TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Mobilní pokladní systém pro elektronický obchod

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Jan Prokop**
Vedoucí práce: Ing. Igor Kopetschke

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

MOBILE CASH SYSTEM FOR E-SHOP

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Jan Prokop**
Supervisor: Ing. Igor Kopetschke

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Prokop**
Osobní číslo:	**M11000113**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Mobilní pokladní systém pro elektronický obchod**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Proveďte rešerši zaměřenou na případné již existující řešení + jejich srovnání
2. Navrhněte vlastní řešení, zdůrazněte rozdíly oproti řešením existujícím
3. Navrhněte komunikaci s externími periferiemi
4. Navrhněte a implementujte úlohu včetně grafického interaktivního rozhraní
5. Proveďte testování při nasazení v praxi
6. Vypracujte vyhodnocení a závěr

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **35 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **RISCHPATER, Ray. Application development with Qt creator a fast-paced guide for building cross-platform applications using Qt and Qt Quick. New Edition. Birmingham, UK: Packt Pub, 2013. ISBN 978-178-3282-319..**
[2] **Qt project [online]. [cit. 2014-10-06]. Dostupné z: http://qt-project.org/.**
[3] **CASTLEDINE, Earle, Myles EFTOS a Max WHEELER. Vytváříme mobilní web a aplikace pro chytré telefony a tablety. 1. vyd. Brno: Computer Press, 2013, 288 s. ISBN 978-80-251-3763-5.**

Vedoucí bakalářské práce: **Ing. Igor Kopetschke**
Ústav nových technologií a aplikované informatiky

Datum zadání bakalářské práce: **20. října 2014**
Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 20. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Abstrakt

Bakalářská práce řeší úlohu vytvoření aplikace pro kamenný obchod, která bude přímo propojená s elektronickým obchodem fungující na platformě Kupshop. Zadaný úkol, který má umožnit rychlé vytvoření a zpracování objednávky v kamenném obchodě, byl řešen ve dvou částech. První částí je webová aplikace, která umožňuje vykonávat všechny požadované operace. Druhou částí je vytvoření desktopové aplikace, která zobrazuje webovou aplikaci, pro rychlou identifikaci produktů využívá čtečku čárových kódu a pro tisk dokladu využívá stolní tiskárnu.

Abstract

The aim of this Bachelor thesis is to programme application for permanent shop, which will be directly linked up with the Kupshop platform based e-shop. The assigned task, which should provide the clients with fast placing and processing of an order in a permanent shop, is divided into two parts. The first part deals with the web application running all required operations. The second part deals with programming of the desktop application, displaying a web application, which for the fast product identification uses a barcode scanner and for invoice printing uses a printer.

Poděkování

Rád bych poděkoval panu Ing. Igoru Kopetschkemu, za rady a pomoc při tvoření této bakalářské práce.

Děkuji také celému kolektivu firmy wpj s.r.o za pomoc a možnosti, které mi poskytli.

Také bych chtěl poděkovat panu doc. RNDr. Pavlu Satrapovi, Ph.D. za vytvoření šablony[5] pro LaTeX která mi v mnohém usnadnila psaní této práce.

V neposlední řadě děkuji své přítelkyni a rodině za podporu během mého studia.

Obsah

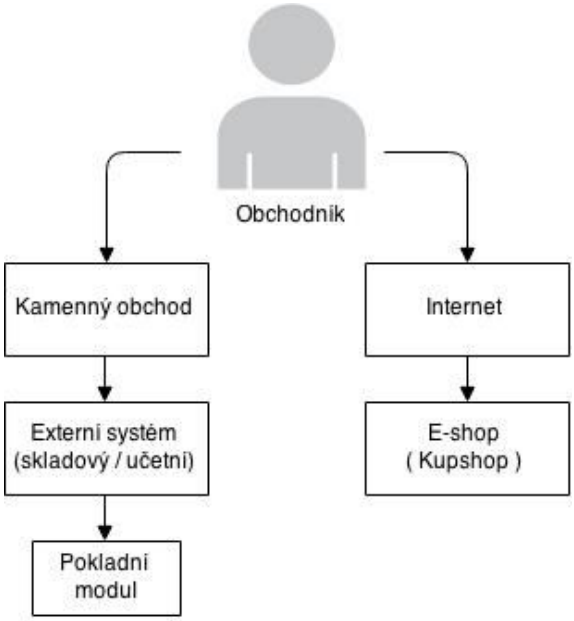
Seznam obrázků

Seznam zkratek

TUL	Technická univerzita v Liberci
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
POS	Point Of Sale
OS	Operační Systém
IO	Input Output
PHP	PHP: Hypertext Preprocessor
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
MVC	Model View Controller
HTTP	Hypertext Transfer Protocol
UML	Unified Modeling Language
ER	Entity Relationship
API	Application Programming Interface
AJAX	Asynchronous JavaScript and XML
RFID	Radio Frequency Identification
QML	Qt Meta Language

1 Úvod

Obchodník, který má zavedený e-shop na platformě Kupshop[10] a zároveň provozuje kamenný obchod, používá pro administrativní účely dva na sobě nezávislé systémy, jak naznačuje obrázek 1.1. Pro správu kamenného obchodu používá některý ze skladových nebo účetních systémů, které budou dále označovány jako „externí systémy“. Pro správu elektronických objednávek z internetu obchodníci využívají administraci e-shopu.



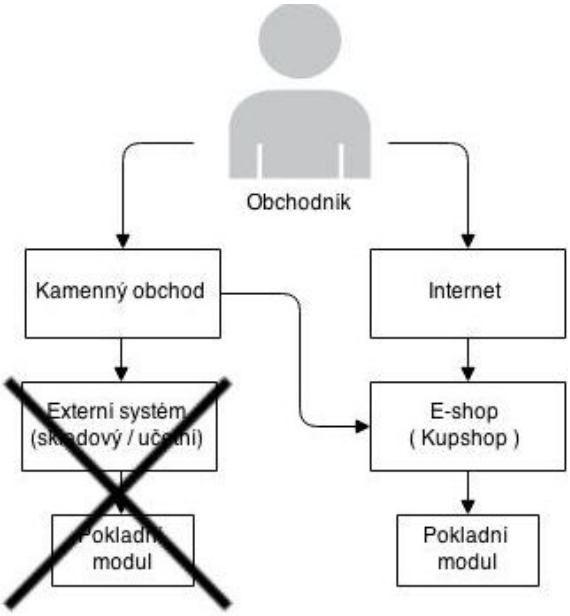
Obrázek 1.1: Aktuální stav

Externí systém obchodníkovi umožňuje vykonávat specifické operace v určitých oblastech obchodu, jako je například práce s produkty, které jsou předmětem prodeje. V rámci těchto operací je možno vytvářet produkty, upravovat je, vytvářet varianty k těmto produktům (například barevné varianty, velikosti, druhy materiálu, atd.), přidávat jim identifikační kódy a případně sdružovat produkty do různých skupin.

Jednou z dalších oblastí externího systému je práce se skladem. V rámci této oblasti je možno například měnit počty kusů k jednotlivým produktům. Ve většině případů je možno pomocí externího systému provádět inventury skladů, což v tomto případě znamená vytisknutí kompletního seznamu produktů a následná fyzická

kontrola těchto produktů ve skladu. Pokud je na prodejnu doručeno nové zboží, je možné ho tzv. „naskladnit“. To v praxi znamená prohlédnout celou fakturu naskladnění, ručně vyhledat jednotlivé položky a přidat jim kusy, které jsou v této naskladňující faktuře. V některých obchodech, jak v kamenných, tak e-shopech, se zboží vůbec neopakuje, jen se vytváří stále nové produkty, které je nutné zavést do systému. Z toho vyplývá požadavek, aby proces přidávání nových produktů byl co nejrychlejší a nejjednodušší.

Většinu těchto procesů musí obchodníci v kamenných obchodech kvůli elektronické verzi obchodu dělat dvakrát. Pokud je na prodejnu doručeno nové zboží k prodeji, musí se naskladnit jak v externím systému, tak zároveň v administraci e-shopu, což znamená dvojnasobné časové nároky na jedno naskladnění. Z důvodu rozdílných principů externích systémů a systému Kupshop je velmi komplikované tyto externí systémy spárovat. V aktuální době je systém Kupshop propojen pouze s externím systémem Pohoda, avšak i toto propojení je jen v rámci synchronizací skladů, viz následující kapitolu. Proto vzniká potřeba výše zmíněné externí systémy odstranit, aby se obchodníci vyhnuli zbytečným procesům a s tím spojenými časovými a finančními náklady na provoz a údržbu obou systémů (obr. 1.2). Aktuálně je již systém Kupshop téměř schopný nahradit skladový a účetní systém, jedním z posledních chybějících prvků je pokladna.



Obrázek 1.2: Cílový stav

Pokud přijde zákazník do kamenného obchodu a chce zakoupit vybrané zboží, obsluha obchodu má dvě možnosti jak zpracovat prodej zboží:

- **Externí skladový systém**
  První možností je vytvořit objednávku v externím systému, který má modul pokladny. Tento modul umožňuje komunikaci se čtečkou čárových kódů, která

je schopná rychle detekovat produkt a není tak třeba ho manuálně vyhledávat a vybírat. Zároveň je nejčastěji připojen k tiskárně malých účtenek.

- **E-shop**
  Druhou možností je vytvořit manuálně objednávku v administraci e-shopu. Zde je nutno vyhledat a vybrat produkty, které chce zákazník koupit, což značně zpomaluje vyřizování objednávky zákazníka. Neexistuje zde žádná možnost opravdu efektivně připojit čtečku čárových kódů, aby fungovala plně automaticky a při sejmutí čárového kódu přidala sejmutý produkt jako položku k objednávce. Taktéž chybí automatizovaná možnost tisku účtenek.

Téměř ve všech případech si obsluha obchodu vybere možnost první, tedy externí systém. Po vyřízení obchodu se zákazníkem práce obsluhy v rámci této objednávky nekončí. Dále je ještě nutné v administraci e-shopu vyhledat produkty které byly prodány, odebrat počty prodaných kusů aby se shodovaly s reálným množstvím produktů na skladě. Z důvodu vzájemného nepropojení systémů vznikají nepříjemnosti. Například, zákazník si na e-shopu objedná poslední kus zboží které je skladem, než ale obsluha obchodu stihne internetovou objednávku zpracovat, prodá zboží fyzicky v kamenném obchodě a tudíž zboží není pro zákazníka e-shopu k dispozici, ačkoliv si ho objednal dříve.

Prodej v kamenném obchodě je srovnatelný s objednávkou vytvořenou pomocí e-shopu, pouze s rozdílem dopravy a platby objednávky. Chybí zde však způsob, jakým by bylo možné během pár okamžiků identifikovat produkt, přidat ho k objednávce a následně celkově vyřídit objednávku. Pokud by takový způsob existoval, bylo by možné opustit externí systémy úplně.

2 Rešerše stávajících řešení

Tato kapitola obsahuje rešerši existujících řešení pokladní systémů, které jsou zkráceně nazývány POS[1]. Moderní POS systémy se v dnešní době nevyužívají jen v prodejnách zboží, ale čím dál častěji také například v restauracích, rychlých občerstveních a dalších stravovacích zařízeních. Například v restauracích a kavárnách je vhodné mít takovou aplikaci na tabletu, kdy je možné aby číšník díky bezdrátovému připojení poslal objednávku kuchaři do kuchyně přímo od stolu zákazníka. Pro účely kamenného obchodu musejí být vyloučeny aplikace, které neumějí skladové operace, což v tomto případě znamená, že nemají žádnou databázi produktů.

V následující části práce je používáno slovo „kasa“. Ačkoliv je toto slovo považováno za neformální výraz, v této bakalářské práci bude použito pro označení fyzického místa pro ukládání finanční hotovosti. Není bohužel možné použít výraz „pokladna“, protože v celé této práci slovo „pokladna“ označuje souhrn zařízení[2], pomocí kterého je možné zpracovat nákup zákazníka.

2.1 Prozkoumané POS systémy

Jako potenciálně vhodná pokladní řešení byly prozkoumány následující systémy:

- **ERPLY[26] (viz obrázek 2.4)**
  · umožňuje běh aplikace i v případě výpadku internetu, data jsou synchronizována ve chvíli, kdy je připojení obnoveno
  · obsahuje skladový systém
  · mnoho připojitelných periférií - čtečky čárových kódů, tiskárny účtenek, čtečky platebních karet
  · možnost upravovat a vytvářet produkty přímo v pokladně
  · statistiky v reálném čase (například majitel obchodu může v reálném čase sledovat průběh zpracování objednávky v kamenném obchodě)
  · je možné spouštět v prohlížeči nebo jako aplikaci pro Android, iOS
  · možnost vytvářet a spravovat slevové kupóny
  · podpora mnoha platebních terminálů
  · možnost propojení s libovolným systémem přes API

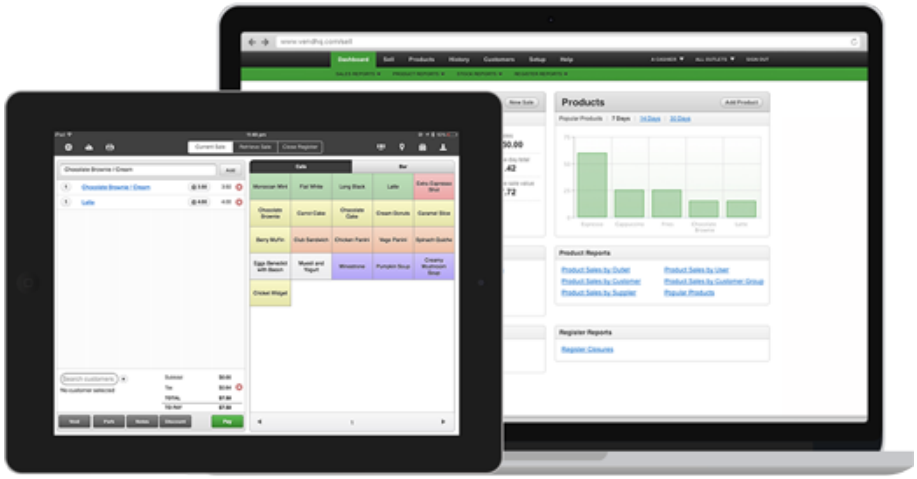
[1] POS je zkratka z anglického Point Of Sale, v překladu „místo prodeje“
[2] stolní počítač, klávesnici, myš, čtečku identifikačních štítků, tiskárnu účtenek, atd.

- **Imonggo [11] viz obrázek 2.1**
  · skladový systém
  · připojitelný hardware
  · vytváření vlastních záloh do souboru
  · umožňuje dynamicky měnit sazby daní
  · statistiky prodejnosti
  · mobilita - v prohlížeči nebo jako aplikaci pro iOS

Obrázek 2.1: Pokladna Imonggo na zařízení iPad (zdroj: [6])

- **Vend [27](viz obrázek 2.2)**
  · jednoduché ovládání
  · vlastní hardwarové zařízení
  · PayPal platby
  · statistiky
  · propojení s dalšímy systémy (účetní systémy, správy zaměstnanců, …)
  · možnost vytvořeních vlastních daňových hladin

- **POHODA Kasa Offline [12](viz obrázek 2.3)**
  · vzdálené připojení k systému POHODA
  · mnoho podporovaných externích periférií
  · přihlašování uživatelů pomocí pinu a čtečky čárového kódu
  · náhledy zboží
  · odložení rozpracovaného nákupu na později - možnost se k němu zpětně vracet
  · nastavení různých práv pro různé uživatele
  · tréninkový režim - žádné provedené operace se nikde neprojeví
  · nutnost vlastnit některou ze základních licencí systému POHODA



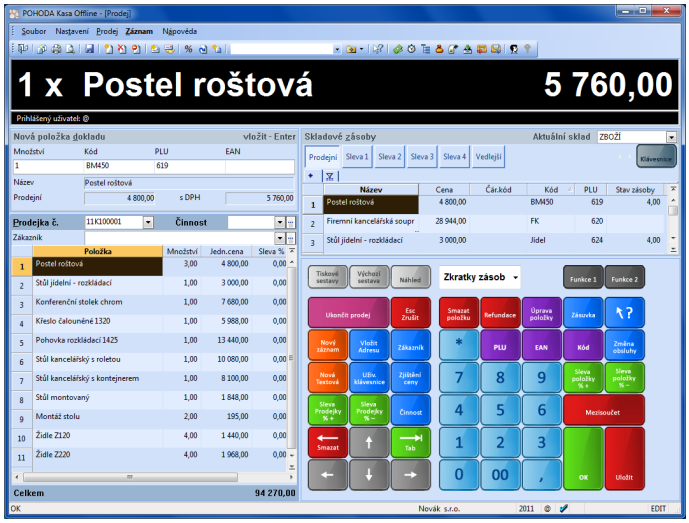
Obrázek 2.2: POS aplikace Vend na iOS (zdroj: [7])

2.2 Přehled aktuálně používaných POS systémů

V rámci bakalářské práce byl proveden přehled mezi čtyřmi majiteli e-shopů, kteří provozují e-shop postavený na platformě Kupshop a zároveň vlastní kamennou prodejnu. Přehled byl zaměřen na POS systémy, které obchodníci využívají, případně čím a jak jej nahrazují, nebyl však zaměřen na jejich externí skladové nebo účetní systémy. Z přehledu vyplývá, že všichni dotázaní obchodníci až na jednoho využívají integrovanou POS aplikaci v rámci jejich zavedeného účetního nebo skladového systému. V přehledu bylo také zjištěno, co jim na POS aplikacích, které používají, vadí, nebo naopak, které funkčnosti jim vyhovují. Bylo také zjištěno jak by případně aplikaci ještě vylepšili.

- **Móda prádlo - aplikace Kasa Offline v systému POHODA (viz obrázek 2.3)**
  Obchod je zaměřen zejména na prodej dámského spodního prádla. Pro účetní a skladové operace používají český systém POHODA. Pro pokladní operace v kamenném obchodě používají aplikaci Kasa Offline, který je doplňkovým modulem systému POHODA. Aplikace je implementována pouze pro operační systém Windows.

  Největší výhodu vidí v propojení kasy s celým účetním systémem. Účetní systém má vlastní databázi produktů, přidání produktu k objednávce tedy funguje rychle, odezva po přečtení čárového kódu čtečkou je necelá sekunda, kdy je produkt identifikován a přiřazen k objednávce. Aplikace také spravuje fyzický stav kasy, na konci každého dne je nutné uzavřít kasu, spočítat peníze a zapsat do aplikace. Tím se dají zjistit případné nesrovnalosti peněz na pokladně. Nejproblémovější se jeví nahlížení do skladů. Po sejmutí čárového kódu je odezva kolem tří sekund, než se objeví seznam produktů. Jako výhodu vidí propojení s čtečkou čárových kódů a tiskárnou štítků.



Obrázek 2.3: Aplikace KASA Offline v systému POHODA (zdroj: [8])

- **PetrBakos - systém ERPLY**
  Tento obchod je zaměřen na prodej originálních designových a módních doplňků. Pro účetní a skladové operace využívají systém ERPLY. Tento obchod pracuje spíše s jednotkami produktů, jak při správě, tak při prodejích, z čehož vyplývá, že zde není kladen důraz na velkou variabilitu a rychlost systému. Pro pokladní operace využívají zařízení iPad s instalovanou pokladní aplikací, která je dokoupitelným modulem systému ERPLY. Jako velkou výhodu uvádí proces, který je nutné každý den vykonat, než se pokladna může používat. Při otevření pokladny se totiž zobrazí dialog, který vyzve obsluhu aby spočítala peníze v kase. Tím se porovná aktuální částka v kase vůči částce na konci předchozího dne. Na konci dne obsluha opět musí uzavřít kasu spočítáním peněz a zadáním do pokladny. Je možné vytisknout denní report pohybů peněz na kase. Jako nevýhodu vidí nutnost měsíčních plateb za provoz tohoto systému.

Obrázek 2.4: POS systém ERPLY na zařízení iPad (zdroj: [9])

- **Centrum Muziky - Microsoft Excel**
  Obchod je zaměřen na prodej hudebních nástrojů a příslušenství. Pro účetní a skladové systémy využívají program Microsoft Excel, ve kterém mají vytvořené tabulky a několik maker. Návštěva tohoto obchodníka ovšem znamenala zajímavý pohled na danou problematiku, protože nebyl ovlivněn žádným dosud používaným pokladním systémem a byly pouze vyjmenovány funkce, které jsou opravdu potřebné již v základu aplikace.

- **Díly na Kotle - Účetnictví Halbrštát**
  Obchod nabízí náhradní díly na zařízení působící v tepelné technice. Pro účetní a skladové operace využívají software od českého výrobce, Účetnictví Halbrštát 2015. V doplňkovém modulu k účetnictví je možnost vytvořit a zpracovat objednávku od zákazníka. Ovšem jakou velkou nevýhodu obchodník uvádí absenci elektronické verze kasy v systému. Při každé zpracované objednávce musí do fyzické knihy zapsat objem přijatých peněz od zákazníka a jaká částka je aktuálně v kase. V systému také neexistuje žádné naskladnění, je nutné ručně vyhledat jednotlivé produkty a následně jim zvětšit počet kusů. Naopak si obchodník chválí velkou stabilitu a spolehlivost celé aplikace. K aplikaci mají připojenou stolní tiskárnu na tisk faktur, malé doklady netisknou. Pro identifikaci předmětů mají připojenou čtečku čárových kódů.

Při použití některého z existujících komerčních POS systémů je nutnost propojit e-shop a aplikaci skrze API[3], kterou ve většině případů tyto systémy nabízejí. To však znamená velkou časovou náročnost a byla by nutnost upravit některé funkce a vlastnosti e-shopu.

Pro příklad lze uvést propojení systému Kupshop s účetním systémem POHODA. V rámci tohoto propojení dochází pouze k synchronizaci skladů, tedy že se synchronizují pouze počty kusů. Pokud se přidá produkt v jednom systému, přidá se jeho „základní verze“[4], ovšem další podrobné informace, včetně obrázků, je opět třeba přidat ručně do obou systémů. Vytvoření takovéto synchronizace skladů zabrala do spolehlivé funkčnosti zhruba 5 „člověkoměsíců“[5]. Vytvoření propojení v rámci produktů, uživatelů a dalších agend by podle odhadů zabralo dalších nejméně 8 „člověkoměsíců“.

V případě volby propojení s některým externím POS systémem není bohužel možné z větší části ovlivnit vzhled a funkčnost vybraného POS systému.

Nabízí se také možnost využití některého z existujících open-source[6] POS systémů. Tato varianta se může zdát nejlepší alternativou, avšak je zde opět nutnost propojit tento systém s e-shopem. Jako výhodou se může zdát to, že open source systém je možné upravovat podle svých potřeb. Ovšem tyto úpravy většinou znamenají nutnost znalostí onoho systému a byly by opět časově velmi náročné.

[3] API je definované rozhraní pro možnost vnější komunikace s aplikací

[4] základní verzí je myšlen pouze název, počet kusů a unikátní identifikátor produktu

[5] člověkoměsíc - počet hodin práce jednoho pracovníka za jeden kalendářní měsíc, vypočítáno počtem pracovních dnů v měsící vynásobených 8 hodinami.

[6] Open source, v překladu otevřený software, je počítačový software s volně přístupným a upravitelným zdrojovým kódem.

Dalším společným negativem výše zmíněných dvou možností realizace je nezačlenění do celkového konceptu e-shopu. Pracovníci v kamenných obchodech často vykonávají pokladní operace a ve volných chvílích provádějí některé skladové či jiné operace přímo v administraci e-shopu. Pro potenciální obsluhu POS systému je tedy velmi důležité, aby celkový vzhled a funkčnost aplikace odpovídala co nejvíce konceptu e-shopu. Proto se jako nejlepší možnost jeví vytvoření vlastní aplikace, která bude celým konceptem co nejvíce odpovídat e-shopu.

3 Požadavky

Při shrnování požadavků na aplikaci je vhodné využít všech informací které byly získány vypracováním rešerše existujících řešení v kamenných obchodech.

3.1 Požadavky na aplikaci

Z rešerše vyplývají následující obecné požadavky na aplikaci:

- **Elektronická podoba fyzické kasy**
  Aby bylo možné kontrolovat stav kasy, zjišťovat ztráty peněz, případně méně časté přebytky peněz, je vhodné, aby aplikace spravovala stav fyzické kasy. Při každé pokladní operaci je zřejmé, jakým směrem a v jakém objemu jde tok peněz (od zákazníka k obsluze, nebo naopak). Pokud tedy máme počáteční stav fyzické kasy na začátku každého dne, je možné na konci dne v elektronické pokladně zobrazit částku, která by správně měla být ve fyzické kase.

- **Výběr a vklad pěněz do kasy**
  Pokud se uskuteční výběr peněz z kasy, například na zaplacení kurýrovi za zboží, nebo za nákup materiálu, je třeba umožňit tyto vklady a výběry pěněz zaznamenat v elektronické podobě.

- **Propojení s externími periferiemi**
  Z důvodu nutnosti rychlé identifikace předmětu je požadováno, aby aplikace byla propojena se čtečkou čárových kódů. Pro tisk pokladních dokladů, případně faktur je třeba, aby byla aplikace propojena také s tiskárnou účtenek nebo klasickou stolní tiskárnou.

- **Možnost nahlížení do skladu**
  Dalším požadavkem je zjednodušená možnost nahlížení do skladu obchodu. Pokud obsluha pokladny potřebuje vědět, zda-li je na skladě daný produkt například jiné barvy, je vhodné, aby pokladna po identifikaci produktu uměla zobrazit kolik kusů je fyzicky na skladě, případně jestli máme jiné varianty produktů.

- **Možnost zpracovat objednávku vytvořenou elektronicky**
  V případě, že zákazník vytvoří objednávku prostředníctvím e-shopu a zvolí si vyzvednutí zboží na kamenné prodejně, je nutné, aby aplikace uměla objednávku vyhledat a zpracovat.

- **Přiřazení uskutečněného nákupu k uživateli e-shopu**
  Pokud zákazník kupuje zboží přímo v kamenném obchodě bez předchozí objednávky na e-shopu, je vhodné mít možnost přiřadit uskutečněný obchod k jeho elektronickému účtu na e-shopu, v případě, že takový účet vlastní. Tato funkce je využívána třeba při věrnostních akcích, kdy zákazníkovi větší počet objednávek dává nějakou výhodu nebo bonus.

- **Možnost aplikování slevového kupónu**
  Pokud je zákazník držitelem některého slevového kupónu, které jsou na e-shopu vytvořeny, je požadavkem, aby bylo možné i v případě nákupu v kamenném obchodě tento slevový kupón aplikovat.

- **Možnost různých druhů plateb**
  Jedním z posledních kroků pokladní operace je výběr způsobu platby. Zde je požadavek na více druhů plateb. Platba v hotovosti, platba kartou, platba tzv. na fakturu(vytiskneme fakturu a zákazník - nejčastěji právnická osoba, zaplatí zboží pozdějším převodem na účet obchodníka).

- **Přidání slevy na celou objednávku**
  Jednoduše umožnit přidat na celou objednávku slevu např. 10%.

- **Denní statistiky**
  Dalším z požadavků je zobrazovat statistiky, jaká byla denní tržba. Dále jaké množství peněz bylo zaplaceno v hotovosti, kartou, fakturou. Ve statistikách by měla být možnost zobrazení denní historie pohybů na pokladně, případně umožňovat vytisknutí tohoto seznamu pro účetní záležitosti.

- **Možnost nastavení různých práv**
  Protože v aplikaci bude možné provádět mnoho různých operací a ne všichni zaměstnanci by měli mít možnost je provádět všechny, je nutností umožnit definování práv jednotlivým uživatelům.

- **Tisk identifikačních štítků**
  Možnost vytisknutí identifikačních štítků na daný produkt přes některou z externích periférií.

Ve výše uvedených požadavcích lze nalézt některé specifické skladové operace. Znamená to tedy, že pokud by se k výše zmíněným požadavkům přidaly další operace specifické pro sklad, mohla by být tato aplikace využívána i ve skladech. Pokud budeme mít možnost v aplikaci určité funkce zakázat pomocí nastavení práv, bylo by možné pro pracovníka ve skladu zakázat pokladní funkce a naopak. Tím se aplikace stává potenciálně ještě využitelnější.

3.2 Způsoby používání

Pokusme se tedy nyní nastínit několik základních způsobů použití této aplikace. Pro která zařízení by byla určena? Ve kterých částech obchodu bude aplikace používána?

3.2.1 Zařízení

Jsou dva předpokládané druhy zařízení, které by aplikace měla podporovat:

- **Stolní zařízení**
  S ohledem na zavedené obchody, které mají z velké části na pokladních místech výkonově dostačující zařízení, je vhodné aplikaci navrhovat tak, aby bylo možné ji spustit i na těchto zařízeních. V kamenných obchodech a skladech je nejrozšířenejším zařízením kancelářský počítač, na kterém běží některá z posledních verzí operačního systému Windows.

- **Přenosné zařízení**
  Aby bylo možné lépe využívat všechny funkce aplikace, je vhodné, aby běžela i na mobilních zařízeních, jako jsou tablety s operačním systémem Android. V dnešní době mají tablety výkon srovnatelný s kancelářským počítačem. Velkou výhodou je ale přenosnost. A v neposlední řadě je výhodou i nižší pořizovací cena, menší spotřeba energie a menší požadavek na prostor, než u stolního počítače.

3.2.2 Místo využití

Jsou dvě předpokládaná místa využití:

- V obchodě

- Ve skladu

Pokud by byla k tabletu připojena podstatně dražší verze bezdrátové čtečky čárových kódů a tiskárny účtenek, bylo by možné pohybovat se s pokladnou po obchodě a zpracovávat tak zákazníkovu objednávku kdekoliv v obchodě. Stejně tak by bylo možné pohybovat se s tabletem po skladu a provádět inventuru efektivnějším způsobem než v případě papírové verze. Ze zřejmých důvodů tedy bude aplikace vyvíjena a přizpůsobena také pro tablety s nejvíce rozšířeným mobilním operačním systémem Android.

4 Návrh

Z provedeného přehledu POS systémů a sběru požadavků vyplývá, že aplikace by měla být spustitelná na stolních počítačích s OS Windows a na tabletech s OS Android. K aplikaci by měly být připojené externí periferie, jako je čtečka čárových kódů, tiskárna účtenek nebo obyčejná stolní tiskárna. Jak již bylo také zmíněno, pokud by aplikace obsahovala některé další doplňující funkce, bylo by možné využívat jí jako plnohodnotnou skladovou aplikaci. Vzhledem k tomu, že aplikace by měla být připojena k elektronickému obchodu který se poměrně rychle vyvíjí, je vhodné, aby aplikace byla snadno upravitelná, bylo možné přidávat do ní jednoduše další funkce, nejlépe bez opětovné nutnosti znovu instalovat novou aplikaci v kamenném obchodě. Jak již bylo v úvodu zmíněno, uskutečněný obchod v kamenném obchodě se dá považovat za obyčejnou objednávku v e-shopu, pouze s několika jinými příznaky, jako je způsob dopravy, nebo způsob platby. Pokud tedy budeme navrhovat aplikaci, je vhodné co nejvíce využívat principy a funkce e-shopu.

4.1 Katalog požadavků

Katalog požadavků zachycuje požadavky tak, aby bylo možné vhodně podle nich navrhnout řešení problému. Katalog obsahuje většinou co nejméně informací o konkrétní implementaci. Funkční požadavky definují jednotlivé služby a funkce, které jsou očekávány. Nefunkční požadavky pak definuje systém jako celek, řeší zde obecné vlastnosti cílové aplikace.

4.1.1 Funkční požadavky

- přihlášení/odhlášení pracovníka z aplikace
- načtení existující objednávky podle identifikačního čísla objednávky
- vytvoření nové objednávky
- přidání/odebrání produktu z objednávky
- změna počtu kusů produktů v objednávce
- přidání/změnění/zrušení slevy na celou objednávku
- aplikování/odstranění existujícího slevového kupónu

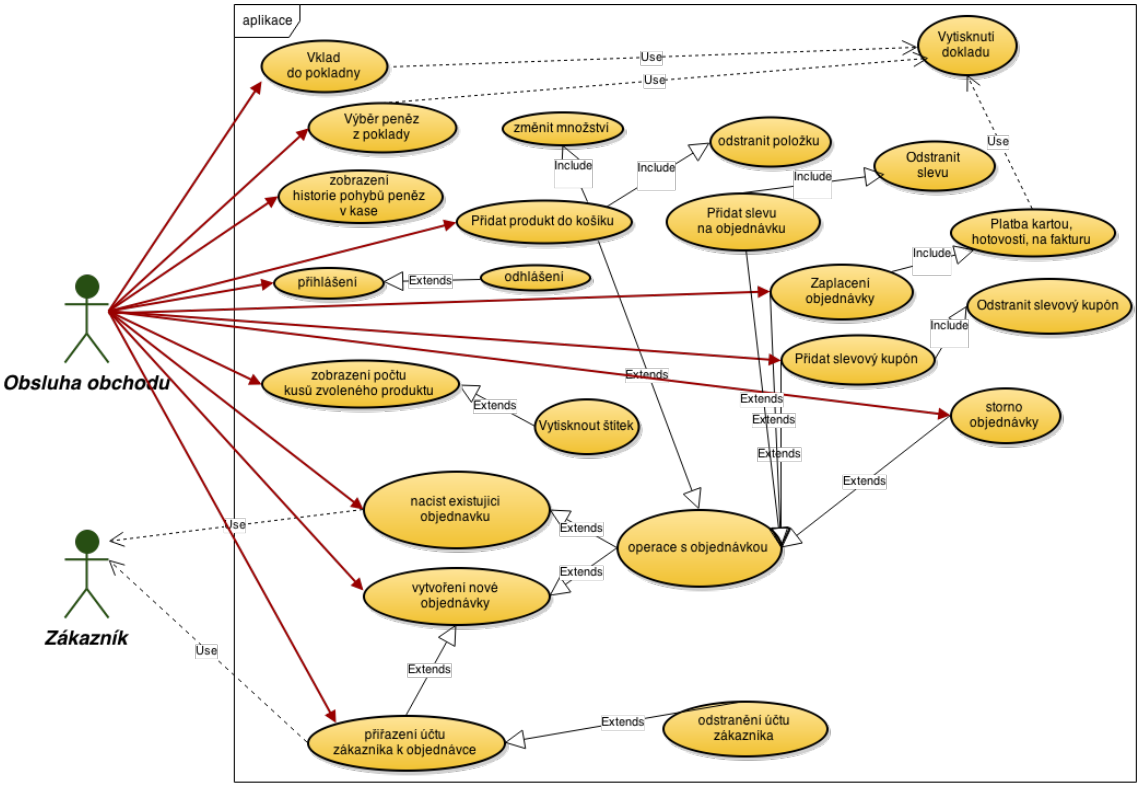
- přiřazení/odstranění zaregistrovaného uživatele e-shopu z objednávky
- zvolení možnosti platby v hotovosti, kartou, na fakturu
- stornování existující objednávky
- vložení/vybrání peněz s možností připsat poznámku
- zobrazení historie pohybů peněz v kase
- zobrazení počtu kusů zvoleného produktu
- možnost tisku identifikačního štítku na zvolený produkt
- tisk faktury

4.1.2 Nefunkční požadavky

- grafické uživatelské prostředí co nejvíce podobné systému Kupshop
- uživatelská aplikace, která bude mít připojené externí periferie
- aplikace by měla být intuitivní a pro její používání by mělo stačit jednoduché zaškolení
- aplikace bude multiplatformní
- provoz na osobním počítači
- provoz na mobilním zařízení
- externí periferie jako je čtečka čárových kódů a tiskárna

4.2 Use case diagram

Use case diagram se používá na zachycení funkčních požadavků. Obsahuje aktéry, use cases (případy užití), hranice systému a relace. Na obr. 4.1 je znázorněn use case diagram aplikace.



Obrázek 4.1: Use-case diagram aplikace

4.3 Základní návrh aplikace

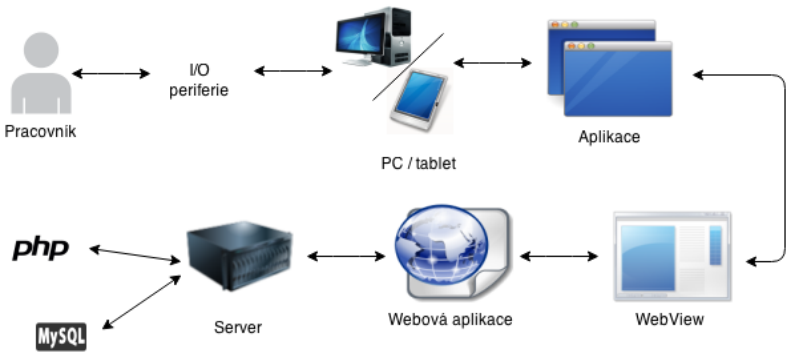
Prvotním rozhodnutím, které ovlivňuje celý návrh aplikace je, zda-li bude aplikace celá na straně uživatele, nebo část aplikace přesuneme na stranu serveru.

- **Celá aplikace na straně uživatele**
  První způsob řešení má velikou výhodu v samostatnosti - není potřeba připojení k internetu. Znamenalo by to ovšem, že by v zařízení, na kterém aplikace poběží, musela být poměrně velká databáze všech produktů, uživatelů a všech částí, které jsou pro zpracování objednávky potřeba. Další nevýhodou je nutnost vytvoření synchronizací mezi pokladnou a e-shopem. Pokud by vypadlo spojení mezi aplikací a serverem, bylo by nutné provést synchronizaci ihned po opětovném připojení. Zde ovšem hrozí kupříkladu stejný typ problému jaký mají majitelé obchodů s externím systémem a e-shopem. Pokud by zákazník zakoupil v kamenném obchodě poslední kus některého z produktů,

tak v případě výpadku spojení aplikace a e-shopu se na e-shopu stále zboží ukazuje jako skladem (které už skladem vlastně není) do doby, než je spojení obnoveno a je provedena synchronizace aplikace a e-shopu.

- **Část aplikace na straně serveru**
  Tento způsob řešení má několik výhod. První výhodou je možnost využití již existujících funkcí e-shopu, které se dají vhodně použít pro vytvoření objednávky a následné zpracování objednávky. Další výhodou je, že není nutná žádná synchronizace dat mezi aplikací a serverem. Nevýhodou tohoto řešení je naopak nutnost neustálého připojení k internetu. V případě výpadku internetu by to znamenalo nefunkčnost celé aplikace. Tato nevýhoda se dá ovšem ošetřit zálohováním jednoho připojení druhým proti výpadku. Nespornou výhodou je také možnost upravovat nebo přidávat nové funkce přímo na serveru, čímž odpadá nutnost po každé úpravě aktualizovat aplikaci na počítači v kamenném obchodě. Tím, že se téměř všechny výpočetní operace převedou na stranu serveru, se také sníží hardwarové požadavky na cílové zařízení.

Pro výše zmíněné výhody je zvolena druhá možnost, čili přenést větší část aplikace na stranu serveru. Aplikace bude provádět jen minimum výpočetních operací. Většinu operací by měl provádět server. Aby bylo možné aplikaci v budoucnu upravovat bez nutnosti znovu instalovat novější verzi, nabízí se řešení naprogramovat jí jako webovou aplikaci a k ní příslušnou uživatelskou aplikaci, která by zobrazovala webovou aplikaci v tzv. webview[1] a starala by se jen o komunikaci s externími periferiemi (viz. obr. 4.2).



Obrázek 4.2: Základní princip navrhované aplikace

4.4 Volba vhodného programovacího jazyka

Pro řešení bakalářské práce budou vytvořeny dvě aplikace. Webová aplikace poběží na straně serveru, uživatelská aplikace pak poběží na straně klienta. Pro každou z nich je třeba určit programovací jazyk, v kterém bude napsána:

[1] Webview - komponenta, která umožňuje zobrazit v aplikaci webový obsah

- **Webová aplikace**
  Systém Kupshop je vytvořen v programovacím jazyce PHP na straně serveru s využitím databáze MySQL a na straně webového prohlížeče jsou použity technologie HTML, CSS, Javascript. Aby bylo možné efektivně využívat všechny části e-shopu, bude pro webovou aplikaci nejvhodnější stejný programovací jazyk, tedy PHP. Pro stranu webového prohlížeče bude využito opět HTML, CSS, Javascript.

- **Aplikace na straně uživatele**
  Aplikace by měla být multiplatformní - zejména pro desktopovou platformu Windows a mobilní platformu Android. Pro vývoj této aplikace se zdá být nejvhodnější použít multiplatformní knihovnu Qt [13]. Jako programovací jazyk bude zvolen QML, pouze pro nutné minimum (komunikace s externími periferiemi) bude použit jazyk C++.

4.5 Postup vytvoření aplikace

Protože bude nutné vytvořit v podstatě dvě různé aplikace, celkový proces vytváření výsledného řešení bude probíhat ve dvou fázích. První fáze je vytvoření webové aplikace, kterou bude schopná běžet v prohlížeči bez připojených periférií. Druhá fáze bude spočívat v naprogramování aplikace, která bude mít ve webview webovou aplikaci z první fáze a bude umět komunikovat s externími periferiemi.

4.6 Návrh webové aplikace

Pro vytváření webové aplikace, která má zapadat do celkového konceptu systému Kupshop budou použity stejné programovací jazyky, tedy PHP na straně serveru, HTML, CSS, Javascript pak na straně uživatele. E-shop jako celek je obrovský, vzájemně provázaný systém. Například přidávání produktu k objednávce není tak jednoduché, jak se na první pohled může zdát. Je třeba při něm například odečíst počet kusů daného produktu, aktualizovat sklad, přepočítat cenu celé objednávky, atd. Proto bude snaha v rámci návrhu a následné implementace využívat co nejvíce existujících funkcí e-shopu.

Při vývoji systému Kupshop firma wpj vytvořila vlastní framework[2] implementovaný v jazyce PHP na principu softwarové architektury MVC[3]. Tento framework bude v rámci návrhu webové aplikace prozkoumán a následně použit pro implementaci.

4.6.1 MVC

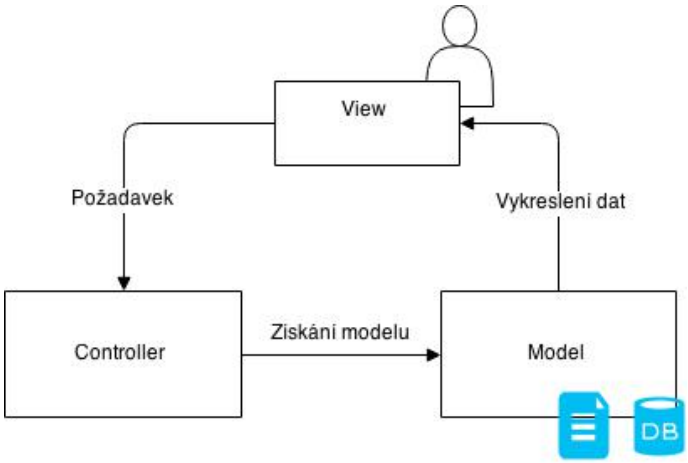
MVC rozděluje aplikaci do tří vrstev (viz. obr. 4.3). Toto oddělení vrstev dává větší kontrolu nad jednotlivými částmi, což umožňuje snadnější vývoj a úpravy.

[2] Framework - knihovna, která rozšiřuje a zjednodušuje práci s daným jazykem.
[3] MVC - Model View Controller, softwarová architektura

- **Model**
  Model, častěji označován jako datový model je funkčním základem celé aplikace. Každá akce uživatele volá některou z funkcí tohoto modelu, která má jasně dané rozhraní. Pomocí těchto funkcí model spravuje svůj vnitřní stav. Může ukládat data do databáze, měnit je, může vytvářet soubory, atd. Důležité je, že model nemusí vědět o existenci view ani kontroleru.

- **View**
  View, neboli pohled, je vrstva aplikace, která má na starosti zobrazení výsledku požadavku od uživatele. Výsledek je nejčastěji uživatelské rozhraní. Pro vytvoření pohledu se obvykle používá některý šablonovací systém.

- **Controller**
  Kontroler je řadič, který na základě zpracování požadavku zavolá příslušný model a následně může požádat view o vykreslení dat.



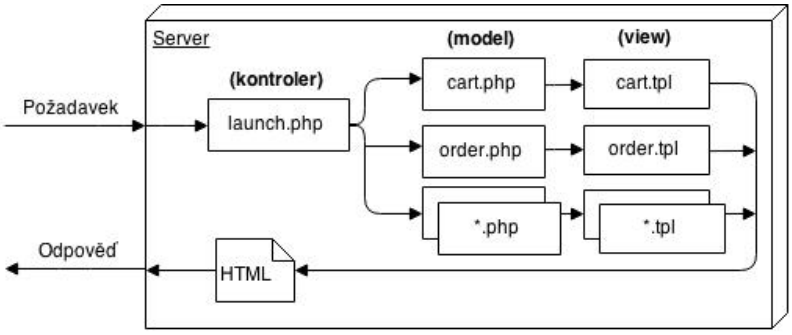
Obrázek 4.3: Zjednodušený diagram MVC

4.6.2 Framework Kupshop

V rámci jedné konzultace ve firmě wpj proběhlo bližší seznámení s frameworkem Kupshop. Framework je použit pro backend[4] elektronického obchodu. Některé jeho části jsou v ojedinělých případech použity i na frontendu.[5] Základní princip frameworku Kupshop je znázorněn na obrázku 4.4.

- **Kontroler**
  Úlohu řadiče, který rozhoduje, jak a čím bude požadavek webového prohlížeče zpracován, je soubor `launch.php`. Tento soubor na základě GET[6] parametru

[4] Backend - část webové aplikace, která slouží ke zpracování dat a administraci aplikace.
[5] Frontend - opak backendu, tedy část aplikace, která stojí na popředí webové aplikace a je viditelná běžným návštěvníkům
[6] GET - dotazovací metoda protokolu HTTP



Obrázek 4.4: Zjednodušený princip frameworku Kupshop podle MVC

rozhoduje, které view se má spustit. Zároveň zastává kontrolu, zda-li je uživatel přihlášen a v případě že není, tak vždy vyzve uživatele k přihlášení.

- **Model**
  Pro vytvoření vlastního modelu je ve frameworku Kupshop v hierarchické struktuře několik podpůrných tříd. Struktury tříd si můžeme prohlédnou pomocí diagramu na obrázku 4.6. Model využívá pro komunikaci s databází vlastní třídu implementovanou pomocí frameworku Doctrine[7]. V rámci modelu zde máme dvě hlavní obecné varianty, tedy seznamy objektů a detail objektu. List se používá na zobrazení seznamu, tedy například seznamu produktů. Pokud chceme editovat některý produkt z listu, je nám zobrazen jeho editační dialog.

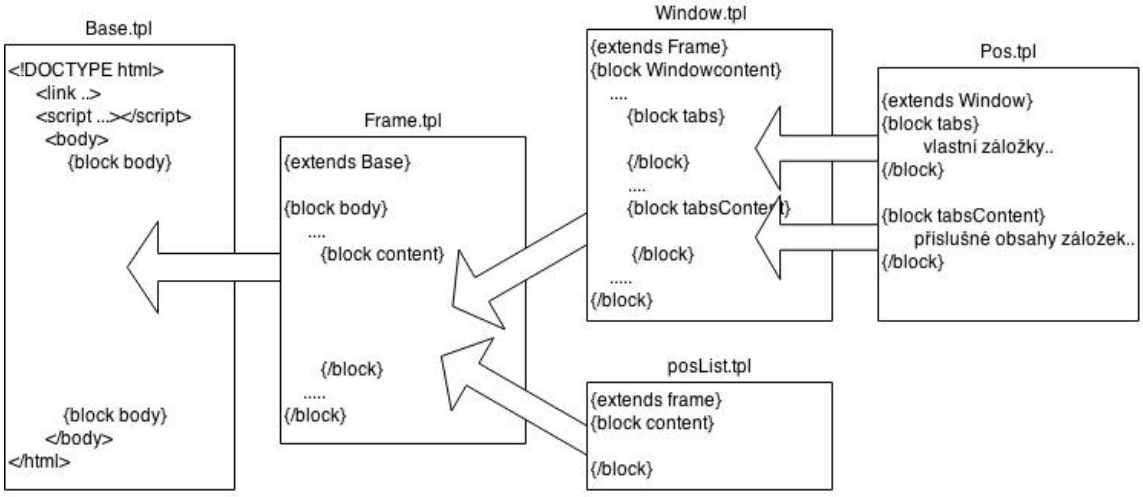
- **View**
  Pro vytvoření view je používán framework Smarty[8]. Tento framework umožňuje načtení souboru s příponou `*.tpl` a vyplnění jeho obsahu daty, které jsme mu předali. Framework Smarty podporuje dědění[9] šablon. Například v rámci párového tagu `<block>` je možné docílit rozdělení částí HTML kódu do samostatných bloků, které je následně možné při dědění šablony předefinovávat, nebo k nim připojovat pomocí parametrů `append` či `prepend` doplňující část kódu. V rámci šablony framework Smarty umožňuje používat všechny HTML tagy ale také dovoluje používat většinu standardních PHP funkcí.

  Pro vytvoření vlastního view je ve frameworku Kupshop připraveno několik view, které tvoří taktéž hierarchickou strukturu (viz obr. 4.5) obdobně jako výše zmíněné modely. Tyto view ze kterých je možné dědit značně zjednodušují vytvoření nového dialogu a zajišťují, že bude mít v základu vložené stejné soubory například pro vzhled dialogu. Pro samotnou grafickou část je použit framework Bootstrap[16], který opět značně zjednodušuje práci a

[7]Doctrine[14] je framework psaný v jazyce PHP určený pro komunikaci s databází

[8]Smarty[15] - šablonovací systém pro PHP, který umožňuje oddělit PHP kód od zobrazovací HTML logiky.

[9]Dědičnost - organizace souborů do stromové struktury, kdy děděním z nadřazené třídy přebíráme její schopnosti, ke kterým můžeme přidat schopnosti vlastní.



Obrázek 4.5: Princip děděných šablon pro tvorbu dialogů a listů

sjednocuje vzhled všech prvků(rámečků, vstupních polí, tlačítek) napříč celou administrací e-shopu.

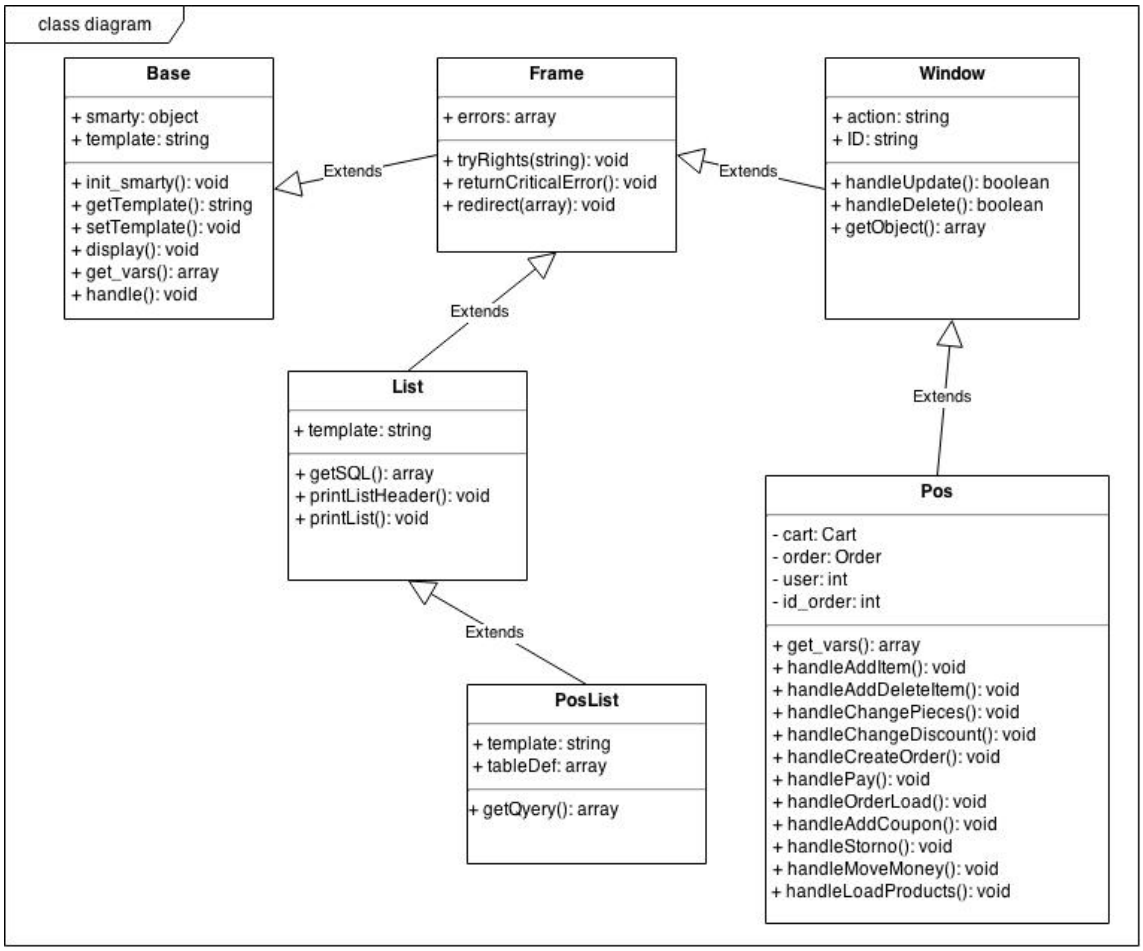
4.6.3 Návrh vlastního modelu

Dalším krokem po seznámení se s frameworkem Kupshop je návrh vlastní třídy nazvané `Pos`. Třída bude mít metody které budou vykonávat všechny funkční požadavky, které jsou sepsány v katalogu požadavků. V rámci vlastního modelu je také třeba vytvořit vlastní tabulku do databáze MySQL, do které bude model ukládat jednotlivé provedené transakce pokladny a následně vytvořit model pro list, který bude provedené transakce zobrazovat v seznamu.

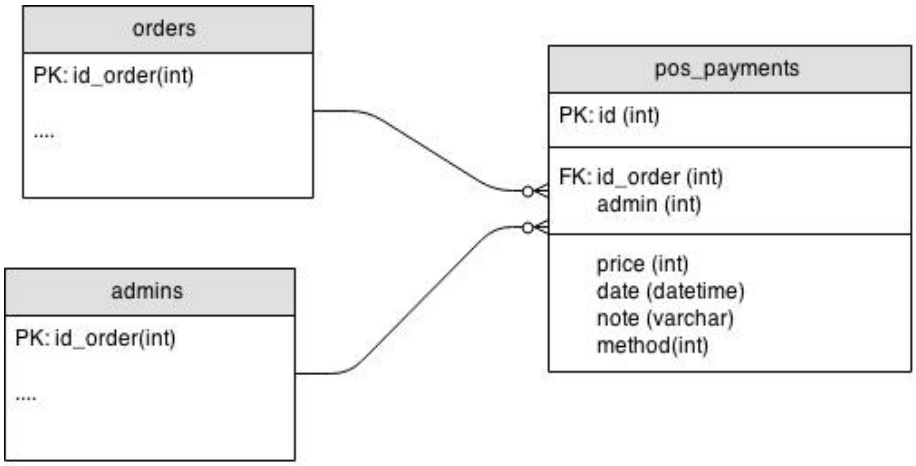
Navržené třídy společně s metodami můžeme vidět na obrázku 4.6. Názvy metod třídy `Pos` mají vždy na začátku vložené slovo „handle“ z podstatného důvodu. V rámci frameworku Kupshop je implementována funkce, kdy podle obsahu GET parametru `acn`, tedy například `addItem`, objekt Window zavolá funkci `handleAddItem`, pokud taková funkce existuje.

V rámci návrhu vlastního modelu byla také navržena tabulka do databáze MySQL, viz ER diagram[10] 4.7.

[10]ER diagram vytváří ucelený obrázek o tom, jaké tabulky budou v databázi. V tomto případě obsahuje i konkrétní datové typy a integritní omezení.



Obrázek 4.6: Class diagram



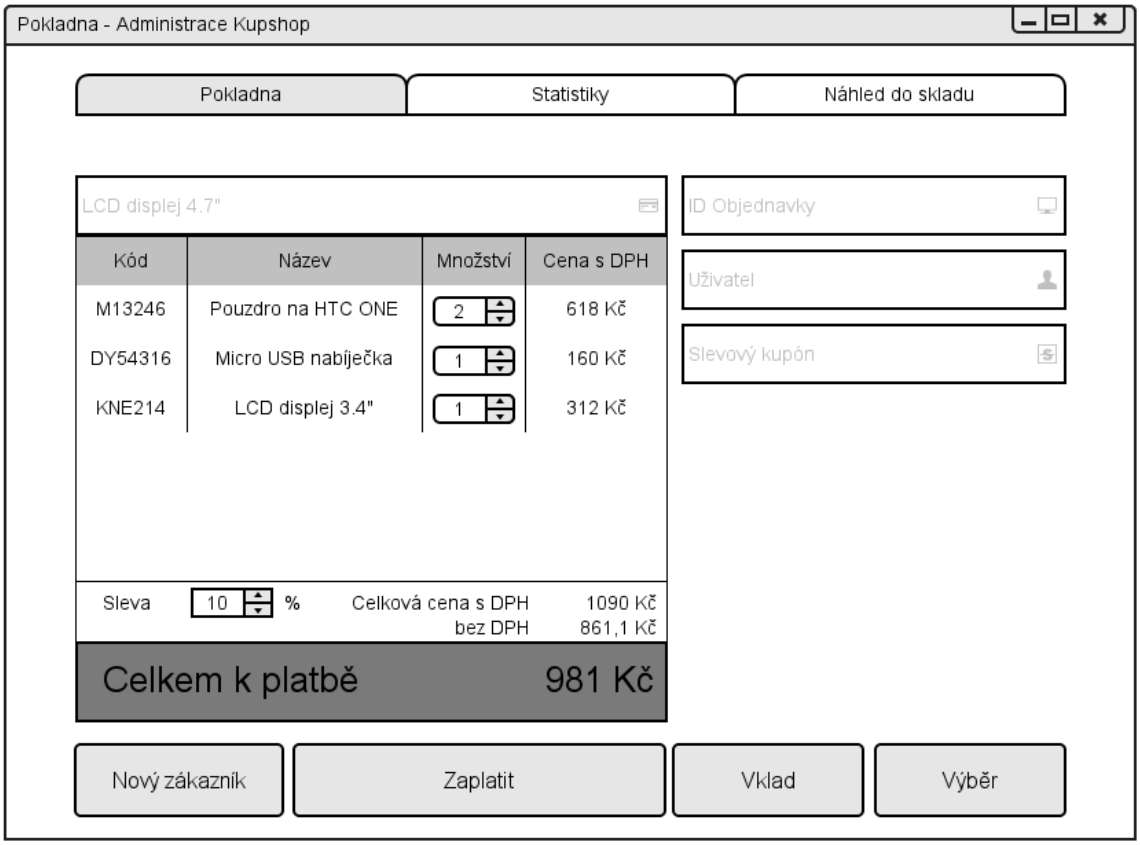
Obrázek 4.7: ER diagram propojení s tabulkami e-shopu

4.6.4 Grafický návrh

Grafický návrh aplikace je jedna z částí, kterou uživatel nejvíce vnímá. V tomto případě je však grafický návrh velmi omezen použitím frameworku Kupshop. V rámci frameworku je totiž používán jednotný vzhled pro celý dialog pomocí frameworku Bootstrap.

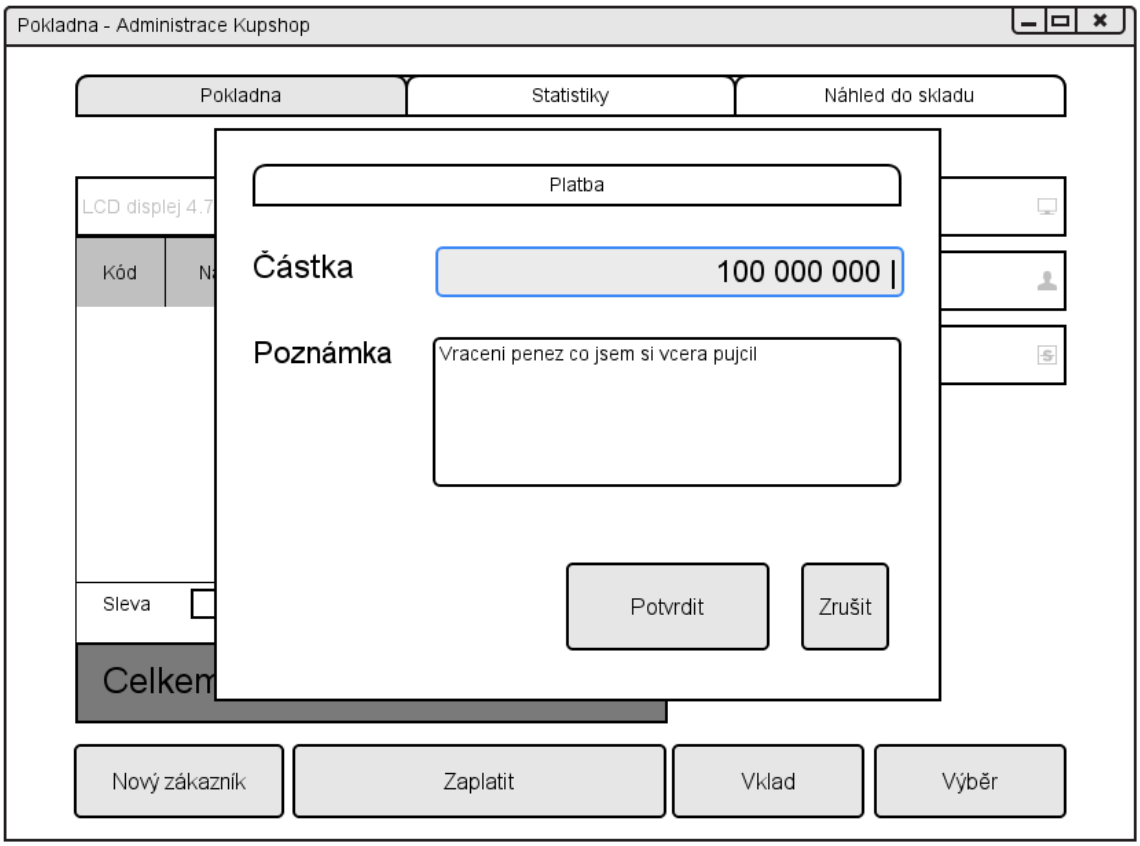
Z toho vyplývá, že není třeba dělat grafický návrh ve smyslu vzhledu tlačítek, případně barevných kombinací, ale půjde především o umístění jednotlivých prvků ve webové aplikaci. Ačkoliv se to může zdát na první pohled jako věc méně důležitá, je umístění prvků ve webové aplikaci velmi podstatné a je nutné je více než dobře rozmyslet.

Pro vytvoření wireframu[11] byl firmou wpj poskytnut přístup do placené sekce online softwaru Moqups [25]. V prvním kroku byl navrhnut základní dialog pro pokladnu. Po schválení hlavního dialogu (viz obrázek 4.8) pokladny firmou wpj byly vytvořeny některé menší dialogy (např. vzhled dialogu pro vklad peněz, viz obrázek 4.9), další záložky pro náhled do skladu a záložka statistik.



Obrázek 4.8: Hlavní dialog webové aplikace

[11] wireframe - náhled, který umožňuje zobrazit a předvést vzhled aplikace bez nutnosti programovat funkčnost



Obrázek 4.9: Dialog pro vklad peněz do pokladny

4.7 Návrh desktopové aplikace

Základní vlastností desktopové aplikace bude komunikace s externími periferiemi. Aplikace by měla být navržena tak, aby bylo možné co nejjednodušším způsobem přidat podporu dalších čteček identifikačních kódů, případně tiskáren dokladů.

4.7.1 Externí periférie

V rámci návrhu desktopové aplikace je nutné definovat periferie, které bude aplikace podporovat. Při výběru externích periférií je nutné uvažovat dva druhy zařízení, na kterých bude aplikace používána - stolní počítač a tablet.

- **Čtečka identifikačních štítků**
  Pro rychlou identifikaci produktu bude k aplikaci vhodné připojit čtečku čárových kódů. K identifikaci je možné použít některou z moderních technologií, jako je RFID[12] nebo QR kódy a další. Z důvodu již existujících čárových kódů na produktech je zvoleno použití standardních čteček čárových kódů. Firma wpj s.r.o zakoupila čtečku čárových kódů MJ-2808A.

[12]RFID - identifikátor fungující na principu bezkontaktní elektronické komunikaci na krátkou vzdálenost

- **Tiskárna**
  V e-shopu nyní neexistuje možnost jakým by bylo možné vytisknout malý daňový doklad - účtenku, je zde možnost tisknout pouze klasickou fakturu ve formátu A4. Každý obchodník často vlastní nějakou tiskárnu, je tedy vhodné, aby aplikace mohla tisknout pomocí jakékoliv tiskárny. V aplikaci by tedy měla být možnost, jaké zařízení pro tisk bude použito. Dále by měla být navržena tak, aby bylo možné co nejsnadněji implementovat tisk malých daňových dokladů.

Do externích periférií bychom taktéž mohli zahrnout klávesnici a myš. Avšak předpokládáme, že u každého stolního počítače jsou takovéto periferie k dispozici. U tabletů takovéto periferie nemusí být, protože disponuje dotykovou obrazovkou a je možné na ní zobrazit virtuální klávesnici. Pro efektivní práci na pokladním místě však v případě tabletu bude doporučováno připojit klávesnici.

4.7.2 Qt

Jak již bylo výše zmíněno, aplikace bude programována s využitím knihovny Qt ve vývojovém prostředí Qt Creator. Aplikace se bude skládat z jednoho okna - dialogu, který bude naprogramován v jazyce QML. Pomocí tohoto jazyka je možné definovat vzhled celé aplikace, přidávat funkčnost jednotlivým tlačítkům, atd. Komunikace s externími periferiemi pak bude programována v jazyce C++. Jak již bylo zmíněno, desktopová aplikace bude zobrazovat webovou aplikace ve webview.

Jsou zde dvě možnosti jak zobrazit webový obsah:

- **Webová aplikace se třemi záložkami v jednom webview**
  Toto řešení má výhodu v jednoduchosti implementace, vše by totiž bylo zobrazené přímo z webové aplikace. Jako nevýhoda se však jeví obtížná implementace rozlišení, v které záložce se uživatel aktuálně nachází. Bylo by nutné provést parsování HTML kódu webview a hledat příznaky pro aktivní záložku. Další nevýhodou je fakt, že při obnovení jedné záložky se společně s ní obnoví všechny ostatní, což může například pracovníkovi zapříčinit ztrátu rozdělané práce v ostatních záložkách.

- **Tři webview, kde každé bude zobrazovat jednu záložku webové aplikace**
  Toto řešení má oproti předchozí možnosti složitější implementaci. Je zde nutné vytvořit logiku na přepínání webview. Výhodou je však jednoduché rozlišení, ve které záložce se pracovník aktuálně nachází. Taktéž je výhodou obnovení jedné záložky, která však neobnoví ostatní.

V rámci konzultaci ve firmě byla rozhodnuta druhá možnost. Desktopová aplikace by měla obsahovat tři webview, které budou obsahovat jednotlivé záložky webové aplikace. V horní části aplikace budou tři tlačítka, které budou zastávat funkce přepínače mezi záložkami. Na jednotlivých webview je možné volat javascriptové funkce které se nachází na zobrazovaných webových stránkách. To bude využito pro přidávání předmětů nebo pro vyhledání produktu ve skladu.

Z důvodu, že veškerá funkčnost je na straně serveru a desktopová aplikace zastává pouze úlohu prostředníka mezi externími periferiemi a webovou aplikací, není zde žádné bezpečností riziko a není tedy nutné vytvářet přihlášení do desktopové aplikace. Je zde tedy pouze nutnost přihlásit se do webové aplikace. Tuto možnost lze realizovat dvěma způsoby.

- **Přihlašovací dialog desktopové aplikace**
  První možností je vytvořit v desktopové aplikaci dialog pro přihlášení, který by odeslal na server požadavek o přihlášení a odpověďí by mu bylo kupříkladu, zda-li byl uživatel úspěšně přihlášen, či ne. V případě použití této metody bychom museli ručně pracovat s cookies webview, abychom zrealizovali přihlášení do webové aplikace.

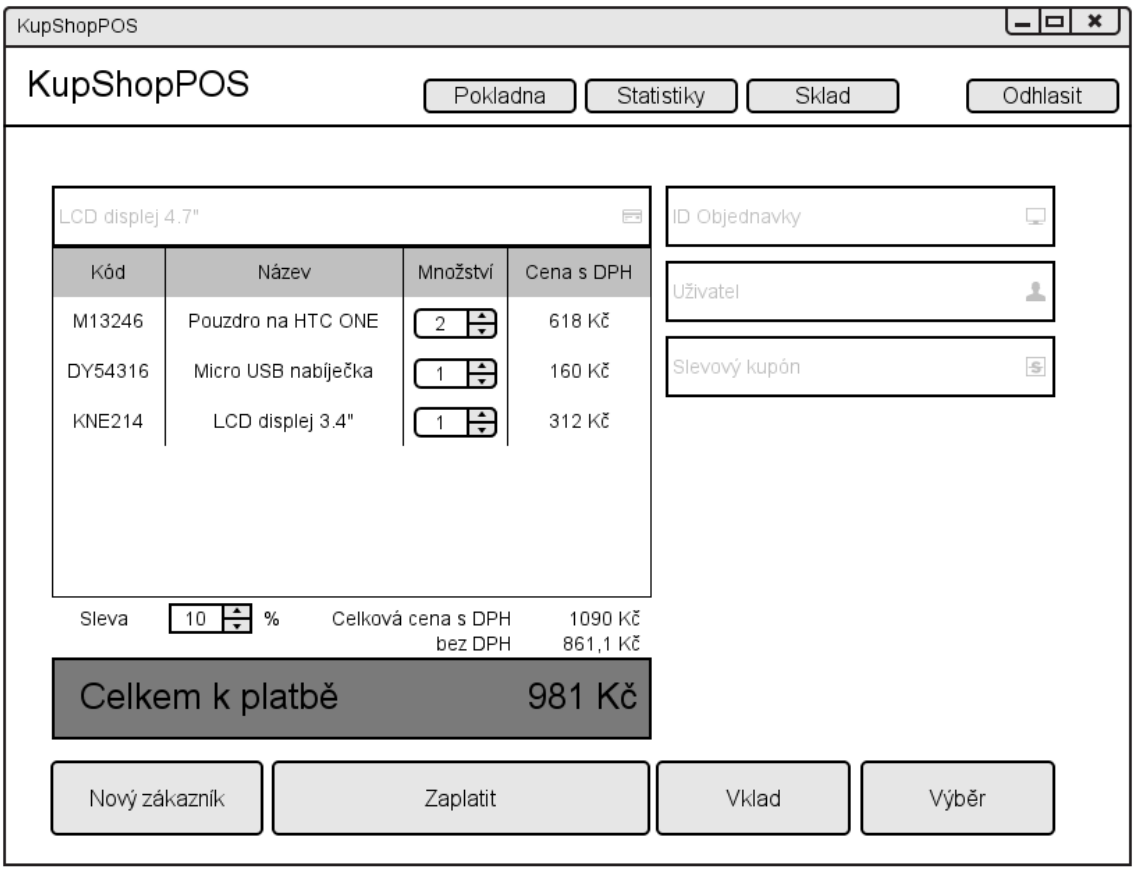
- **Přihlášení pomocí webview**
  V rámci této možnosti by přihlášení probíhalo přímo ve webview, které by zobrazovalo úvodní přihlašovací stránku administrace e-shopu. Tato možnost se jeví z bezpečnostního rizika jako nejlepší a zároveň je nejjednodušší na implementaci.

Přihlášení bude tedy probíhat na základě druhé možnosti, tedy přihlášením pomocí webview, aby se o celý proces přihlášení postarala webová aplikace společně s webview.

4.7.3 Grafický návrh

Na obrázku 4.10 si můžeme prohlédnout grafický návrh desktopové aplikace v softwaru Moqups.

Aplikace bude obsahovat pouze jedno okno. Toto okno v sobě bude obsahovat tři webview, z kterých bude vždy viditelné pouze jedno. K přepínání jednotlivých webview budou sloužit tři tlačítka umístěné v horní části aplikace. Vedle tlačítek pro přepínání záložek bude umístěno tlačítko pro odhlášení.



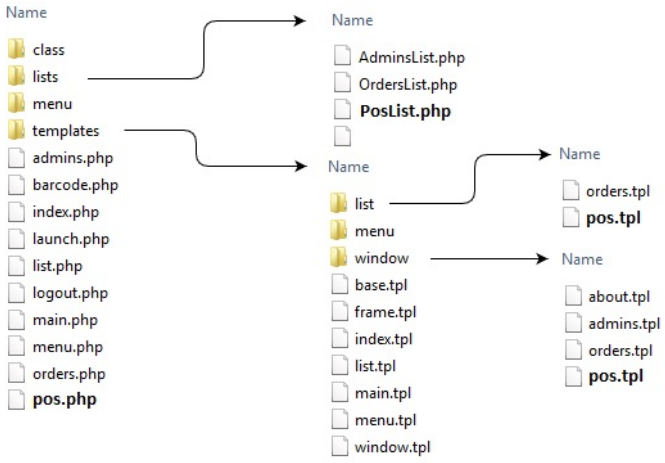
Obrázek 4.10: Grafický návrh aplikace v softwaru Moqups

5 Realizace

Cílem této kapitoly je podat podrobnější informace a ukázky některých „zajímavých“ částí aplikace. Postupně bude na příkladech ukazováno, jak byly některé problémy řešeny a jaké techniky byly při jejich řešení použity. V rámci této kapitoly bude také popsáno několik „neúspěšných“ cest, které byly slepými uličkami vývoje aplikace.

5.1 Webová aplikace

První částí vývoje bylo vytvoření webové aplikace podle návrhu, který byl vytvořen v předchozí kapitole. Vývoj webové aplikace lze rozdělit do dvou částí. První částí je vytvoření dialogu pokladny, pro který je třeba vytvořit model, tedy třídu pomocí jazyka php a zároveň k němu vytvořit view, tedy šablonu s příponou `tpl`. Druhou částí je vytvoření seznamu, tzv. „listu“, který bude zobrazovat ve webové aplikaci transakce provedené na pokladně.



Obrázek 5.1: Adresářová struktura frameworku Kupshop

Pro vytvoření dialogu a seznamu je nutné vytvořit příslušné soubory s třídami a šablonami na správné místo, aby je framework Kupshop mohl načíst a zpracovat. Adresářovou strukturu si je možné prohlédnout na obrázku 5.1. Soubory s tučně psaným názvem jsou soubory, které jsem vytvořil pro běh aplikace.

5.1.1 Dialog

Jak již bylo v předchozí kapitole „Návrh“ zmiňováno, framework Kupshop obsahuje několik podpůrných tříd, ze které lze dědit. V našem případě objektu Pos dědím ze třídy *Window*, která umožňuje základní vytvoření dialogu bez nutnosti v potomkovi cokoliv definovat.

```
<?
$main_class = 'pos';
class Pos extends Window{}
?>
```

Po pouhém definování třídy je již možné díky frameworku Kupshop zobrazení dialogu pomocí webového prohlížeče. HTML soubor, který prohlížeč získá má již hlavičku, všechny potřebné CSS a Javascriptové soubory pro správnou funkčnost všech komponent. Dokument je zatím prázdný a nemá žádný obsah. Vytvořením šablony, tedy souboru `pos.tpl` ve složce `Window`, můžeme tomuto dialogu vytvořit obsah. Následující ukázka kódu zobrazuje základní strukturu šablony, která je třeba pro vytvoření dialogu se třemi záložkami.

```
{extends file="../window.tpl"}

{block title}
    Pokladna
{/block}

{block tabs}
    {windowTab id='flapPOS' label="Pokladna"}
    {windowTab id='flapStat' label="Statistiky"}
    {windowTab id='flapStock' label="Náhled do skladu"}
{/block}
{block tabsContent}
  <div id="flapPOS" class="tab-pane fade boxStatic">
      .. HTML obsah 1. záložky "Pokladna"
  </div>

  <div id="flapStat" class="tab-pane fade boxStatic">
      .. HTML obsah 2. záložky "Statistiky"
  </div>

  <div id="flapStock" class="tab-pane fade boxStatic">
      ...
  </div>
{/block}
```

V první řádku ukázky vidíme dědičnost této šablony od šablony `window.tpl`. Blok `tabs` umožňuje pomocí implementované funkce `windowTab` frameworku

Kupshop vložit do dialogu záložku, kde parametrem `id` definujeme identifikátor HTML bloku `<div>`, který je následně přiřazen jako obsah dané záložky. Parametrem `label` definujeme název této záložky. Ke každé záložce je následně v bloku `tabsContent` přiřazen příslušný `<div>` blok, ve kterém je HTML kód obsahu dané záložky.

První verzí dialogu byl formulář, ve kterém bylo možné přidat položky a další prvky k objednávce. Pro přidávání produktu jsem vytvořil vstupní pole na principu „autocomplete“[1]. Dialog se po každém přidání produktu odeslal na server, kde proběhly všechny požadované operace, například se přidal produkt k objednávce a následně se do webového prohlížeče zobrazil dialog s přidanými produkty a možností přidat další produkt. Během vývoje této první verze jsem postupně zjišťoval, že tento způsob řešení je chybný. Obnovení celého dialogu při každé provedené operaci totiž znamenalo velký čekací čas mezi operacemi na pokladně.

Po konzultaci ve firmě mi byl následně navrhnut způsob řešení pomocí framworku jQuery na straně uživatele. Bohužel jsem do té doby nikdy framework jQuery nepoužíval, tedy prvním krokem pro mě bylo naučit[2] se základní způsoby používání.

V následujícím kroku jsem tedy vytvořil další verzi dialogu. Princip spočíval v přidávání produktu pomocí metody „autocomplete“[3] a následném zpracování pomocí javascriptu. Při akci „přidání produktu“ jsem pomocí javascriptu zduplikoval první řádek tabulky produktů[4], zviditelnil ho[18], což tedy přiřadilo další produkt k objednávce. (případně zvětšil množství u již přidaného produktu). Následně pomocí několik dalších funkcí jsem přepočítal cenu celého nákupu. Na konci, pokud byl nákup dokončen, se teprve dialog se všemi přidanými produkty odesílal na server. Ovšem postupným přidáváním dalších funkcí, jako odebírání produktu[19], zvětšování množství, atd. se čim dál více komplikoval kód. Toto řešení bylo velmi rychlé, čas mezi prováděními operacemi se mnohonásobně snížil. Ovšem v některých případech vznikaly nutné duplicity kódu. Dalším problémem byl fakt, že pokud uživatel v průběhu vytváření objednávky ztratil připojení k internetu nebo ručně obnovil stránku, všechna data byla ztracena a bylo nutno celý proces přidávání produktů opakovat. Z těchto důvodů jsem toto řešení odložil a označil ho za nevhodné.

Následovala další konzultace, po které jsem vytvořil vlastní funkci pomocí jazyka Javascript, která se spustila při stisknutí tlačítka „přidání produktu“. Použitím jQuery funkce `getJSON`[20], která využívá technologie AJAX[5], moje funkce odesílala požadavek na server, který zavolal objekt `Pos` a jako odpověď odeslal celý zabalený objekt objednávky. Ten jsem díky několik dalším javascriptových funkcí rozložil a obnovil jednotlivé položky v dialogu jako je celková cena objednávky, množství produktů v objednávce, případně přidal nový řádek s produktem. Bohužel čím

[1] Tato funkcionalita je implementována v rámci frameworku Kupshop, jde o „našeptávání“ při psaní slova.

[2] V tom mi velmi pomohl blog[17] studenta Tomáše Jadrného, který vysvětluje základy použití frameworku jQuery.

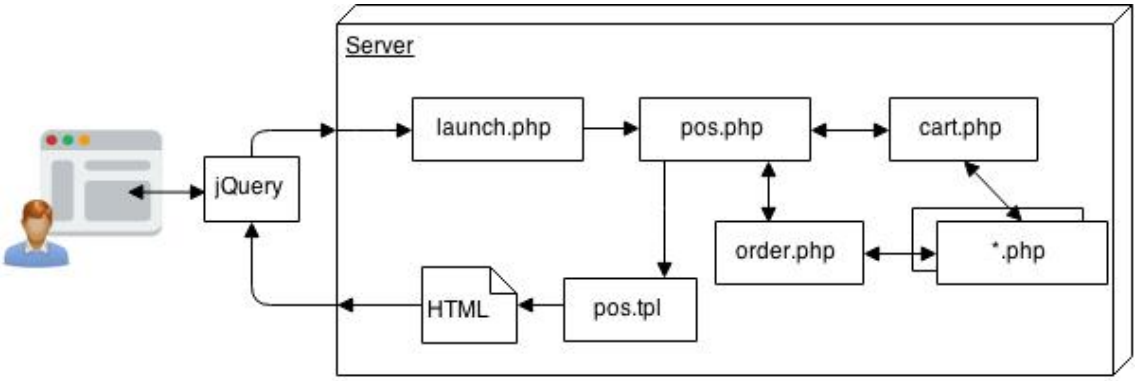
[3] v rámci této metody je možné získat i cenu, EAN a další informace o produktu

[4] tento řádek byl skrytý a neobsahoval žádné informace - byla to „šablona“ řádku

[5] Balík technologií, který umožňuje měnit obsah stránky bez nutnosti ji znovu celou načítat

více funkcí přibývalo, tím víc se kód stával nepřehledný a vzájemně se proplétal, nepříznivě ovlivňoval. Opět jsem tedy narazil na slepou uličku vývoje aplikace.

Po bližším prozkoumání dokumentace k frameworku jQuery jsem se rozhodl, že nejvýhodnější bude použít asynchronní funkci `load`, která umožňuje odeslat požadavek s parametry. Server jako odpověď vrací v tomto případě HTML kód. Celý proces je znázorněn na diagramu 5.2.



Obrázek 5.2: Diagram průběhu akce od uživatele

V rámci pokladny je možné vytvářet novou objednávku nebo zpracovávat již vytvořenou objednávku pomocí e-shopu. Pro vytváření nových objednávek jsem využil třídu `Cart`, která je v rámci e-shopu použita pro funkce košíku. Pro zpracování již existujících objednávek jsem použil třídu Order, která reprezentuje objednávku.

V třídě `Pos` jsem implementoval funkci `handleAddItem`, která je určena pro zpracování požadavku přidání produktu. Jelikož ale předem nevíme, zda-li budeme přidávat produkt k nové objednávce z kamenného obchodu, nebo budeme přidávat produkt k již vytvořené objednávce, jsou ve funkci implementovány oba případy. Níže najdete ukázku funkce třídy `handleAddItem`.

```
function handleAddItem(){
  $id_product = getVal("id_product");
  $id_variation = getVal("id_variation");
  $this->id_order = getVal("id_order");
  if (!empty($id_product)){
    if (empty($this->id_order)) {
      $this->createCart();
      $res = $this->cart->addItem(['id_product' => $id_product,
              'id_variation' => $id_variation, 'pieces' => 1 ]);
    }else{
      $this->order = new Order($this->id_order);
      $res = $this->order->insertItem($id_product, $id_variation, 1);
    }
   $this->err_str = ($res == true)? "Produkt přidán." : "Chyba: ".$res ;
  }
}
```

Funkce `getVal` je obecná funkce frameworku Kupshop, pomocí které je možnost získat obsah proměnných GET nebo POST. Ze strany prohlížeče se pak na funkci `handleAddItem` dotážeme pomocí jQuery funkce, jak je vidět v následující ukázce.

```
function addProduct(id_product, id_variation){
    $('#items').load('launch.php?s=pos.php&acn=addItem #items',
    { id_product: id_product,
    id_variation: id_variation,
    id_order: getOrderID(),
    user_id:  getUserID() },
    function(response, status) {
                if (status == "success") {
                    printRecapitulation(response);
                }
    });
}
```

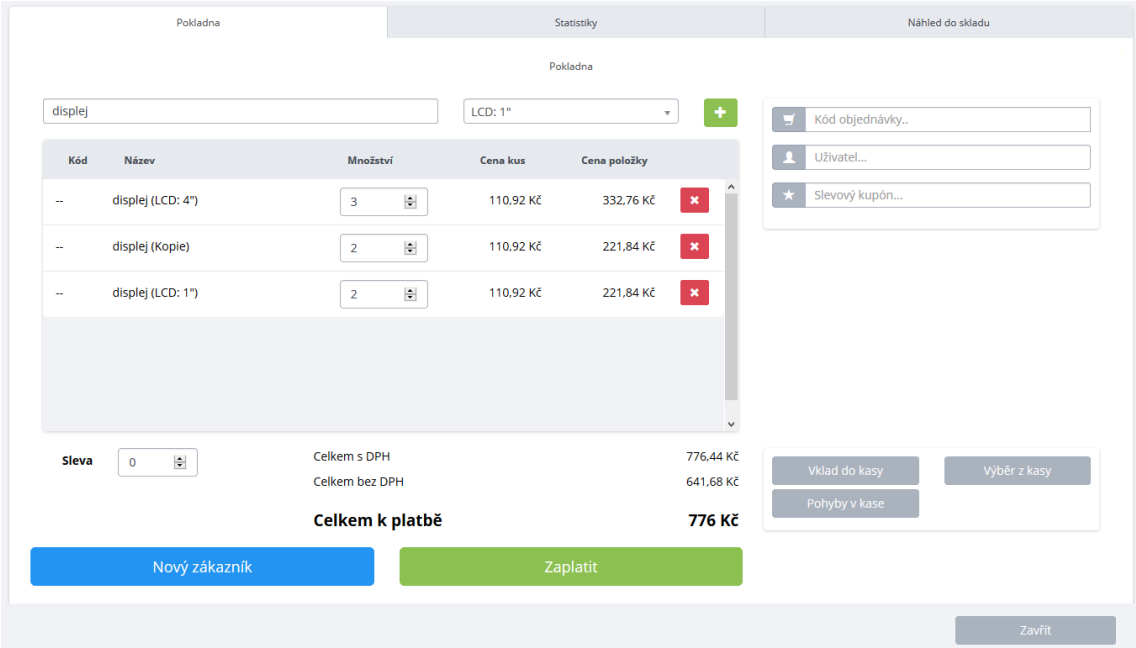
Ve výše uvedené ukázce si lze všimnout GET parametru `acn=addItem` v adrese, kterým docílíme zavolání konkrétní funkce třídy `Pos`. Pokud se požadavek na serveru zpracuje a úspěšně získáme odpověď, provede se v HTML stránce obnova bloku[21] s identifikátorem `items` (který obsahuje celý blok seznamu produktů) a následně se zavolá funkce `printRecapitulation` která obnoví na stránce doplňkové informace o celkové ceně objednávky s daní, bez daně, atd.

V následujících krocích jsem implementoval postupně všechny funkce podle návrhu. Všechny mají obdobnou strukturu jako výše zmíněná funkce `handleAddItem`, tedy že má podmínku, zda-li jde o objednávku, nebo o košík. Stejně tak paralelně s tím byla vždy k dané PHP funkci implementována i javascriptová funkce, která se na ní dotazuje ze strany webového prohlížeče. Například načtení již existující objednávky probíhá stejným způsobem, tedy dotazem na funkci `handleOrderLoad` a při získané odpověďi se pouze nahradí blok s produkty, případně některé další informace, jako je například jméno uživatele který objednávku vytvořil.

Na následujícím obrázku si můžeme prohlédnout výsledný vzhled první a zároveň hlavní záložky pokladny.

Dalším krokem bylo přidat funkčnost tlačítkům „Vklad do kasy“, „Výběr z kasy“ a „Pohyby v kase“. V první kroce jsem vytvořil tabulku v databázi pro ukládání dat z těchto operací. V následující ukázce můžeme vidět databázový dotaz pomocí kterého jsem tabulku vytvořil.

```
CREATE TABLE IF NOT EXISTS `pos_payments` (
    `id` int(11) NOT NULL,
    `id_order` int(11) unsigned DEFAULT NULL,
    `price` double NOT NULL,
    `date` datetime NOT NULL,
    `note` varchar(100) DEFAULT NULL,
    `admin` int(11) DEFAULT NULL,
    `method` int(11) NOT NULL
 )
```



Obrázek 5.3: Vzhled hlavního dialogu pokladny

Pro realizace těchto „poddialogů“ jsem se rozhodl využít knihovnu jQuery UI [22]. V následující ukázce javascriptového kódu je vidět, jakým způsobem se takový dialog definuje. V dolní části ukázky je pak příklad události, která se vyvolá při stisknutí tlačítka „Vložit peníze“.

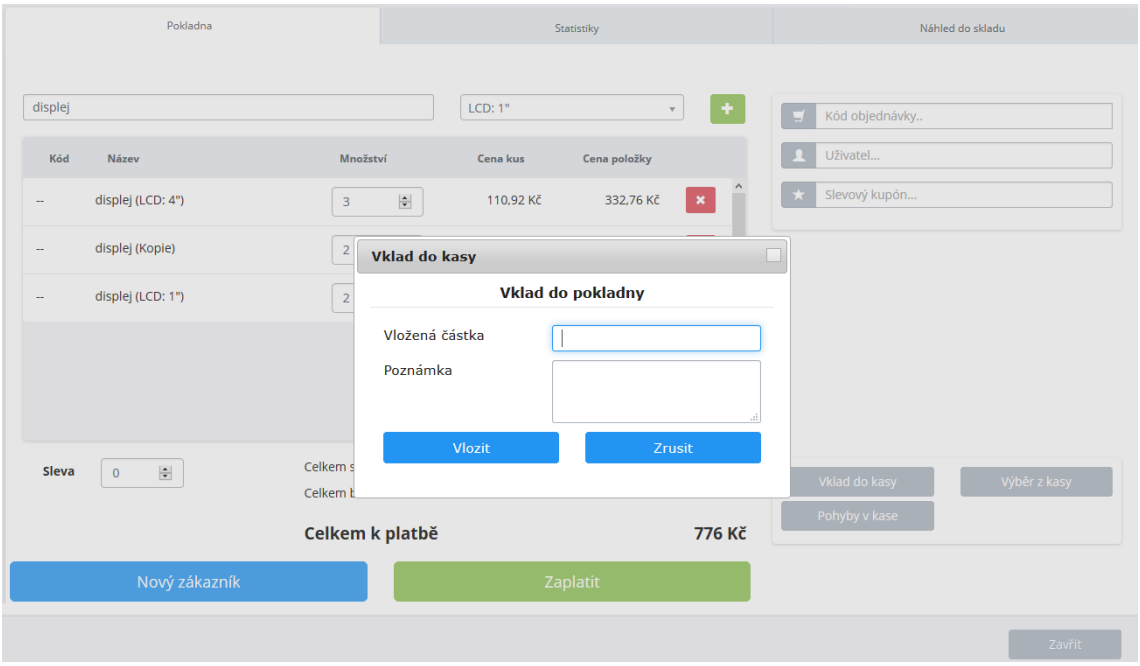
```
$( "#dialog_insert_money" ).dialog(
            { autoOpen: false,
                width: 500,
                height: 300,
                title: "Vklad do kasy",
                resizable: false,
                modal: true,
                draggable: false
            }
    );

 $(document).on('click', '.insert-money', function(){
    $("#dialog_insert_money").dialog("open");
 });
```

Dále je třeba mít v HTML stránce blok, nejčastěji tag `<div>` obsahující kód který chceme zobrazit ve vyskakovacím okénku. V následující ukázce můžete vidět jak vypadá část HTML kódu, který se zobrazí do vyskakovacího okénka a níže obrázek 5.4, jak vypadá výsledný dialog.

```
<div id="dialog_insert_money" title="Vklad do pokladny">
     <div class="row" style="margin-bottom: 10px;">
```

```
            <div class="col-md-12 text-center">
                <h1 class="h4 main-panel-title">Vklad do pokladny</h1>
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-5 control-label">
                Vložená částka
            </div>
            <div class="col-md-7">
                <input type="text" style="margin-bottom: 10px;"
                class="form-control input-sm" id="inserted_money">
            </div>
        </div>
        ....
    </div>
```



Obrázek 5.4: Vyskakovací dialog pro vklad peněz

Tento kód je obdobný pro všechny vyskakovací okénka. Tento typ vyskakovacího okénka byl také použit pro tlačítko „Zaplatit“, které po stisknutí vytvoří objednávku a je možné jí v okénku zaplatit. Následně po zaplacení vyskočí další okénko se zobrazenou fakturou, kterou je možné vytisknout. Všechny tyto dialogy mají obdobný kód. Jediná výraznější změna je v okénku tlačítka „Pohyby v kase“ a zobrazení faktury, které zobrazují v tagu „iframe“ seznam pohybů a fakturu, proto je nutné je před zobrazením aktualizovat[23].

V rámci druhé záložky nazvané „Statistiky“, jsem implementoval funkci do třídy `Pos` která vygeneruje tyto statistiky, vrátí HTML kód celé stránky, ze které jQuery

opět vybere část HTML které nahradí. Samozřejmě by bylo možné statistiky zobrazit jen při načtení dialogu, ovšem to by již pro provedení první operace pokladny nebyly statistiky aktuální.

Do třetí záložky jsem vytvořil jednoduché nahlédnutí do skladu. Je zde vstupní pole, do kterého je možné zadat název produktu. Toto pole funguje opět na principu „autocomplete“. Pokud do něj vložíme slovo a potvrdíme, je nám zobrazen seznam všech variant zadaného produktu.

Pro potvrzování vykonané akce jsem následně přidal do třídy `Pos` globální proměnnou `err_str`, do které se ukládá odpověď na operaci, tedy např. „Produkt byl přidán k objednávce“, „Produkt byl odebrán z košíku“, atd. Tato odpověď se vypisuje uživateli aplikace v horní části dialogu při jakékoliv provedené akci. Protože v rámci implementace jednotlivých funkcí vznikly i funkce pomocné, výsledný kód se zvětšil a stával se nepřehledným. Do budoucna se také počítá s rozšiřováním funkcí jednotlivých záložek, čímž by se kód ještě zvětšoval. Rozhodl jsem se tedy třídu rozdělit podle jednotlivých záložek do tří samostatných tříd a k nim příslušných šablon. V rámci odchytávání neočekávaných chyb jsem následně volání všech funkcí sjednotil do jedné. Funguje jako rozcestník, z webové stránky se již volá pouze `acn=byAction` s příslušným parametrem, kterou akci provádíme. V následující ukázce můžeme tuto funkci vidět.

```
function handlebyAction(){
   $acn = ucfirst(getVal('action'));
   try{
      switch($acn){
         case "AddItem": $this->handleAddItem(); break;
         case "DeleteItem": $this->handleDeleteItem(); break;
         case "ChangePieces": $this->handleChangePieces(); break;
          ....
         case "AddCoupon": $this->handleAddCoupon(); break;
         case "ClearOrder": $this->handleClearOrder(); break;
         case "Storno": $this->handleStorno(); break;
         case "MoveMoney": $this->handleMoveMoney(); break;
         default: return;
      }
   }catch(Exception $e){
      $this->err_str = $e->getMessage();
   }
}
```

5.1.2 List

Pro zobrazení seznamu dat z databáze má framework Kupshop opět řešení, které jsem využil. Následující ukázka zobrazuje implementaci souboru `posList.php` podle návrhu.

```
class PosList extends AdminList{
protected $template = "list/pos.tpl";

protected $tableDef = [
   'id' => '',
   'fields' => [
      'ID' => ['field' => 'id', 'size' => 1],
      'Objednávka' => ['field'=>'id_order', 'size' => 1],
      'Částka' => ['field'=>'price', 'size' => 1.5],
      'Typ platby' => ['field'=>'method', 'size' => 0.7],
      'Popis' => ['field'=>'note', 'size' => 5],
      'Zpracoval' => ['field'=>'admin', 'size' => 1],
   ],
];

function getQuery(){
   $var['header'] = 'Pohyby v kase';
   $var['fields'] = ' pos.id, pos.note, pos.price,
                  a.login as admin, pos.method, pos.id_order ';
   $var['from'] = " ".getTableName("order_payments")." as pos
                   LEFT JOIN ".getTableName("admins")." AS a
                   ON pos.admin=a.id";
   return $var;
}
```

Pro zobrazení listu je potřeba frameworku Kupshop definovat jak má tento list zobrazit. V proměnné `TableDef['fields']` jsou vypsány všechny sloupce, které se mají zobrazit ve výsledném seznamu. Jednotlivé položky v sobě obsahují proměnnou `field`, která udává příslušný sloupec z databáze a proměnná `size` udává šířku jednotlivých sloupců seznamu. Funkce `getQuery` zde slouží pro definování databázového dotazu. Framework po získání pole `var` automaticky složí databázový dotaz z jednotlivých proměnných.

5.2 Desktopová aplikace

Druhou částí celého vývoje bylo vytvoření desktopové aplikace dle návrhu. Nejdříve jsem vytvořil základní kostru aplikace. Pak následovalo vytvoření třídy pro komunikaci s čtečkou čárových kódů a propojení s příslušnými webview.

5.2.1 Webview

V prvním kroce jsem vytvořil základní kostru aplikace pomocí jazyka QML, resp. s využitím interaktivního grafického prostředí Qt Creatoru. V následující ukázce můžeme vidět definici hlavní záložky pokladny.

```
WebEngineView {
      id: pos
      anchors.top: parent.top
      anchors.topMargin: 58
      anchors.right: parent.right
      anchors.left: parent.left
      anchors.bottom: parent.bottom
      visible: true
      url: "http://www."+shop_url+"/admin/index.php?
            url=http%3A//www."+shop_url+"/admin/
            launch.php%3Fs%3Dpos.pos.php"
      onLoadingChanged: reloadOthers(loadRequest);
 }
```

V ukázce si můžeme také všimnout vlastnosti `url`, ve které je definována webová adresa stránky, kterou má webview zobrazit. Tato adresa obsahuje GET parametr `url`. Při přihlášení do webové administrace e-shopu existuje možnost nastavit GET parametr `url`, přičemž v případě úspěšného přihlášení do administrace jsme přesměrování na danou adresu nacházející se v GET parametru `url`. Pro přihlášení do webové aplikace tedy bude sloužit první záložka webview, v které je jako parametr `url` nastavena webová stránka s pokladnou. Po úspěšném přihlášení tím docílíme přesměrování do webové aplikace s pokladnou.

Při spuštění aplikace zobrazuje pouze hlavní webview, v kterém je zobrazena přihlašovací stránka. V předchozí ukázce si také můžeme všimnout vlastnosti `onLoadingChanged`, která udává dění v případě, že probíhá načítání stránky. Tato vlastnost se volá vždy na začátku a při dokončení načtení webového obsahu. Z důvodu detekce přihlašování jsem vytvořil vlastní funkci `reloadOthers`, která detekuje, zda-li byl uživatel úspěšně přihlášen. Na následující ukázce můžeme vidět, jak tato funkce vypadá.

```
function reloadOthers(loadRequest){
   var url = loadRequest.url.toString();
   if (!logged && url.indexOf("index.php") < 0
               && url.indexOf("logout.php") < 0){
      pos_button.visible = true;
      stats_button.visible = true;
      store_button.visible = true;
      logout_button.visible = true;

      store.url =
          "http://www."+shop_url+"/admin/launch.php?s=pos.store.php";
      stats.url =
          "http://www."+shop_url+"/admin/launch.php?s=pos.stats.php";

      logged = true;
   }
}
```

Jak můžeme v předchozí ukázce vidět, úspěšné přihlášení je možné detekovat jednoduchou kontrolou změny adresy. Následně při úspěšném přihlášení jsou zobrazeny tlačítka pro přepínání mezi záložkami a také dojde k přenačtení ostatních záložek. V následující ukázce si můžeme prohlédnout funkci která je volána při stisknutí tlačítka „Odhlášení“.

```
function logout(){
        pos_button.visible = false;
        stats_button.visible = false;
        store_button.visible = false;
        logout_button.visible = false;
        logged = false;
        store.visible = false;
        stats.visible = false;
        pos.visible = true;

        pos.runJavaScript("logout(3)");
}
```

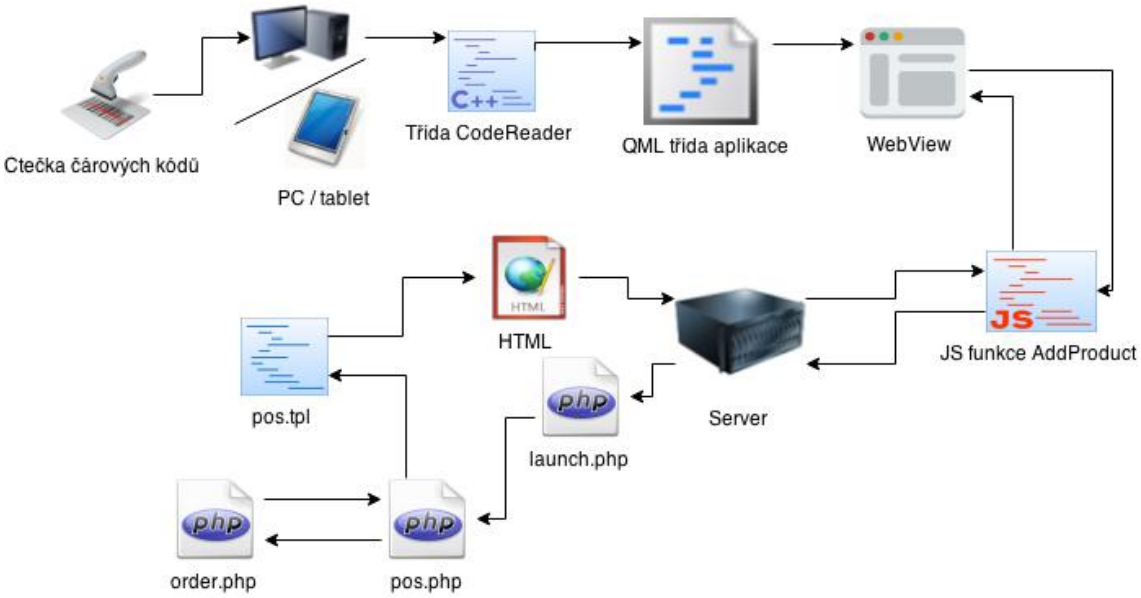
V ukázce můžeme vidět zavolání javascriptové funkce `logout(3)` na webview `pos`, která s tímto parametrem zajístí na serveru smazání `session` s přihlášením a následně provede přesměrování na přihlašovací stránku. Přihlašovací stránku však doplní o „zpětný“ GET parametr `url`, aby po dalším přihlášení proběhlo opětovné přesměrování do webové stránky s pokladnou.

5.2.2 Externí periferie

Pro komunikaci se čtečkou čárových kódů MJ-2808A bylo předpokládáno využití vlastnosti virtuálního sériového portu. Bohužel čtečka navzdory uváděným parametrům od výrobce takovou vlastností nedisponuje, funguje pouze na principu „klávesnice“. To znamená, že v případě sejmutí čárového kódu je výsledek stejný, jako bychom čárový kód napsali ručně pomocí klávesnice. V rámci konzultací ve firmě se však podařilo tuto čtečku úspěšně propojit s aplikací.

Řešení pro čtečku typu „klávesnice“ má jednu nespornou výhodu. Při připojení jakékoliv čtečky, která disponuje stejnou vlastností, není nutné do aplikace programovat podporu pro konkrétní čtečku například po sériové lince.

Pro zajímavost je na obrázku 5.5 uvedena posloupnost prvků a funkcí, které proběhnou při načtení čárového kódu v hlavním dialogu aplikace, čímž dojde k přidání produktu k objednávce.



Obrázek 5.5: Zjednodušený diagram sejmutí čárového kódu

6 Závěr

Cílem této bakalářské práce bylo vytvořit aplikaci, pomocí které bude pracovník v kamenném obchodě schopen rychle vytvořit a zpracovat nákup zákazníka bez použití externího skladového nebo účetního systému.

Jelikož aplikace je již v základu velmi rozsáhlá, bylo v rámci bakalářské práce úspěšně navrženo a naprogramováno jádro aplikace včetně několika základních funkcí. Celé řešení se skládá ze dvou částí - webové aplikace a desktopové aplikace.

Výhodou webové aplikace je zejména možnost využití existujících tříd e-shopu, které spolehlivě zajišťují základní operace.

Desktopová aplikace pak zobrazuje aplikaci webovou a zároveň figuruje jako zprostředkovatel mezi externími periferiemi a webovou aplikací. Díky tomu, že je veškerá logika přenesena na webový server a desktopová aplikace ji „jen zobrazuje“, je možné velmi snadno přidávat další funkce, případně provádět úpravy bez nutnosti aktualizace desktopové aplikace v kamenném obchodě.

Jedním z cílů bakalářské práce bylo testování nasazení v praxi, které mělo proběhnout přímo v kamenných obchodech. Firma se ale na základě dosažených výsledků a nečekané obsáhlosti celé problematiky rozhodla testování v praxi uskutečnit až v měsíci srpnu tohoto roku. Proběhlo však interní testování aplikace v rámci firmy. Při testech se ukázaly některé chyby, které jsem následně opravil a bylo zjištěno několik funkcí, které chybí, ačkoliv nebyly vyjmenovány jednotlivými obchodníky a nenarazil jsem na ně ani při rešerších existujících řešení. Jednou z takových chybějících funkcí je například možnost úpravy stavu platby „zaplaceno“ a „nezaplaceno“. Pokud některý zákazník platí tzv. „na fakturu“, dostane zboží s fakturou, kterou zaplatí později převodem na účet. Následně k objednávce je přiřazena platba se stavem „nezaplaceno“ a je tedy nutné aby bylo možné platbu zpětně editovat a změnit její stav na „zaplaceno“.

V projektu bych rád nadále pokračoval. Aplikaci bych dále rozšířil o výše zmíněné chybějící funkce. Aplikaci by bylo možné také rozšířit například o automatické propojení s platebními terminály. Další možností by bylo vytvoření knihovny pro komunikaci s některými dalšími druhy čteček čárových kódů, zejména pak čtečkami bezdrátovými nebo síťovými.

Literatura

[1] RISCHPATER, Ray. Application development with Qt creator a fast-paced guide for building cross-platform applications using Qt and Qt Quick. New Edition. Birmingham, UK: Packt Pub, 2013. ISBN 978-178-3282-319.

[2] Qt project [online]. [cit. 2014-10-06]. Dostupné z: http://qt-project.org/.

[3] CASTLEDINE, Earle, Myles EFTOS a Max WHEELER. *Vytváříme mobilní web a aplikace pro chytré telefony a tablety*. 1. vyd. Brno: Computer Press, 2013, 288 s. ISBN 978-80-251-3763-5.

[4] SATRAPA, Pavel. 2015. *LaTeX pro pragmatiky*. Dostupné také z: http://www.nti.tul.cz/~satrapa/docs/latex/.

[5] SATRAPA, Pavel. *Balík tul pro LaTeX* [online]. [vid. 17. 4. 2015]. Dostupné z: http://www.nti.tul.cz/~satrapa/vyuka/latex-tul.

[6] *Imonggo on Ipad* [online]. [cit. 2015-04-02]. Dostupné také z:https://www.imonggo.com/2013/images/imonggo_ipad.png.

[7] *Vend POS on Ipad* [online]. [cit. 2015-04-02]. Dostupné také z: http://www.thinn.co.za/wp-content/uploads/2014/06/ipad_sellmore.png.

[8] *POHODA Kasa Offline* [online]. [cit. 2015-04-02]. Dostupné také z: http://www.stormware.cz/Image/POHODA/POHODA_KasaOffline_dotykovy_displej.png.

[9] *Erply POS on iPad* [online]. [cit. 2015-04-02]. Dostupné také z: http://erply.com/wp-content/uploads/2013/07/Erply_iPad_PayPal-Here.png.

[10] wpj s.r.o *Webdesign studio Vrchlabí* [online]. [cit. 2015-05-10]. Dostupné z: http://www.wpj.cz/sluzby/e-shopy-snadno-a-rychle-kupshop/.

[11] MOVMENTO PTE. LTD. *POS Software Imonggo - Free Online Web-based Point of Sale System* [online]. [cit. 2015-05-10]. Dostupné z: https://www.imonggo.com/.

[12] STORMWARE S.R.O. *Účetní program POHODA* [online]. [cit. 2015-05-10]. Dostupné z: http://www.pohoda.cz/.

[13] THE QT COMPANY. Qt | *Cross-platform application & UI development framework* [online]. [cit. 2015-05-10]. Dostupné z: http://www.qt.io/.

[14] *Doctrine 2 ORM's documentation* [online]. [cit. 2015-05-10]. Dostupné z: http://docs.doctrine-project.org.

[15] NEW DIGITAL GROUP, INC. *Documentation | Smarty* [online]. [cit. 2015-05-10]. Dostupné z: http://www.smarty.net/documentation.

[16] *Bootstrap* [online]. [cit. 2015-05-10]. Dostupné z: http://getbootstrap.com/.

[17] JADRNÝ, Tomaš. *JQuery návod po začátečníky* [online]. [cit. 2015-05-10]. Dostupné z: http://jquery-navod.cz.

[18] JQuery - remove style added with .css() function. *Stackoverflow* [online]. [cit. 2015-05-10]. Dostupné z: http://stackoverflow.com/questions/4036857/jquery-remove-style-added-with-css-function.

[19] What is the best way to remove a table row with jQuery? *Stackoverflow* [online]. [cit. 2015-05-10]. Dostupné z: http://stackoverflow.com/questions/170997/what-is-the-best-way-to-remove-a-table-row-with-jquery.

[20] THE JQUERY FOUNDATION. *JQuery API Documentation* [online]. [cit. 2015-05-10]. Dostupné z: https://api.jquery.com/.

[21] Parsing returned HTML from jQuery AJAX request. *Stackoverflow* [online]. [cit. 2015-05-10]. Dostupné z: http://stackoverflow.com/questions/20007721/parsing-returned-html-from-jquery-ajax-request.

[22] THE JQUERY FOUNDATION. *Dialog | jQuery UI* [online]. [cit. 2015-05-10]. Dostupné z: http://jqueryui.com/dialog/.

[23] Reload an iframe with jQuery. *Stackoverflow* [online]. [cit. 2015-05-10]. Dostupné z: http://stackoverflow.com/questions/4249809/reload-an-iframe-with-jquery.

[24] How to get table row above (or before) my current row in jQuery? *Stackoverflow* [online]. [cit. 2015-05-10]. Dostupné z: http://stackoverflow.com/questions/7969387/how-to-get-table-row-above-or-before-my-current-row-in-jquery.

[25] Moqups. https://moqups.com/.

[26] Systém erply. *POS Software - Retail Point of Sale Software - POS System* [online]. [cit. 2015-05-10]. Dostupné z: https://erply.com/.

[27] Systém vend. VEND LIMITED. *POS Software - Best Retail Point of Sale Systems | Vend* [online]. [cit. 2015-05-10]. Dostupné z: http://www.vendhq.com/.