce ve formátu PDF
- text bakalářské práce ve formátu LaTeX
- zdrojové kódy aplikace
- script pro vytvoření a inicializaci MySQL tabulek

Příloha B

ER diagram a Use Case diagram



Obrázek B.1: ER diagram



Obrázek B.2: Use Case diagram