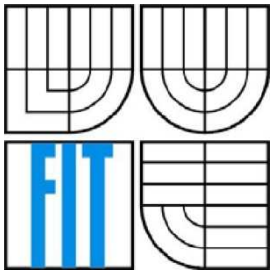
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO MARKETINGOVÉ ŠÍŘENÍ AGENDY ON-LINE PRODEJE

SYSTEM FOR MARKETING PROPAGATION OF ON-LINE SALE AGENDA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ DRDLA

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. Tomáš Hruška, CSc.

BRNO 2015

Abstrakt

Tato práce se zabývá návrhem informačního systému pro podporu internetového prodeje. V úvodní kapitole definuje řešený problém, dále analyzuje konkurenční softwarové řešení a shrnuje jejich klady a zápory. Z těchto údajů jsou dále definovány požadavky na navrhovaný systém. Jednotlivé funkce jsou podrobně rozepsány a výsledkem je návrh a popis implementace informačního systému. V poslední kapitole jsou rozebrány možnosti dalšího rozšíření informačního systému.

Abstract

This paper describes the design of an information system for online sales. There is the problem definition in first chapter, then the requirements are defined and competitive software solutions are analyzed. In the next chapters, individual functions and their implementations are described in detail and result of the chapter is the design of information system.

Klíčová slova

Informační systém, webová aplikace, internetový prodej, ASP.NET MVC, Visual Basic .NET, správa objednávek.

Keywords

Information systém, web application, online sales, ASP.NET MVC, Visual Basic .NET, order management.

Citace

Bc. Jiří Drdla: Systém pro marketingové šíření agendy on-line prodeje, diplomová práce, Brno, FIT VUT v Brně, 2015

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením prof. Tomáše Hrušky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

…….

Jiří Drdla

20. května 2015

Poděkování

Chtěl bych poděkovat vedoucímu své diplomové práce, panu prof. Tomáši Hruškovi, za poskytnuté konzultace a vhodné směrování při práci na projektu a zejména při tvorbě technické zprávy.

Obsah

1. Úvod 4

2. Analýza problému 5

2.1 Definice požadavků 5

2.1.1 Shrnutí požadavků 7

2.2 Konkurenční řešení 7

2.2.1 Manažer prodeje 8

2.2.2 Money S3 8

2.2.3 OpenCart 8

2.2.4 PrestaShop 9

2.2.5 Zhodnocení konkurenčních produktů 9

2.3 Diagram případů užití 10

3. Technické prostředky 11

3.1 ASP.NET MVC 11

3.1.1 Controller podrobně 12

3.2 Entity Framework 13

3.3 MySQL server 13

4. Návrh 15

4.1 Základní rozhraní aplikace 15

4.2 Funkce aplikace 16

4.2.1 Adresář 16

4.2.2 Objednávky 17

4.2.3 Zboží 18

4.2.4 Zásilky 19

4.2.5 Reklamace 20

4.2.6 Import objednávek 22

4.2.7 Import bankovních plateb 24

4.2.8 Uživatelé systému 25

4.3 Databáze 25

4.4 Diagram tříd ... 28
5. Implementace ... 29
5.1 Struktura projektu ... 29
5.2 Vzhled aplikace ... 31
5.2.1 Bootstrap a jQuery ... 31
5.2.2 Použitá rozšíření ... 32
5.2.3 Grafická šablona ... 34
5.2.4 Struktura pohledů ... 36
5.3 Správa uživatelů ... 37
5.3.1 ASP.NET Identity ... 38
5.3.2 Proces registrace uživatelů ... 39
5.4 Agendy v systému ... 39
5.5 Dashboard ... 42
5.6 Zajímavé části implementace ... 43
5.6.1 Akce vykonávané na pozadí ... 43
5.6.2 Generování emailových zpráv a nabídek zboží ... 45
5.6.3 Tisk dokumentů ... 48
5.6.4 Import a export dat ... 52
5.6.5 Zálohování a obnova ... 56
6. Propagace produktu ... 58
6.1 Možnosti propagace ... 58
6.1.1 Webová prezentace ... 58
6.1.2 Google AdWords ... 59
6.1.3 Bannerová inzerce ... 59
6.1.4 Sociální sítě ... 59
6.2 Zvolený způsob propagace ... 60
6.3 Reálné nasazení ... 61
7. Závěr ... 63
8. Literatura ... 64

9. Seznam použitých zkratek a symbolů 68

10. Seznam příloh 69

Kapitola 1

Úvod

Trendem posledních let ve světě maloobchodu je prodej zboží a služeb přes internet. Kromě velkých hráčů na tomto poli působí i celá řada malých internetových prodejců, kteří také tvoří nemalé procento celkového obratu v online prodeji. Nedílnou součástí při každém internetovém prodeji je však velké množství agendy, kterou se musí internetový prodejce zabývat. Větší korporace v tomto ohledu zpravidla sází na robustní informační systémy, jež dokáží mnohé rutinní činnosti ve velké míře automatizovat. Řešení v podobně na míru sestrojeného informačního systému je ovšem pro malé internetové prodejce ve většině případů nepřijatelně nákladné. Předmětem této diplomové práce je návrh a implementace informačního systému, zaměřujícího se právě na tyto menší prodejce. Cílem navrhovaného informačního systému je stejně jako u systémů větších společností usnadnit a zefektivnit činnosti spojené s internetovým prodejem.

Následující kapitola rozebírá řešený problém, definuje požadavky na navrhovaný informační systém a navrhnuté řešení srovnává s konkurenčními produkty. Podrobněji jsou popsány klady a zápory vybraných existujících řešení a nastíněn způsob, jakým bude chybějící funkčnost implementována v tomto projektu.

Kapitola třetí pojednává o technických prostředcích použitých pro implementaci navrženého systému. Popsány jsou zde použité technologie a prostředky, jejich srovnání s konkurencí a je zde zdůvodněn právě jejich výběr.

Čtvrtou kapitolou je návrh informačního systému, podrobně je popsána celková funkčnost systému a samostatně jednotlivé funkce. Součástí kapitoly jsou i nezbytné diagramy komplexně popisující návrh databáze a architekturu systému.

Kapitola pátá se zaměřuje na samotnou implementaci informačního systému. Nejprve jsou představeny její obecné části a principy fungování a poté jsou detailně rozebrány některé její zajímavé pasáže. Složitější části jsou opět graficky popsány pomocí diagramů.

V předposlední kapitole jsou rozebrány některé možnosti propagace, z nichž jedna je zvolena a realizována. Z realizace jsou posléze vyvozeny potřebné závěry a je nastíněn další postup v propagaci produktu.

Závěrem práce jsou shrnuty dosažené výsledky a jsou nastíněny možnosti pro další vývoj aplikace.

Kapitola 2

Analýza problému

V této kapitole se zaměříme na analýzu problému řešeného v tomto diplomovém projektu, určíme si jeho cíle a navrhované řešení srovnáme s jinými konkurenčními produkty.

Jak již bylo nastíněno v úvodu, navrhovaný informační systém se zaměřuje na menší internetové prodejce, kterým má poskytnout kvalitní a robustní nástroj pro usnadnění rutinních činností spojených s jejich prodejními aktivitami. Mezi základní činnosti, které internetového prodejce denně zaměstnávají, patří správa objednávek, zákazníků, evidence prodávaného zboží a expedice zásilek. K těmto činnostem se často nabaluje také vyřizování reklamací jednak od zákazníků, ale i směrem k dodavatelům.

Mnoho z těchto menších internetových prodejců nabízí své zboží kromě svého internetového obchodu také na celé řadě dalších internetových portálů, jako jsou aukční servery, slevové weby nebo internetová inzerce. A právě zde prodejci naráží na problém jak efektivně spravovat objednávky na několika různých prodejních portálech najednou. Zpravidla každý z takových portálů, ať už se jedná o internetový obchod, aukční server nebo slevový portál, nabízí prodejcům rozhraní pro správu zboží a objednávek. Obsluha několika takových rozhraní je však výrazně časově náročnější než objednávky a zboží spravovat z jednoho místa pro všechny portály najednou. A právě takové místo, zastřešující prodej na několika portálech pod jedno rozhraní, se snaží tato diplomová práce navrhnout.

Po několikerém mylném pochopení při představování tohoto projektu je třeba zdůraznit, že se nejedná o internetový obchod, a to ani v žádné jeho obměně. Implementovaný systém bude sloužit pouze prodejcům ke správě prodejní agendy a nikoli jejich zákazníkům. S tím také souvisí plánovaná existence uživatelského rozhraní systému výhradně pro tyto prodejce, ne pro zákazníky. Cílem systému je *agregovat data o prodejích* na jiných prodejních portálech, kterými mohou být mimo jiné i internetové obchody.

2.1 Definice požadavků

Správa objednávek je sice hlavním, ne však jediným cílem navrhované webové aplikace. Další podstatnou součástí je *evidence prodávaného zboží* a s ním souvisejících nákupů. Evidované zboží bude potřeba dále exportovat na prodejní portály popsané níže. Z tohoto důvodu je nutné u zboží uchovávat informace o jeho typu, zařazení do kategorie a je vhodné uvažovat o evidenci produktových obrázků a videí.

Zejména prodejci elektroniky a jiného finančně hodnotnějšího zboží mají potřebu v informačním systému evidovat nejen množství naskladněného zboží, ale také jednotlivé kusy a jejich *sériová čísla* či jiné specifické údaje. Tato funkčnost bude tedy do systému rovněž implementovaná, avšak u jednotlivých produktů bude volitelná.

O odstavec výše byla zmíněna nutnost implementace automatického *exportu evidovaného zboží* na prodejní portály. To s sebou nese nutnost automaticky generovat popisy zboží na základě požadavků jednotlivých prodejních portálů. Kupříkladu Aukro.cz vyžaduje čistě textový popis nebo popis ve formátu HTML. Avšak při použití HTML popisu nedovoluje vůbec užít některé tagy nebo JavaScript. Jiné, například inzertní, portály naopak povolují použití pouze textového popisu, z čehož vyplývá, že generování popisů pro všechny prodejní portály nebude možné provádět jednotně. V našem informačním systému tedy bude možné si pro každý prodejní portál definovat vlastní šablonu popisu zboží. Na jejím základě poté bude možné pro libovolný evidovaný produkt vygenerovat jeho popis pomocí jeho názvu, parametrů, atd. Teprve takto vytvořený popis se bude exportovat na zvolený prodejní portál.

Další podstatnou funkcí je spojenou s prodejními portály je možnost *stahovat objednávky zákazníků* uskutečněné na těchto prodejních místech. Je to klíčová funkce definující celý informační systém a pro prodejce takřka nejdůležitějším aspektem pro usnadnění práce spojené s prodejem. V závislosti na prodejním portále bude také možné stahovat data zákazníků a přidávat je do adresáře nebo evidovat platby uskutečněné přes daný prodejní portál, viz dále.

Dalším požadavkem v navrhovaném informačním systému bude *evidence reklamací*. Jelikož většina prodejců vyřizuje reklamace nejen svých zákazníků, ale rovněž reklamace k dodavatelům, bude tato agenda dvojího druhu. Přesto si budou reklamace zákazníků a u dodavatelů velice podobné. V každé reklamační zakázce bude evidováno přijaté a vydané zboží v rámci reklamace. V reklamačním procesu může být ve finále velice nápomocná evidence konkrétních sériových čísel u zvolených produktů. Díky tomu si bude moci prodejce ověřit, že kupující u něj reklamuje přesně ten kus zboží, který u něj před časem nakoupil. Toto je opět vlastnost, kterou disponují pouze ekonomické softwary a pokročilé informační systémy, ale žádné z dále uvedených konkurenčních řešení. Systém bude navíc umožňovat zasílat zákazníkům informační emaily o vyřízení reklamace, ale také například o odeslání objednávky.

Evidence zákazníků je spíše záležitostí podružnou, avšak nesmí chybět v žádném podobném informačním systému. Výjimkou je Manažer prodeje od Aukro.cz, který speciální evidenci zákazníků nemá a k adresám zákazníků se přestupuje pouze prostřednictvím objednávek. Náš navrhovaný systém však tuto funkci bude implementovat.

Systém bude umožňovat evidovat k objednávkám přijaté bankovní platby a také platby realizované přes platební brány jednotlivých prodejních portálů, jmenovitě například platební brána PayU od Aukro.cz. S *evidencí bankovních plateb* půjde ruku v ruce také *evidence bankovních účtů*. Pro vybrané bankovní subjekty a jejich systémy internetového bankovnictví bude v našem informačním systému implementována možnost automatického importu přijatých plateb a spárování s objednávkami.

Systém bude dále disponovat *evidencí odeslaných poštovních zásilek* s možností tisku nejrůznějších poštovních dokumentů, jako jsou balíkové štítky, podací lístky, poštovní poukázky a další. Evidence odeslaných zásilek není přítomna ani v jednom z uvedených existujících řešení, přitom pro prodejce je přehled o zásilkách také jednou z priorit.

Poslední, ale neméně důležitou součástí navrhovaného systému je samozřejmě možnost registrovat do našeho systému nové prodejce, kteří jej budou používat pro své prodejní aktivity. Systém bude pro běžné používání zpoplatněn, avšak po omezenou dobu jej bude možné využívat bezplatně pro účely vyzkoušení a seznámení se.

2.1.1 Shrnutí požadavků

Z výše uvedených odstavců si tedy v bodech určeme, jaké nároky máme na navrhovaný informační systém:

- Správa objednávek
- Import objednávek z prodejních portálů
- Evidence zboží, včetně parametrů a sériových čísel
- Export zboží na prodejní portály, generování popisů
- Správa reklamací zákazníků a reklamací u dodavatelů
- Evidence odeslaných zásilek
- Tisk poštovních dokumentů a přehledů dle šablon
- Sledování přijatých plateb
- Import přijatých plateb z internetového bankovnictví
- Evidence zákazníků
- Registrace uživatelů systému

Z těchto základních požadavků vzejdou v průběhu návrhu další dílčí potřeby a požadavky, které bude nutné v aplikaci implementovat. Řešení těchto a dalších požadavků si představíme v následujících kapitolách.

2.2 Konkurenční řešení

V této podkapitole si představíme některá potenciálně konkurenční řešení tohoto projektu. Ve srovnání budou představena řešení vhodná pro použití malými internetovými prodejci, nikoli rozsáhlé a drahé informační systémy určené pro korporátní použití.

2.2.1 Manažer prodeje

Webová aplikace s názvem Manažer prodeje [9] je malý informační systém vyvinutý společností Aukro s.r.o. pro podporu prodejců na stejnojmenném prodejním portálu. Systém opravdu cílí pouze na prodej přes Aukro.cz a nedovoluje napojení na žádný jiný prodejní systém jako například e-shop. Není ani možné modifikovat zákazníkem vytvářené objednávky. Chybí také napojení na systémy internetového bankovnictví a nemá správu reklamací. Obsahuje však vcelku povedený systém pro správu skladu, evidenci nákupů a také správu plateb přes partnerskou platební bránu PayU. Výhodou je rovněž možnost přehledné grafické editace popisů evidovaného zboží. Systém dále umožňuje export dat z objednávek pro přepravní službu a omezeně i tisk poštovních dokumentů. Prodejci využívající tento prodejní systém ocení také jeho cenu, kterou je symbolická 1 koruna měsíčně.

Zásadní výhodou našeho navrhovaného systému tedy bude napojení na vícero prodejních portálů, správa reklamací a širší možnosti tisku dokumentů.

2.2.2 Money S3

Tento systém není přímým konkurentem navrhované webové aplikace. Především se jedná o desktopové řešení a ze svého principu ekonomický software. Tento software poskytuje ve svém základu velice pokročilou správu objednávek a nákupů, vedení databáze zákazníků a propracovanou evidenci skladu. Díky množství zásuvných modulu ovšem umožňuje rozšířit svoji funkcionalitu také o evidenci reklamací a import plateb z banky. Cena tohoto řešení včetně zásuvných modulů ovšem přesahuje 5500 Kč včetně DPH. Zcela zvláštní záležitostí je napojení systému Money S3 na nějaký prodejní portál. To je možné realizovat pouze prostřednictvím speciálního datového konektoru v ceně dalších 4999 Kč včetně DPH. Slouží však zejména pro import objednávek a faktur z e-shopu. Neposkytuje možnost evidované zboží do e-shopu exportovat. [10]

Navrhovaný systém nebude softwaru Money konkurovat v oblastech účetnictví, ale poskytne zejména cenově přijatelnější alternativu pro usnadnění činnosti při práci se zbožím a automatizaci v oblasti exportu nabízeného zboží.

Společnost Cígler Software je provozovatelem ještě jednoho potenciálně konkurenčního produktu a tím je iDoklad. Jedná se o webovou aplikaci určenou výhradně pro tvorbu faktur. Tato aplikace je poskytována zdarma, avšak nenabízí nic jiného než možnost vytváření faktur a evidenci zákazníků.

2.2.3 OpenCart

Systém s tímto názvem je v prvé řadě open source řešením internetového obchodu. Pro naše porovnání se však zaměříme na administrační část systému, která může potenciálně nabízet podobnou funkčnost jako námi navrhovaný informační systém. [11]

OpenCart opět poskytuje propracovanou správu objednávek, evidenci zboží a zákazníků. Pomocí zásuvných modulů lze dodat také evidenci nákupů, tisk dle přednastavených tiskových šablon a v omezené míře též správu reklamací. Chybí však jakékoli napojení na české bankovní systémy a rovněž nejsou dostupné datové konektory pro žádný z českých prodejních portálů. Jakkoli by šla tato funkčnost více či méně doplnit, u zásuvných modulů je prakticky vždy problém s jejich funkčností napříč jednotlivými verzemi systému.

Oproti tomuto systému tedy navrhovaná webová aplikace nabídne napojení na další prodejní portály, správu bankovních plateb, zásilek a poskytne širší možnosti tisku.

2.2.4 PrestaShop

Stejně jako u systému OpenCart se jedná především o e-shopové řešení a potenciální konkurence tkví zejména v jeho administrační části. Ta stejně jako u OpenCart nabídne velice dobrou správu objednávek, zboží a zákazníků. Díky rozšířenosti větší než má systém OpenCart, existuje také širší nabídka zásuvných modulů. Ty dokáží přidat mezi stávající funkcionalitu reklamace, zásilky, širší možnosti tisku i napojení na prodejní portály jako Aukro.cz nebo Slevomat.cz. Problémem však je stále chybějící podpora pro napojení na systémy internetového bankovnictví nebo vytváření popisu zboží na základě parametrů. Výhodou systému PrestaShop i výše změněného OpenCart je, že jsou open-source a i pro komerční účely zdarma. [12]

Oproti systému PrestaShop tedy navrhované řešení nabídne napojení na systémy internetového bankovnictví, automatizovanou tvorbu popisu zboží na základě parametrů a dle zkušeností také kvalitnější podporu tisku. Ať už půjde o tisk poštovních dokumentů k zásilkám nebo například tisk faktur.

2.2.5 Zhodnocení konkurenčních produktů

Výše bylo představeno pouze několik produktů, které by v budoucnu mohly být konkurencí pro navrhovanou webovou aplikaci. Jedná se samozřejmě o úzký výběr a na trhu existuje mnoho jim podobných řešení.

Současné existující produkty lze obecně rozdělit do dvou kategorií. Tou první je skupina software primárně zaměřená na vedení účetnictví, která funkce spojené s prodejem implementuje spíše okrajově a tím pádem i dosti omezeně. Takovým případem je právě Money S3 a jemu podobné ekonomické systémy.

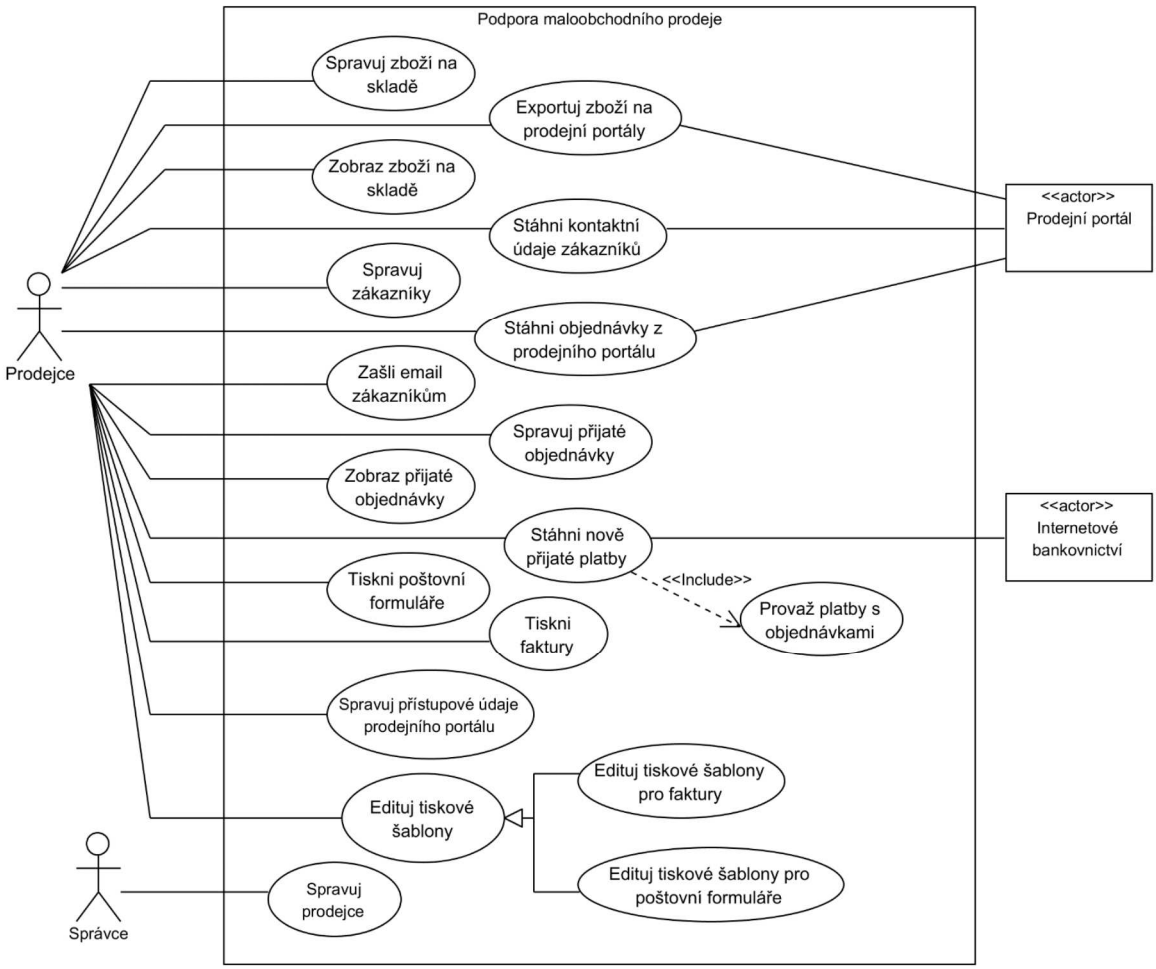
Druhou kategorií je skupina aplikací, z nichž většina existuje v online verzi, která se zaměřuje více na činnosti spojené se samotným prodejem, ale jejich použití je spojeno pouze s jediným prodejním portálem. Příkladem takové aplikace je Manažer prodeje, který lze použít pouze k administraci prodejů na Aukro.cz. Představená řešení v podobě open-source internetových obchodů sice v některých případech poskytují napojení i na jiné prodejní portály kromě vlastního

prodejního rozhraní, avšak z osobních zkušeností je takové napojení vždy poněkud násilné ne vždy bezproblémové.

Cílem projektu je tedy vytvořit webovou aplikaci, která dokáže agregovat data o prodejích z více prodejních portálů a tyto portály také pomocí jednotného rozhraní plnit nabídkou zboží prodejce. Aplikace by měla být zaměřená primárně na usnadnění samotných prodejních procesů a méně řešit otázky vedení účetnictví.

2.3 Diagram případů užití

V předchozích podkapitolách jsme si specifikovali požadavky na navrhovaný informační systém a tyto požadavky zkonfrontovali s několika existujícími řešeními. Pro lepší představu při následném návrhu informačního systému si navržené požadavky vizualizujeme. K tomu použijeme diagram případů užití, výsledek je vidět na obrázku 2.1.



Obrázek 2.1: Diagram případů užití pro navrhovaný informační systém

Kapitola 3

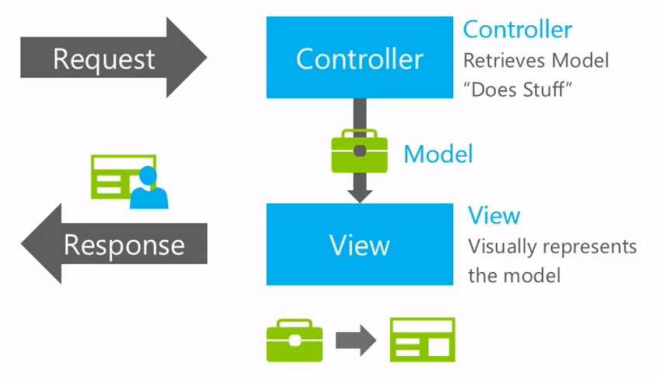
Technické prostředky

Kapitola třetí se zaměřuje na představení nástrojů použitých při návrhu a následné implementaci projektu. Podrobněji rozebrány zde budou použité technologie a frameworky a použitý databázový backend.

3.1 ASP.NET MVC

> *"ASP.NET MVC is a radical shift for web developers using Microsoft platform. It emphasizes clean architecture, design patterns, and testability, and doesn't try to conceal how the Web works."* [33]

V první řadě si uvedeme základní technologii, ve které bude řešen následný diplomový projekt. Tou technologie je ASP.NET MVC vyvinutá společností Microsoft. Framework je vyvinutý pod licencí Apache License 2.0 a je kompletně open-source. Jedná se o framework pro tvorbu webových aplikací, který implementuje vzor Model-View-Controller. Technologie vychází ze staršího ASP.NET, ovšem přináší úplně nový třívrstvý pohled na architekturu aplikace. Těmi vrstvami je již zmiňovaný model, view a controller.



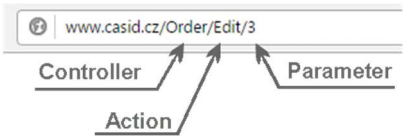
Obrázek 3.1: Schéma vzoru MVC [15]

Model představuje reprezentaci informací, se kterými se v aplikaci pracuje. Jeho obsahem jsou rovněž operace nad těmito daty. View (pohled) je určen k převodu dat modelu do podoby vhodné k reprezentaci uživateli. A ve finále Controller (řadič) je součást, která zodpovídá za reakce na požadavky uživatele, zajišťuje změny v modelu a vytváření pohledu.

Výhodou při použití frameworku ASP.NET MVC je vysoká míra abstrakce nad samotným řízením toku dat, uchováváním session, atd. Framework za nás dokáže řešit validaci uživatelem zadávaných dat, využívá technologii AJAX pro vytváření interaktivnějších webových prezentací a další. [1] Aplikace bude vytvářena v prostředí Visual Studia 2013 získaného na základě akademické licence v programu DreamSpark.

3.1.1 Controller podrobně

Pokud nějaká část architektury MVC stojí za bližší představení, je to právě controller. Controller v terminologii ASP.NET MVC je specializací třídy System.Web.Mvc.Controller. [6] Implementované metody této třídy se nazývají Akce (Action). Tato terminologie je velice důležitá a nezbytná pro pochopení způsobu jakým architektura ASP.NET MVC funguje. Na obrázku 3.2 vidíme adresu požadavku na webový server s nasazenou ASP.NET MVC aplikací.



Obrázek 3.2: Adresa požadavku

Nyní si pojďme adresu z obrázku rozebrat a vysvětlit si na ní funkci controlleru. První část tvoří standardně doménové jméno webového serveru, za prvním lomítkem vidíme název controlleru, který náš požadavek bude zpracovávat, za dalším lomítkem název Akce, tedy metody, do níž se zasílaný požadavek předá, a nakonec parametry. Ukázka na obrázku tedy říká, že požadujeme stránku pro editaci objednávky s identifikačním číslem 3. Samotný kód controlleru pak může vypadat následovně:

```
Public Class OrderController
    Inherits Controller
    ...
    Function Edit(ByVal Id As Integer?) As ActionResult
        ...
    End Function
End Class
```

Logika ASP.NET pomocí reflexe [16] z požadavku automaticky zjistí, na který controller a kterou jeho akci je požadavek cílen, vytvoří instanci požadovaného controlleru a jeho metodu zavolá i s předávanými parametry.

Předávaným parametrem nemusí být pouze hodnoty primitivních datových typů. ASP.NET disponuje funkcí zvanou *model binding*, která přijaté parametry dovoluje interně konvertovat a volané metodě předat ve formě objektu. Metoda využívající funkci model binding může vypadat následovně.

```
Function SendEmail(Model As Order, IdTemplate as Integer) As ActionResult
    ...
End Function
```

Na příkladu vidíme akci pro zaslání emailové zprávy k objednávce, přičemž objednávka je předávána již ve formě objektu, avšak z webového prohlížeče byla odeslána jako HTML formulář, tedy jako kolekce dvojic klíč-hodnota. Model binding je velice mocný nástroj, usnadňující mnoho rutinních činností. Na druhou stranu se zatím nedokáže vypořádat například s hierarchickou strukturou objektů. Jak byl tento problém vyřešen v tomto projektu, se dozvíme v kapitole 5.4. Předmětem této práce však není popis funkce ASP.NET, představeny byly pouze funkce nezbytné pro další čtení této práce. Více o technologii je možné se dozvědět z odkazované literatury.

3.2 Entity Framework

Jedná se o další technologii z dílny Microsoftu používanou v tomto projektu. Entity Framework je opět open-source framework zajišťující objektově-relační mapování. Jak bude zmíněno dále, pro ukládání dat v navrhované aplikaci bude užito klasické relační databáze. Tento způsob uložení dat je v současné době nejrozšířenější, avšak pro zpracování dat v objektově orientované aplikaci je tabulková reprezentace záznamů obtížněji zpracovatelná. Z tohoto důvodu existují frameworky, ke kterým se řadí i Entity Framework, pro mapování klasické relační databáze na sadu objektů přímo využitelnou v aplikaci. [3] Entity Framework je také vydáván pod licencí Apache License 2.0.

3.3 MySQL server

Jako databázový backend poslouží volně šiřitelný databázový systém MySQL od společnosti Oracle. Jedná se o jednu z nejpoužívanějších relačních databází, která bude pro účely projektu plně postačovat. [4]

Při vybírání databázové technologie se nabízely též technologie od Microsoftu, avšak zejména kvůli technickým omezením jejich bezplatných verzí jsem vsadil právě na MySQL. Jednou z alternativ bylo použít Microsoft SQL Server v bezplatné Express verzi. Ta však nese velice citelné omezení na velikost databáze a to 2GB. To může být problém například při ukládání objemných obrazových dat k evidovanému zboží, které si popíšeme dále v kapitole 4. Další variantou bylo použití Microsoft SQL

Serveru Compact. Ten je sice taktéž dostupný bezplatně, je to ovšem pouze souborová databáze pouze se základní funkčností. Nedovoluje například definovat pohledy, triggery a další.

Kapitola 4

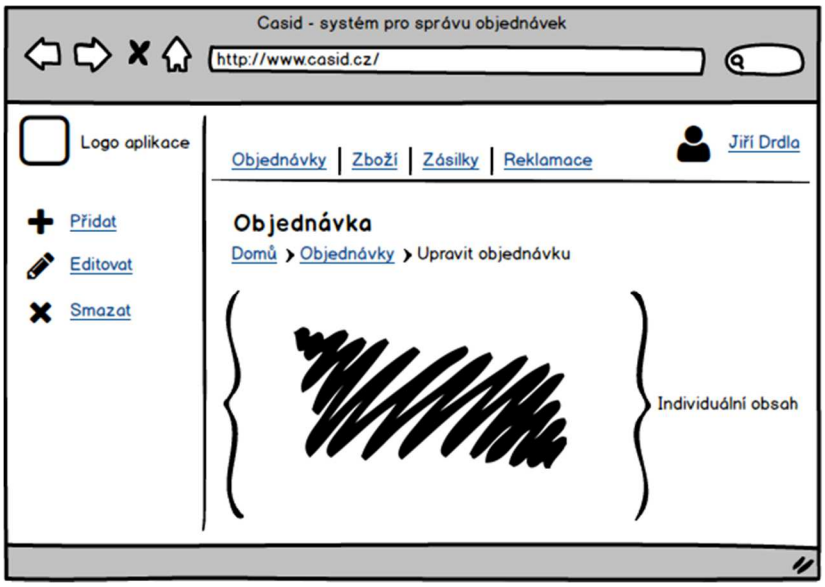
Návrh

V této kapitole se zaměříme na návrh samotné aplikace a jednotlivých programových funkcí a to podle požadavků specifikovaných v kapitole 2 – Analýza problému. Každé navrhované funkcionalitě bude věnována speciální podkapitola a funkčnost bude v případě potřeby reprezentována odpovídajícími diagramy.

4.1 Základní rozhraní aplikace

Do základního rozhraní aplikace vstupuje uživatel po přihlášení do systému. Na náčrtu v obrázku 4.1 je patrné, jak by toto rozhraní mělo vypadat. Obrazovka aplikace se dělí do třech hlavních částí, kterými jsou horní lišta, levý postranní panel a individuální obsah, zaujímající největší prostor. Tímto obsahem může být například tabulkové zobrazení položek při výběru agendy nebo detail konkrétní vybrané položky.

Následující a všechny další náčrty obrazovek byly vytvořeny v programu Balsamiq Mockups. [40] Jedná se pouze hrubé náčrty s omezenou množinou ovládacích prvků, výsledná podoba grafického rozhraní se může lišit.



Obrázek 4.1: Návrh rozhraní webové aplikace

Horní lišta v aplikaci zobrazuje logo informačního systému, v její pravé horní části budou informace o přihlášeném uživateli, ale zejména obsahuje sadu tlačítek pro přepínání agend v aplikaci. Těmito agendami jsou například seznam objednávek evidovaných v aplikaci, výpis předmětů, odeslaných zásilek nebo evidovaných reklamací.

Levý postranní panel v sobě bude sdružovat funkce dostupné pro aktuálně vybraný pohled. Pro příklad, pokud bude v horní liště zvolený pohled *Objednávky*, v levém postranním panelu budou dostupné volby pro přidání nové objednávky, editace nebo smazání aktuálně označené objednávky a dále například funkce pro zobrazení celkové ceny označených objednávek, tisk faktur nebo třeba import a export.

Hlavní část obrazovky ovšem vyplňuje individuální obsah. Tímto obsahem může být tabulkový výpis evidovaných položek nebo detail konkrétní položky. V tabulkovém výpisu se zobrazují položky na základě pohledu zvoleného v hlavním menu. Tedy například při zvolení *Zboží* se v tabulkovém výpisu zobrazí seznam evidovaného zboží, přičemž každý řádek tabulky odpovídá jednomu evidovanému produktu. Ve sloupcích tabulky poté budou informace jako ID produktu, název, typ, prodejní cena, atd. Sloupce tabulkového zobrazení budou uživatelsky přizpůsobitelné, tedy bude možné měnit jejich pořadí nebo bude možné nepotřebné sloupce skrývat. Kliknutím na označený řádek bude možné otevřít detail položky. Detaily položek si probereme jednotlivě v následující podkapitole.

Obrazovky detailu pro všechny typy položek budou mít podobná grafická rozhraní, jak bude vidět na obrázcích v následujících podkapitolách. V horní části obrazovky bude zobrazen název otevřené položky a cesta v rámci agendy. Pod pouto informací bude přítomna řada voleb. Po levé straně to budou tlačítka pro uložení provedených změn a pro zavření detailu bez uložení. Vpravo od nich se poté budou nacházet volby specifické pro konkrétní otevřenou položku. Pod těmito volbami se budou skrývat akce, které bude možné s položkou provádět. Spodní část bude poté vyhrazena jednotlivým vstupním polím, kde budou zobrazeny detailní informace.

4.2 Funkce aplikace

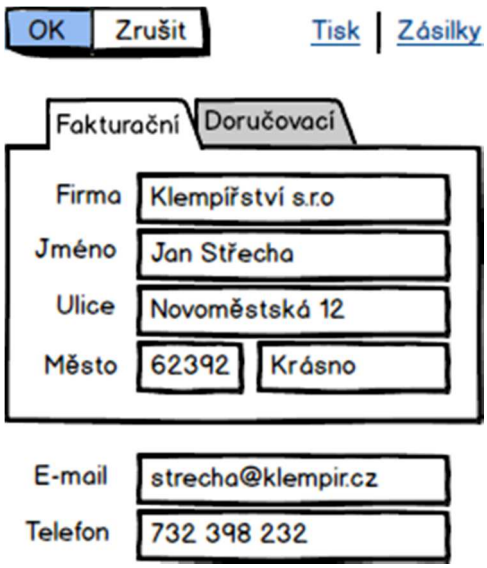
Tato podkapitola shrnuje jednotlivé navrhované funkce aplikace, naznačuje jejich budoucí grafické rozhraní a složitější procesy modeluje pomocí adekvátních diagramů. Rozebrány jsou postupně správa objednávek, zboží, nákupů, zákazníků a reklamací, dále import objednávek a export produktů na prodejní portály a také import plateb z internetového bankovnictví.

4.2.1 Adresář

Adresář slouží k evidenci zákazníků, kteří u prodejce v minulosti nakoupili nebo teprve nakoupí. U zákazníků je nezbytné evidovat dvě adresy – fakturační a doručovací, dále IČO a DIČ a kontaktní údaje jako email a telefon. Stejně jako u objednávek lze zákazníky přidávat do systému ručně nebo automatickým importem z některého prodejního portálu. Teoreticky není nezbytné uchovávat

kontaktní údaje zákazníků v systému zvlášť, jelikož všechny potřebné údaje již máme uloženy v objednávkách. Ani některé konkurenční aplikace, jako například Manažer prodeje od Aukro.cz tuto funkčnost nemají. Funkce však může být nápomocná například při vyhledávání adresy zákazníků dle kontaktních údajů atp.

Na obrázku 4.2 vidíme, jak by mělo ve výsledku vypadat okno pro zobrazení detailu evidovaného zákazníka, říkejme mu například *Karta zákazníka*.



Obrázek 4.2: Návrh okna detailu zákazníka

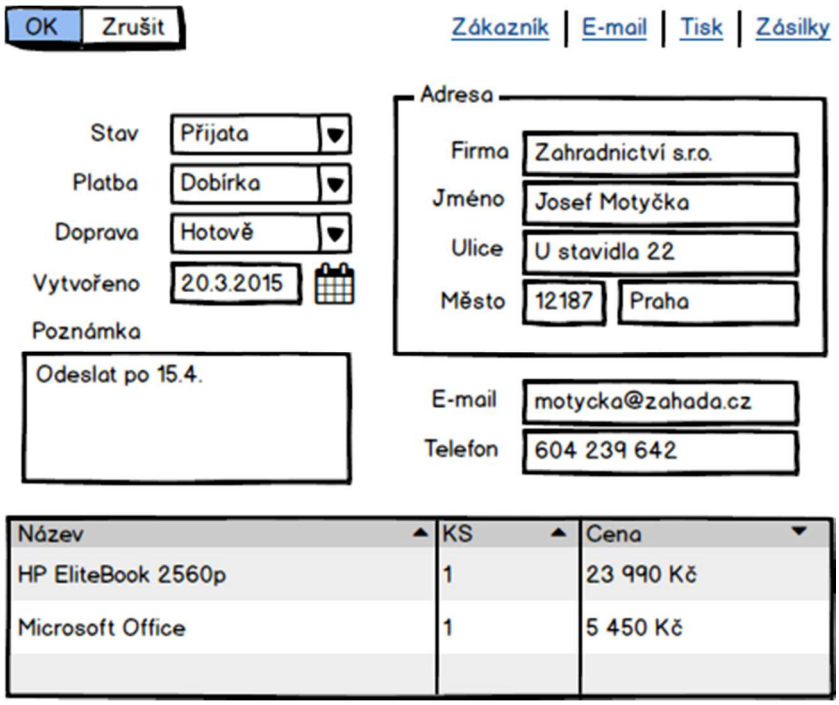
4.2.2 Objednávky

Správa objednávek je jednou z hlavních devíz navrhované webové aplikace a to z toho důvodu, že právě objednávky jsou záležitostí, se kterými budou internetoví prodejci pracovat nejčastěji. Objednávky je možné přidávat buď ručně, nebo automatickým importem z některého prodejního portálu, jak bude ukázáno v dalších podkapitolách. Při zobrazení objednávek budou dostupné standardní CRUD operace, tedy přidání, čtení, editace a mazání objednávky. Dále pak tisk dle zvolené tiskové šablony, funkce pro hromadnou změnu stavu, funkce pro zobrazení součtu cen označených objednávek a speciální volba *Vyřídit objednávky*.

Volbou *Vyřídit objednávky* je funkčnost využívaná vždy při expedování objednaného zboží. Tato volba zajistí odeslání informačního emailu zákazníkovi s informací o expedování zboží, změnu stavu vybraných objednávek na *Vyřízené* a označení souvisejících zásilek jako *Odeslané*.

Na obrázku 4.3 nalezneme první grafický návrh okna pro zobrazení detailu objednávky. V jeho horní části najdeme prvky pro uložení změn, zavření bez uložení a funkce dostupné pro aktuálně zobrazenou objednávku. Pod těmito prvky následují již volby specifické pro objednávku. Těmi budou volby pro změnu stavu objednávky, způsobu dopravy a platby, dále fakturační a doručovací adresa zákazníka, kontaktní údaje nebo poznámka. Spodní část okna detailu poté vyplňuje

editovatelný seznam položek objednávky. Seznam bude možné ručně editovat sadou funkcí dostupných v liště nad ním.



Obrázek 4.3: Návrh okna detailu objednávky

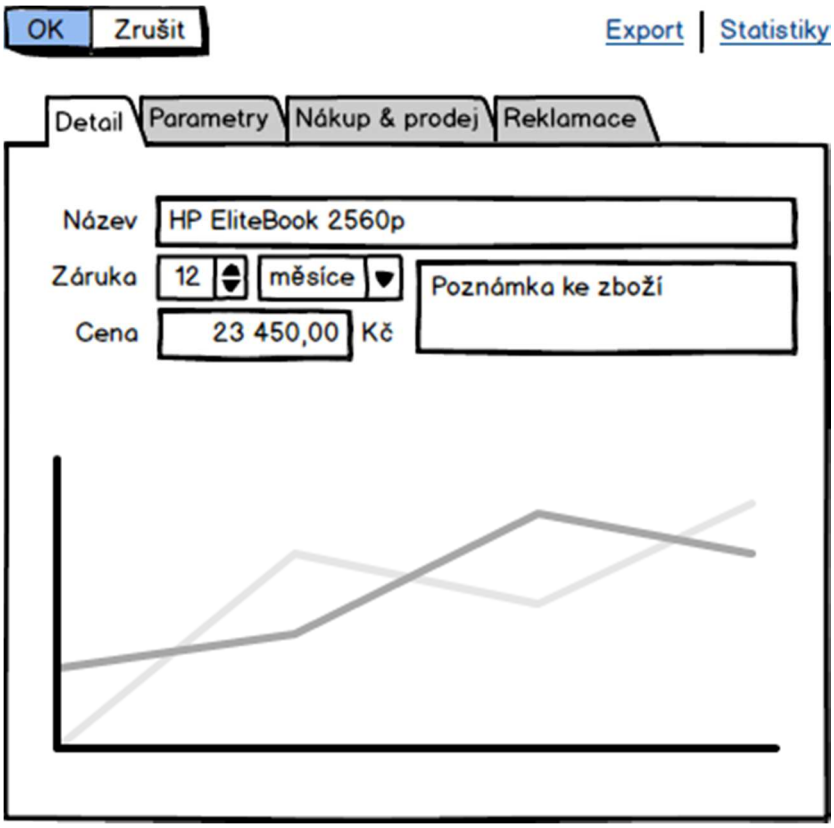
4.2.3 Zboží

Evidence prodávaného zboží je opět důležitou součástí agendy maloobchodního prodejce. U zboží je nutné sledovat jeho skladové pohyby, tedy nákupy a prodeje a počet kusů na skladě. Do systému budou produkty vstupovat prostřednictvím evidovaných nákupů, viz dále, a znovu vystupovat pomocí objednávek zákazníků, více v předchozí kapitole. Kromě toho se v naší aplikaci zaměříme i na evidenci parametrů zboží nebo produktových obrázků. Webová aplikace bude pro své uživatele vytvářet souhrnné statistiky pro konkrétní položky zboží, jako jsou obrat, počty nakoupených a prodaných kusů, vývoj cen nebo ve vztahu k evidovaným reklamacím a také poruchovost.

Potřebnými jednoduchými parametry pro evidenci zboží bude samozřejmě jeho název, typ, aktuální prodejní cena a nezbytná poznámka. Rozložení jednotlivých grafických prvků v okně formuláře pro detail zboží nalezneme na obrázku 4.4.

Jak bylo zmíněno výše, každé evidované zboží bude možné parametrizovat. K tomuto účelu budou v systému přítomny tzv. *šablony parametrů*. Pro každý typ prodávaného zboží bude možné vytvořit takovou šablonu parametrů, která bude předurčovat, jakými parametry může daný produkt disponovat. Kupříkladu, pokud bude naši aplikaci využívat prodejce nábytku, bude typicky mít v evidovaném sortimentu několik druhů židlí, stolů nebo skříní. Pro každý takový druh nábytku bude

definována zvláštní šablona parametrů. V šabloně s názvem *Stoly* poté budou definovány parametry jako materiál, dekor, rozložitelnost, rozměry, atd. Jakmile pak prodejce začne evidovat nový produkt typů *Stůl*, bude možné u něj tyto parametry vyplnit a následně také použít například pro automatické vytváření popisů zboží, viz dále v této kapitole.



Obrázek 4.4: Návrh okna detailu zboží

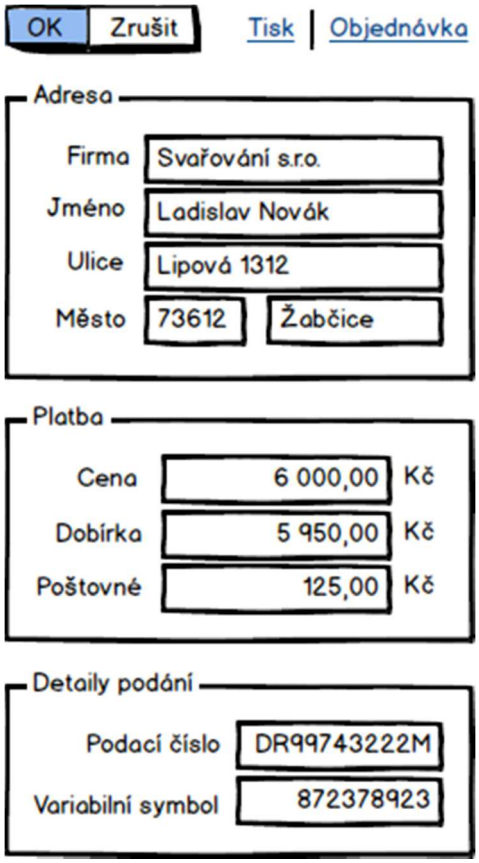
4.2.4 Zásilky

Podstatnou záležitostí pro internetové prodejce je evidence zásilek. Jelikož takový prodejce většinu svého prodeje realizuje na dálku, je pro něj důležité mít přehled také o odchozích zásilkách. Z tohoto důvodu bude evidence zásilek přítomna i v naší navrhované aplikaci. Zásilky bude možné v systému vytvářet ručně, ale i na základě existujících objednávek, zákazníků nebo reklamací. V zobrazení zásilek na hlavní stránce aplikace budou opět k dispozici základní CRUD operace a k nim dále možnost tisku nejrůznějších poštovních dokumentů dle tiskových šablon a export zásilek do formátu CSV či XML pro použití v systémech zásilkových služeb.

U zásilek bude možné definovat adresu příjemce, jeho kontaktní údaje, dále cenu zásilky, dobírkovou částku, cenu poštovného, číslo účtu a variabilní symbol pro zaslání dobírkové částky. Nesmějí chybět ani podací údaje jako číslo zásilky, typ zásilky a podstatná může být i hmotnost.

Většinu těchto údajů je možné při vytváření zásilky přejmout z objednávky, respektive od zákazníka, případně od evidované reklamace.

Jak již bylo nastíněno výše, evidence zásilek slouží nejen pro vytváření přehledu o odeslaných zásilkách, ale rovněž pro možnost tisku nejrůznějších dokumentů k těmto zásilkám. Počítá se s možností tisku tří druhů balíkových štítků (2 obecné a 1 dle vzoru České pošty [13]), s tiskem poštovních poukázek (Poštovní dobírková poukázka A a C od České pošty [14]), podacího lístku a podacího archu (rovněž po vzoru České pošty). Sada připravených šablon bude samozřejmě uživatelsky rozšiřitelná a tak přizpůsobitelná na míru každému prodejci. V budoucnu se počítá s vytvořením dalších defaultních šablon pro všechny prodejce a pro další přepravce. Na obrázku 4.5 vidíme předpokládaný vzhled formuláře pro zásilku.



Obrázek 4.5: Návrh okna detailu zásilky

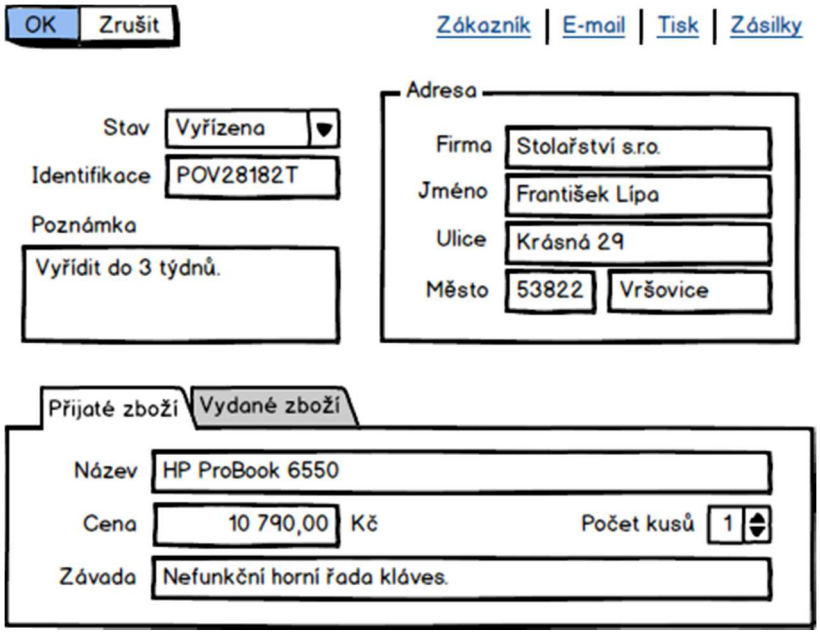
4.2.5 Reklamace

Důležitou navrhovanou součástí webové aplikace je evidence reklamací. Tuto funkci zároveň nenabízí ve výchozím stavu žádné ze zmiňovaných konkurenčních řešení. Zejména pro maloobchodní prodejce techniky různého druhu je to však funkce velice podstatná. Funkce se bude velice podobat funkci evidence objednávek. V pohledu *Reklamace* v hlavním okně aplikace bude

možné znovu provádět standardní CRUD operace a spolu s nimi funkce jako tisk reklamačního protokolu, odeslání emailu o vyřízení reklamace, hromadná změna stavu a export.

Reklamační protokol a vzhled odesílaného emailu budou definovány pomocí šablon. Šablony budou uživatelsky editovatelné pomocí voleb v nastavení, viz dále.

V rámci reklamace bude nutné evidovat fakturační a dodací adresu zákazníka, jeho kontaktní údaje, krátký identifikátor reklamace a poznámku. Tato pole budou dostupná v horní části okna formuláře. Spodní část okna bude vyhrazena dvěma oddílům pro definici přijatého a vydaného zboží v rámci reklamace. V levé části bude zboží k reklamaci přijaté, v pravé poté zboží z reklamace vydané. Tyto položky bude možné do reklamace přidávat buď ručně, nebo na základě předmětů evidovaných v naší webové aplikaci. Celou reklamaci bude také možné vytvořit na základě nějaké již existující objednávky. V tomto případě se do reklamace přejme adresa s kontaktními údaji zákazníka a položkami, nebo z evidovaného zákazníka a v tomto případě se přejme fakturační a doručovací adresa a jeho kontaktní údaje.

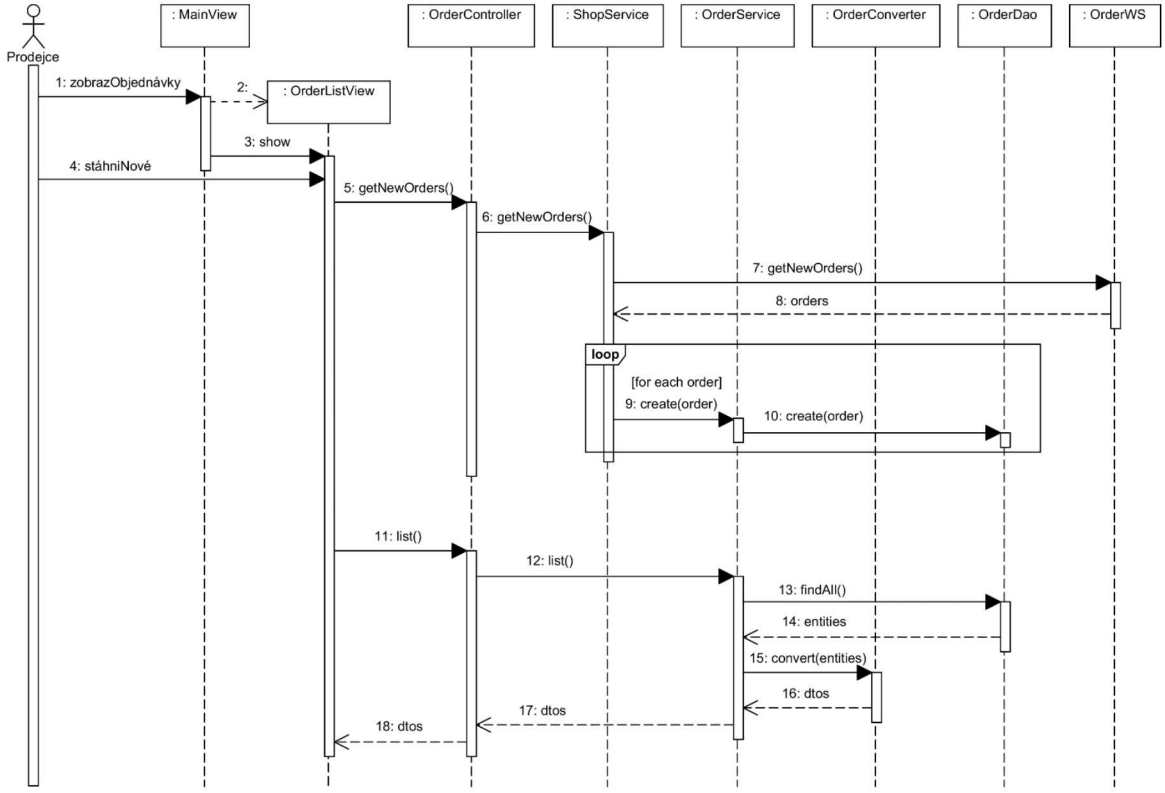


Obrázek 4.6: Návrh okna detailu reklamace

Na obrázku 4.6 vidíme, jak by mělo ve výsledku vypadat okno pro reklamaci zákazníka. V jeho horní části nalezneme volby specifické pro aktuálně otevřenou reklamaci a těmi jsou *Tisk* (například pro tisk reklamačního protokolu), *Zákazník* pro výběr zákazníka, *Email* pro odeslání informačního emailu a *Historie* pro zobrazení dříve prováděných úkonů v reklamaci.

4.2.6 Import objednávek

Import objednávek je jednou z nejdůležitějších funkcí v navrhované aplikaci, jelikož právě ta dokáže velice usnadnit práci spojenou s prodejem zboží. Pod importem objednávek si můžeme představit funkci, která z prodejních portálů, kterých prodejce ke svému prodeji využívá, stáhne objednávky, data a informace o platbách do naší webové aplikace. Takovým prodejním portálem může být například výše zmiňované Aukro.cz. Na tomto portálu má prodejce, uživatel naší aplikace, svůj profil, a pokud zákazník zakoupí některý z jeho vystavených předmětů, importem objednávek se tato informace přenese do námi navrhované webové aplikace. Na obrázku 4.7 nalezneme diagram interakce reprezentující způsob stahování objednávek a dat zákazníků z prodejního portálu.



Obrázek 4.7: Diagram interakce pro import objednávek

V následující podkapitole si navrhneme konkrétní způsob importu objednávek pro jeden z prodejních portálů a tím bude Aukro.cz. Import z dalších navrhovaných prodejních řešení, jako je například e-shopový systém OpenCart, se bude řešit obdobně.

4.2.6.1 Specifikace případů užití

Jelikož se bude jednat o jednu z komplikovanějších operací v našem informačním systému, při níž může docházet k chybám, využijeme jednu z modelovacích technik jazyka UML a tuto operaci si podrobně popíšeme ve specifikaci případů užití. Specifikace tohoto konkrétního případu užití vychází z diagramu případů užití v kapitole 2.

ID:	1
Název:	Stažení nových objednávek z prodejního portálu
Popis:	Informační systém stáhne z prodejního portálu data objednávek do vlastní databáze
Primární účastníci:	Prodejce
Sekundární účastníci:	Prodejní portál
Předpoklady:	Žádné
Následné podmínky:	1. Do databáze systému jsou vloženy nové objednávky z prodejního portálu 2. Systém zobrazí seznam nových objednávek
Akce pro spuštění:	Prodejce zvolí *Stáhnout nové objednávky*
Hlavní tok:	1. Systém provede přihlášení k prodejnímu portálu. 2. Systém provede dotaz na nové objednávky a data zákazníků. 3. Prodejní portál vrátí informace o nově přijatých objednávkách a zákaznících. 4. Systém vloží přijatá data do databáze. 5. Systém zobrazí seznam evidovaných objednávek.
Alternativní toky:	Žádné
Výjimky:	Storno operace Selhání přístupu do databáze Selhání komunikace s prodejním portálem
Frekvence:	Zpravidla denně, dle potřeby
Speciální požadavky:	Žádné

Tabulka 4.1: Specifikace případu užití pro import objednávek

Jak už bylo zmíněno v odstavci výše, při případu užití může docházet k chybám. Některé možné chyby si popíšeme výjimkami z případu užití a to rovněž v konvenci jazyka UML.

ID:	1.E.1
Název:	Stažení nových objednávek z prodejního portálu: Storno operace
Popis:	Prodejce ukončí případ použití
Primární účastníci:	Prodejce
Sekundární účastníci:	Žádní
Předpoklady:	Žádné
Následné podmínky:	1. Databáze systému zůstává v původním stavu 2. Seznam nových objednávek není zobrazen
Akce pro spuštění:	Prodejce zvolí v hlavním toku případu užití *Zrušit*

Tok:	Systém se vrátí do stavu před spuštěním případu užití
Frekvence:	Zřídka

Tabulka 4.2: Výjimka z případu užití 1

ID:	**1.E.2**
Název:	Stažení nových objednávek z prodejního portálu: Selhání přístupu do databáze
Popis:	Systém neotevře spojení s databází nebo spojení ztratí
Primární účastníci:	Prodejce
Sekundární účastníci:	Žádní
Předpoklady:	Při přístupu k databázi došlo k chybě
Následné podmínky:	1. Nové objednávky nebyly staženy 2. Seznam nových objednávek není zobrazen
Akce pro spuštění:	Selhání přístupu do databáze ve 4. nebo 5. kroku hlavního toku případu užití.
Tok:	1. Systém informuje prodejce o selhání přístupu do databáze 2. Systém se vrátí do stavu před spuštěním případu užití
Frekvence:	Zřídka

Tabulka 4.3: Výjimka z případu užití 2

ID:	**1.E.3**
Název:	Stažení nových objednávek z prodejního portálu: Selhání komunikace s prodejním portálem
Popis:	Systém se nedokáže spojit s prodejním portálem nebo obdrží chybnou odpověď
Primární účastníci:	Prodejce
Sekundární účastníci:	Prodejní portál
Předpoklady:	Pří komunikaci s prodejním portálem došlo k chybě
Následné podmínky:	1. Nové objednávky nebyly staženy 2. Seznam nových objednávek není zobrazen
Akce pro spuštění:	Selhání komunikace s prodejním portálem
Tok:	1. Systém informuje prodejce o selhání přístupu do databáze 2. Systém se vrátí do stavu před spuštěním případu užití
Frekvence:	Zřídka

Tabulka 4.4: Výjimka z případu užití 3

4.2.7 Import bankovních plateb

Import bankovních plateb, stejně jako import objednávek, spočívá v napojení navrhovaného systému na jiný systém třetí strany. V tomto případě to bude internetové bankovnictví některého z vybraných bankovních subjektů. Napojení na systém internetového bankovnictví bude spočívat ve využití API bankovního systému a stažení příchozích plateb.

Systém provede připojení k systému internetového bankovnictví a provede přihlášení uživatele, dále z vlastní databáze načte identifikátor poslední importované platby a skrze API vyhledá platby novější než je poslední zaznamenaná. Informace o platbách uloží do vlastní databáze a bezpečně se odhlásí ze systému elektronického bankovnictví. Z bezpečnostních důvodů se v navrhovaném informačním systému nebude ukládat heslo pro přihlášení do bankovnictví. Při každém importu je uživatel ručně zadá.

Importované platby se po importu automaticky spárují s přijatými objednávkami. Pro párování bude použitý variabilní symbol platby, jméno odesílatele nebo zpráva pro příjemce platby. V systému bude dále přístupná funkce pro zpracování objednávek na základě stavu zaplacení. Jakmile budou k objednávce připárovány platby s celkovou hodnotou stejnou nebo vyšší než je celková cena objednávky, objednávka může změnit svůj stav na *K odeslání* nebo podobný, dle definice uživatele.

4.2.8 Uživatelé systému

Aby navrhovaný informační systém byl pro prodejce vůbec použitelný, je samozřejmě nezbytné mít možnost registrovat do systému nové prodejce. Účet prodejce v naší webové aplikaci bude dvojího druhu, a to účet v bezplatné zkušební verzi, nebo účet placený.

Bezplatná verze aplikace bude pro uživatele omezena pouze časově, a to pravděpodobně na 30 dnů. Po tuto dobu nebude fungování aplikace nijak omezeno a prodejce si tak bude moci vyzkoušet všechny aspekty programu tak, jako by jej používal v jeho placené verzi. Po uplynutí této doby bude uživatel při přihlášení do systému vyzván k přechodu na placenou verzi aplikace. K registraci uživatele ve zkušební době nebude nutné uvádět žádné kontaktní údaje, pouze emailovou adresu, heslo a případně jméno, které se bude zobrazovat uživateli při přihlášení.

Placená verze aplikace bude pro uživatele zatížena měsíčními poplatky ve výši přibližně 100 Kč měsíčně. Aplikaci si bude možné předplatit volitelně na 3, 6 nebo 12 měsíců dopředu a po tuto dobu bude aplikace poskytovat plnou funkčnost. Při přechodu z bezplatné verze programu nebo při registrování nového platícího zákazníka bude kromě emailové adresy a hesla vyžadována také jeho fakturační adresa a telefonní kontakt.

Předmětem této práce bude pouze implementace systému v plném rozsahu tak, jak by ji používal uživatel v plné verzi aplikace. Vytvoření bezplatné zkušební doby je jedním z cílů v budoucím rozvoji projektu.

4.3 Databáze

V kapitole 3 – Technické prostředky bylo zmíněno, že jako databázový backend poslouží volně dostupný databázový server MySQL. Jedná se o klasickou relační databázi. V příloze 1 této práce nalezneme navržený ER diagram pro náš informační systém. V této části si pojďme ve stručnosti představit jednotlivé entitní množiny. Jednotlivé atributy nebudeme vypisovat, jsou uvedené v ER diagramu a popsané v podkapitolách 4.2.

Název entitní množiny	Popis
BANK_ACCOUNT	Reprezentuje evidované bankovní účty, Na bankovní účty se vážou bankovní platby a objednávky.
BANK_PAYMENT	Bankovní platby k evidovaným objednávkám. Do této tabulky jsou ukládány automatickým importem.
BANK_PAYMENT_STATUS	Tabulka obsahující výčet možných stavů bankovní platby. Budou zde přítomny stavy jako *Přijata*, *Spárována s objednávkou*, *Nespárována*, atd.
BUY	Tabulka jednotlivé nákupy u dodavatelů evidované prodejcem.
CUSTOMER	Reprezentace evidovaných zákazníků.
ORDER	Reprezentuje objednávky evidované v našem informačním systému, je základem celé aplikace.
ORDER_DELIVERY	Obsahuje výčet možných způsobů dopravy objednaného zboží (*Poštou*, *Kurýrem*, *Osobní převzetí*, …)
ORDER_HISTORY	Obsahuje záznamy o změnách v objednávkách, například změna stavu, přidání čísla zásilky, atd.
ORDER_PAYMENT	Obsahuje výčet možných způsobů platby objednaného zboží (*Předem*, *Dobírkou*, *Hotově*, …)
ORDER_STATUS	Obsahuje výčet možných stavů objednávky (*Přijata*, *K odeslání*, *Odeslána*, …). Obsahuje parametry jednotlivých stavů. Dle parametrů se bude řídit import objednávek i import plateb.
PACKAGE	Nese informace o evidovaných zásilkách k objednávkám a reklamacím.
PRODUCT	Představuje evidované produkty v nabídce prodejce.
PRODUCT_BUY	Vazební tabulka mezi předměty a nákupy, řeší M:M problém a reprezentuje v podstatě položky objednávek.
PRODUCT_ORDER	Vazební tabulka mezi předměty a objednávkami, podobně jako PRODUCT_BUY.
SERVIS_CUSTOMER	Reprezentuje evidované reklamace zákazníků.
SERVIS_CUSTOMER_BULK	Sdružuje reklamace zákazníků, čímž umožňuje reklamovat více kusů zboží najednou.
SERVIS_CUSTOMER_STATUS	Obsahuje možné stavy reklamací zákazníků. Stejně jako u objednávek jsou tyto stavy editovatelné.
SERVIS_DEALER	Reprezentuje evidované reklamace u dodavatelů.
SERVIS_DEALER_BULK	Podobně jako SERVIS_CUSTOMER_BULK dovoluje sdružovat reklamace u dodavatelů, čímž rovněž poskytuje možnost reklamovat více kusů zboží najednou.
SERVIS_DEALER_STATUS	Obsahuje možné stavy reklamací u dodavatelů, stavy budou opět uživatelsky editovatelné.
SETTING	Tabulka obsahující pouze jediný řádek a nesoucí uživatelské nastavení aplikace. Data v ní budou určovat parametry importu dat z prodejních portálů, nastavení prodejce, atd.
TEMPLATE_EMAIL	V této tabulce budou uloženy šablony emailů zasílaných zákazníkům při odeslání zboží, vyřízení reklamace, atd.

TEMPLATE_PRINT	V této tabulce budou uloženy šablony tiskových formulářů, na jejichž základě se budou tisknout balíkové štítky k zásilkám, reklamační protokoly, souhrny objednávek, atd.
VIEW	Tabulka VIEW je zvláštní tabulka, která slouží pro ukládání nastavení pohledů. Pokud například prodejce vyfiltruje zobrazení objednávek pouze na ty se stavem *Přijata*, tato informace se uloží do této tabulky a při příštím spuštění aplikace bude filtr stále aktivní.
VIEW_COLUMN	Obsahuje uživatelské nastavení sloupců v pohledech. Jak bylo napsáno výše, sloupce v pohledech budou uživatelsky přizpůsobitelné, skrývatelné, atd. Pořadí, atribut viditelnosti a uživatelsky nastavená šířka sloupce bude uložena v této tabulce.

Tabulka 4.5: Popis entitních množin

V souvislosti na napojení systému na prodejní portál Aukro.cz budou v databázi modelovány také následující entitní množiny určené k evidenci dodatečných dat právě z tohoto portálu.

Název entitní množiny	Popis
SHOP_AUKRO_ACCOUNT	Obsahuje uživatelem definované účty na portálu Aukro.cz.
SHOP_AUKRO_CUSTOMER	Informace o uživatelích prodejního portálu Aukro.cz, tedy zákaznících.
SHOP_AUKRO_DEAL	Reprezentuje jednotlivé nákupy provedené kupujícími na Aukro.cz.
SHOP_AUKRO_ITEM	Ukládá informace o nabídkách, které uživatel systému (prodejce) zveřejnil na portálu Aukro.cz. V těchto nabídkách mohou následně zákazníci vytvářet objednávky.
SHOP_AUKRO_PAYMENT	Eviduje platby uskutečněné přes platební systém PayU, používaný portálem Aukro.cz.
SHOP_AUKRO_SELL	Obsahuje jednoduché informace o počtech kusů zboží aktuálně vystaveného v nabídkách na Aukro.cz.
SHOP_AUKRO_SETTING	Podobně jako tabulka SETTING obsahuje tato tabulka také pouze jediný řádek s nastavením importu objednávek a exportu zboží na Aukro.cz.

Tabulka 4.6: Popis entitních množin pro práci s Aukro.cz

Z navržených entitních množin je patrné, že žádná z nich nenese informaci o uživatelích systému. Je to z toho důvodu, že data o uživatelích (prodejcích) budou uložena ve zcela oddělené databázi a také pro data každého uživatele bude vyhrazena speciální databáze.

Řešení v podobě zvláštní databáze pro každého uživatele bylo zvoleno z následujících důvodů. Oproti řešení, kde by byla data všech uživatelů uložena v jedné databázi, tento přístup výrazně zjednodušuje návrh databáze a její výsledné schéma. V případě jedné databáze by pro každou její tabulku bylo nutné specifikovat atribut identifikující uživatele, kterému patří daný záznam, a to

společně s cizím klíčem realizujícím vazbu na tabulku uživatelů. Dalším a hlavním důvodem je budoucí napojení na data ve webové aplikaci prostřednictvím desktopové aplikace.

V případě, že uživatel (prodejce) bude mít zájem o přístup k datům skrze externí aplikace, tuto možnost si bude moci nastavení explicitně povolit. Jejím povolením dojde k vytvoření nového databázového uživatele, jemuž bude povolen přístup z veřejné sítě a jemuž bude přiděleno oprávnění nakládat pouze s vlastní databází. Skrze tohoto uživatele se následně prodejce může do systému připojit prostřednictvím desktopové aplikace. Tímto bude prodejce odstíněn od dat jiných uživatelů webové aplikace, což primárně zvýší bezpečnost uložených informací.

4.4 Diagram tříd

V příloze 2 nalezneme konceptuální diagram tříd, které budou přítomny v kódu naší aplikace. Z důvodu přehlednosti byl diagram tříd rozdělen na několik částí popisujících funkční celky aplikace jednotlivě. Navržené třídy budou i díky technologii Entity Framework velice podobné navrženým entitním množinám v databázi. Přesto však budou obsahovat několik změn a rozšíření.

V prvé řadě je to třída *Payment*, ze které je teprve zděděna třída *BankPayment* odpovídající stejnojmenné tabulce v databázi. Tato třída je v aplikaci přítomna z důvodu plánovaného napojení na různé platební brány, elektronické peněženky, atd. Každá nově implementovaná platební metoda poté bude implementovat třídu zděděnou právě z této mateřské třídy Payment.

Další změnou je rozhraní *BankImport* a jeho realizace nazvaná *FioBankImport*. Tato třída slouží k importu plateb ze systému internetového bankovnictví, konkrétně pro FIO Banku.

Poslední změnou je přidané rozhraní *Shop*, jehož realizace nám reprezentuje prodejní portál. Pomocí dvou metod *ExportProducts()* a *GetNewOrders()* bude možné exportovat produkty z nabídky na prodejní portál, respektive stahovat z portálu nové objednávky. V tomto projektu je implementována jedna jeho realizace a tou je třída *ShopAukro*, která reprezentuje prodejní portál Aukro.cz.

Diagram tříd reprezentuje pouze doménový model navrhované aplikace, nezaměřuje se na třídy spojené s obsluhou transakcí, zobrazení atd.

Kapitola 5

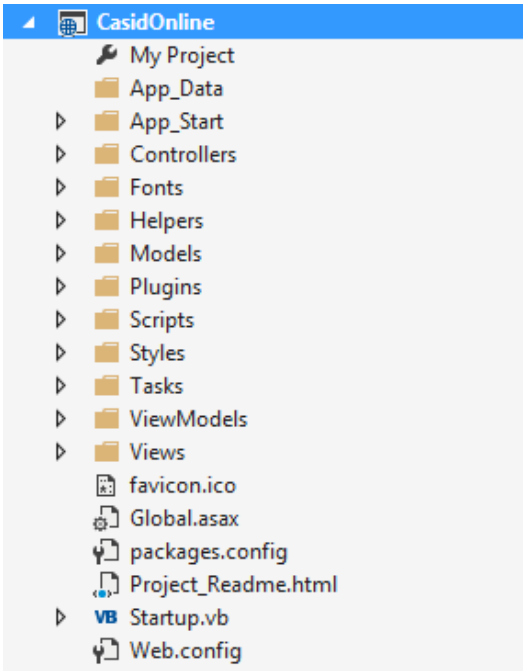
Implementace

Kapitola pátá s názvem Implementace nám popíše některé zajímavé pasáže implementace systému. V kapitole si nejprve probereme strukturu projektu, dále se zaměříme na způsob implementace grafického rozhraní, přičemž si představíme použitý framework a jeho rozšíření. Následovat bude podkapitola o implementaci správy uživatelů a rovněž představení frameworku řešícího tuto problematiku. Dále si ukážeme, jak jsou v systému obecně řešeny jednotlivé agendy a představíme si také přehled pro prodejce, tzv. *dashboard*. Poslední podkapitola se bude věnovat některým zajímavým a složitějším částem implementace. Nelze popsat veškeré implementované detaily, proto se zaměříme výhradně na tyto zajímavě řešené úseky. Jejich příkladem může být implementace úloh běžících na pozadí, řešení problému tisku prostřednictvím jednoúčelové aplikace na klientovi nebo generování emailových zpráv a popisů zboží na základě textových šablon.

5.1 Struktura projektu

Pro implementaci systému byla zvolena technologie ASP.NET MVC, jejíž použití již také předurčuje implementaci projektu v prostředí Microsoft Visual Studio, konkrétně ve verzi 2013. Toto vývojové prostředí je se samotnou technologií ASP.NET velice úzce provázáno, z čehož lze při samotné implementaci vytěžit mnoho výhod. Již samotné založení nového ASP.NET MVC projektu nám vytvoří základní strukturu složek a vzorových modelů, pohledů a kontrolérů. Strukturu projektu je samozřejmě možné libovolně modifikovat, nicméně pokud se jí rozhodneme držet, Visual Studio a samotné ASP.NET nám některé další činnosti výrazně usnadní. Příkladem může být široké použití reflexe [16], jmenovitě například vyhledání požadovaného kontroléru ve složce *Controllers* dle adresy požadavku nebo vyhledání pohledu dle názvu kontroléru a zvolené akce ve složce Views bez nutnosti explicitně zadávat plnou cestu.

Celé řešení (anglicky *solution*, uvedené v názvosloví Visual Studia [6]) se skládá ze dvou projektů. Jsou jimi hlavní projekt s názvem *CasidOnline* obsahující samotnou webovou aplikaci a projekt nazvaný *Model* obsahující business logiku systému. Toto oddělení obou součástí nám zajistí například možnost změnit pouze jednu část bez nutnosti kompilace kompletního kódu. Hlavním důvodem je ovšem sdílení aplikační logiky mezi webovou a existující desktopovou verzí aplikace. V budoucnu nám tento přístup usnadní souběžný vývoj obou projektů. Při vývoji této diplomové práce se obecně snažím využívat sjednocovat aplikační logiku napříč webovou i desktopovou verzí, byť to z počátku vyžaduje poměrně velké úsilí a výrazné úpravy na obou stranách. Na obrázku 5.1 nalezneme strukturu hlavního projektu v prostředí Visual Studia 2013.



Obrázek 5.1: Struktura projektu

V následující tabulce si stručně popíšeme všechny součásti projektu v pořadí, v jakém je vidíme na obrázku výše.

Název součásti	Popis
My Project	Konfigurační soubory projektu.
App_Data	Složka obsahuje databázi uživatelů systému (prodejců).
App_Start	Složka s konfiguračními skripty aplikace, spouštěnými po startu webového serveru.
Controllers	Složka obsahující třídy kontrolérů v našem systému. V této složce se implicitně hledají kontroléry při příjmu požadavku.
Fonts	Složka obsahující vektorové definice fontů použitých v grafickém rozhraní aplikace
Helpers	Speciální třídy určené ke generování částí HTML kódu v pohledech [17].
Models	Třídy modelu, které se nenacházejí v odděleném projektu Model. Jsou to například třídy reprezentující uživatele aplikace.
Plugins	Především rozšíření klientské části aplikace napsaná v jazyce JavaScript.
Scripts	Knihovny JavaScriptu vytvořené pro klientskou část aplikace.
Styles	Sada CSS stylů pro definici celkového vzhledu aplikace.
Tasks	Obsahuje třídy reprezentující úlohy zpracovávané na pozadí systému. Příkladem je import objednávek, odesílání emailů nebo inicializace databáze.
ViewModels	Třídy tzv. ViewModelů, jejichž instance reprezentují data zobrazovaná uživateli na konkrétní webové stránce, viz [18].
Views	Složka obsahující pohledy (z terminologie MVC), tedy HTML kód zobrazovaný uživateli.

favicon.ico	Ikona aplikace.
Global.asax	Hlavní třída ASP.NET aplikace, zde probíhá volání inicializačních skriptů.
package.config	Konfigurační souboru pro NuGET balíčkovací systém také používaný v tomto projektu.
Startup.vb	Konfigurační skript pro autentizaci uživatelů systémem ASP.NET Identity.
Web.config	Konfigurační soubor webové aplikace.

Tabulka 5.1: Struktura projektu

5.2 Vzhled aplikace

V širokém spektru internetových aplikací současného internetového prostředí je vzhled aplikace jedním z hlavních atributů, který může zákazníka přilákat a stejně tak dobře odradit od používání aplikace. Nyní však k samotné implementaci. ASP.NET MVC od základů změnilo způsob, jakým jsou webové stránky uživateli generovány. Dřívější technologie WebForms obsahovala spoustu předpřipravených grafických komponent, které bylo možné jednoduše v grafickém editoru naskládat na webovou stránku. Všechny komponenty se až při požadavku na server konvertovaly do odpovídajícího HTML kódu a příslušných knihoven JavaScriptu a kódu v CSS. Toto řešení přinášelo velice rychlou tvorbu bohatého uživatelského rozhraní, které však bylo velmi těžké nějakým způsobem programově graficky přizpůsobit. Podobně je na tom nyní například technologie Java Server Faces z balíku Java EE.

ASP.NET MVC se od tohoto způsobu odprošťuje a vrací se k základům tvorby webových stránek. Webová stránka, neboli pohled v terminologii MVC, je zpravidla čistý HTML kód. Dynamické chování webových stránek na straně klienta pak zajišťují podpůrné knihovny JavaScriptu a vzhled definují CSS styly, které si programátor sám zvolí. Toto řešení je sice náročnější a zdlouhavější na programování, nicméně zaručuje mnohem větší průhlednost kódu a prakticky neomezené možnosti grafického přizpůsobení stylu aplikace.

Tvořit dnes celé grafické rozhraní svépomocí by však vyžadovalo nemalé úsilí a pro rozsáhlejší projekty stovky hodin práce. ASP.NET MVC se s tímto vypořádává díky podpoře použití různých grafických webových frameworků. Jedním z takových frameworků je například *Bootstrap* [19] používaný i v tomto projektu.

5.2.1 Bootstrap a jQuery

Bootstrap je frameworkem určeným k tvorbě klientské části webové aplikace. Jedná se o knihovnu JavaScriptu a CSS stylů pro vytváření uživatelského rozhraní aplikace. Bootstrap je úzce provázaný se samotným HTML kódem webové stránky, kdy vzhled jednotlivých komponent se určuje na základě atributů *class* jednotlivých HTML elementů. Obrovskou výhodou v použití právě tohoto frameworku je důraz na mobilní zobrazení aplikace. Pomocí frameworku je možné definovat vzhled pro různé šířky webového prohlížeče a tak webovou stránku přizpůsobovat pro zobrazení

počítači, tabletu či mobilním telefonu. To celé v rámci jedné webové stránky bez nutnosti interakce s webovým server.

Další podstatnou součástí projektu je knihovna jQuery, která zastává spoustu rutinních činností při programování klientské části aplikace v jazyce JavaScript. Exemplárním příkladem pomoci knihovny je například získávání dat ze serveru pomocí technologie AJAX. Namísto nutnosti konstruovat objekt *XmlHttpRequest* [20], předávat mu všechny potřebné parametry a ošetřovat všechny stavy, které mohou nastat v průběhu zpracování požadavku, použijeme knihovní funkci *$.ajax().* Parametrem pouze předáme adresu, parametry požadavku a definujeme chování v případě úspěchu a neúspěchu. Samotná práce se nám výrazně zjednoduší. Nad zmiňovanou knihovní funkcí je v tomto projektu implementována knihovna *Task.js*, sloužící pro zpracovávání akcí běžících na pozadí. Tuto knihovnu si blíže představíme v kapitole 5.6.1, jelikož se jedná o jednu z velice zajímavých částí projektu.

5.2.2 Použitá rozšíření

Bootstrap framework nám sám o sobě poskytne bohatý nástroj pro vytvoření kompletního vzhledu aplikace. Pokud však půjdeme hlouběji do funkčnosti klientské části aplikace, narazíme na problémy, jejichž řešení nám samotný Bootstrap nenabídne. V této situaci připadá v úvahu použití rozšíření třetích stran, které nám klientskou část aplikace o požadovanou funkčnost doplní. Funkčnost, kterou nám samotný Bootstrap framework zajistit nedokáže, jsou například tabulky podporující řazení a filtrování, textové pole pro výběr data a další. V této podkapitole si popíšeme rozšíření, která jsou v projektu použita.

5.2.2.1 Bootstrap Table

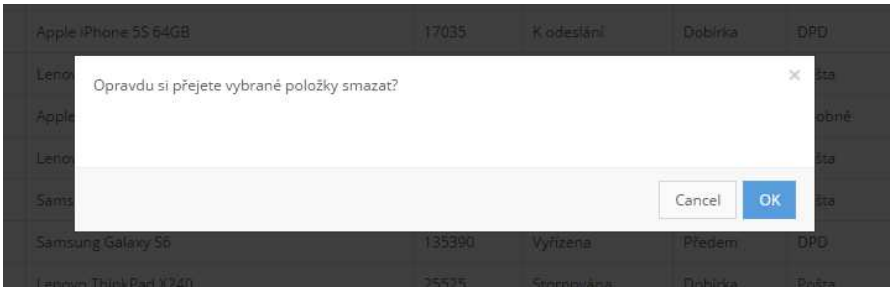
Ač se může zdát, že se jedná o součást výše zmiňovaného Bootstrap frameworku, opak je pravdou. Projekt Bootstrap Table je zcela oddělenou součástí, pouze určenou pro použití se zmiňovaným frameworkem.

Jedná se o rozšíření pro klientskou část webové aplikace, které dodává dynamičnost běžným HTML tabulkám. Rozšíření se skládá z JavaScript knihovny a sady CSS stylů, které nad běžnými tabulkami implementují tuto funkčnost. Výsledné řešení přidává do tabulek možnost řadit zobrazená data, filtrovat je pomocí textového vstupu od uživatele, stránkovat tabulky s mnoha řádky, zobrazovat nebo skrývat sloupce a formátovat v nich zobrazené údaje. Podstatnou součástí je možnost získávat data na žádost uživatele pomocí technologie AJAX nebo vybírat řádky pomocí zaškrtávacích polí. [21]

Tato funkce je v projektu využita při zobrazování všech tabulkových výpisů položek jednotlivých agend. Více o implementaci výpisů v agendách se dočteme v kapitole 5.4.

5.2.2.2 Bootbox

Bootbox rozšiřuje front-end webové aplikace o zobrazování jednoduchých dialogových oken uživateli. Již samotný Bootstrap dovoluje zobrazování dialogových oken, toto rozšíření se však zaměřuje na zobrazování zcela jednoduchých informativních nebo dotazovacích dialogů. Takovéto dialogy je pak možné zobrazit jediným voláním funkce v JavaScriptu, bez nutnosti definovat další HTML kód. [22]



Obrázek 5.2: Vzhled dialogového okna rozšíření Bootbox

Bootbox dialogová okna jsou v projektu použita například při potvrzení smazání položky, při požadavku na vstup od uživatele v průběhu importu dat z prodejního portálu a podobně.

5.2.2.3 Pace

Rozšíření s názvem Pace přidává funkčnost, která jen a pouze zlepšuje uživatelský zážitek z aplikace. Funkčnost tohoto rozšíření spočívá v zobrazení nahrávacího dialogu uživateli po dobu, kdy se součásti aplikace teprve načítají do webového prohlížeče.

Většina internetových prohlížečů ve snaze působit co nejvíce svižně zobrazuje uživateli webové stránky již v průběhu jejich načítání. U složitějších aplikací to má za následek postupné naplňování zobrazené stránky uživatelskými prvky a jejich následné automatické přemisťování či transformování dle aktuálně načtených CSS stylů a knihoven JavaScriptu. Toto chování webové stránky ne příliš přidává na uživatelské přívětivosti aplikace, a proto existují rozšíření jako právě Pace, která načítání a transformace prvků stránky před uživatelem skryjí. [23]

Implementačně se prakticky jedná o malou knihovnu JavaScriptu a malé sady CSS stylů pro zobrazení průběhu načítání. Inkluze této knihovny do dokumentu stránky by měla být provedena co možná nejvýše v kódu HTML. Je to z toho důvodu, aby prohlížeč co nejdříve provedl její zpracování, zobrazil požadovaný indikátor průběhu a před uživatelem dočasně skryl prvky webové stránky.

5.2.2.4 Bootstrap Datepicker

Stejně jako Bootstrap Table není toto rozšíření součástí samotného frameworku Bootstrap. Jedná se o rozšíření HTML prvku input o zobrazení kalendáře po kliknutí pro výběr data. Opět jde o knihovnu JavaScriptu a CSS stylů, které zajišťují požadovanou funkčnost. [24] Výhodou tohoto řešení oproti konkurenčním je lokalizace výběrového prvku do mnoha světových jazyků, včetně češtiny.

Obrázek 5.3: Rozšíření Bootstrap DatePicker v zobrazení objednávky

V tomto projektu je prvek použit ve většině formulářů, kde se nachází vstupní pole pro zadávání data.

5.2.2.5 Morris.js

Rozšíření Morris.js je v projektu použité pouze na jediném místě a to v dashbordu, tedy na úvodní stránce. Toto rozšíření implementuje opět pomocí JavaScriptu a CSS zobrazení grafu. V dashboardu zobrazuje obraty, tedy sumu nákupů a objednávek za dané období.

Oproti jiným konkurenčním řešením vyniká knihovna Morris.js jednoduchostí zobrazených grafů a to s důrazem na grafické ztvárnění. Velice dobře tak zapadne do celkového grafického stylu aplikace. Umožňuje zobrazit klasické spojnicové grafy, ale také sloupcové či koláčové typy grafů. Zobrazované grafy mohou být responzivní, tedy reagující na pohyb myši nad nimi, a tím zobrazující dodatečné informace pro uživatele.

5.2.3 Grafická šablona

Po několika pokusech vytvořit design webové aplikace vlastními silami bylo přistoupeno na variantu použití některé z existujících grafických šablon. Na internetu lze najít mnoho povedených grafických šablon, které dodají aplikaci profesionální vzhled, jehož vytvoření v takovém rozsahu, jaký poskytuje hotová šablona, by vyžadovalo desítky až stovky hodin práce a zejména grafické tvůrčí nadání.

Grafické ztvárnění aplikace vychází webové šablony s názvem *Nifty* vybrané v internetovém katalogu *Bootswrap*, viz [25]. Tento a další internetové katalogy nabízejí stovky šablon vhodných

pro použití v tomto projektu. Takřka všechny nabízejí obecnou funkčnost jako klasické i mobilní zobrazení, integraci dynamických tabulek nebo grafické rozložení pro přihlašovací obrazovku. O finálním výběru šablony tedy zpravidla rozhodují detaily, které je třeba pečlivě zkoumat. Z celkem 78 prohlédnutých šablon napříč katalogy bylo vybráno 9 nejlepších kandidátů, mezi nimiž rozhodlo právě zkoumání jejich drobných odlišností. Následující tabulka srovnává 9 kandidátních šablon.

Dle srovnávací tabulky vycházejí nejlépe grafická témata s názvy *Nifty*, *BePro* a *Lander v2*. Konečný výběr tedy rozhodne především celkový dojem z grafického ztvárnění šablony a ten subjektivně nejlépe splňuje grafické téma Nifty.

	Light Blue	Ace Admin	Nifty	BePro	Lander v2	Prism	King Admin	Beyond	Cen-taurus
Export dat z tabulky	✗	✓	✓	✗	✓	✗	✗	✓	✗
Víceřádko-vý výběr v tabulce	✓	✓	✓	✓	✓	✗	✓	✗	✗
Scrollable dialogy	✓	✓	✓	✓	✓	✓	✓	✗	✓
Vyhledáva-cí box	✓	✓	✓	✓	✗	✓	✓	✓	✓
Nahrávací obrazovka	✓	✗	✓	✓	✓	✓	✗	✓	✗
Full-screen tabulky v mobilním zobrazení	✗	✗	✗	✓	✓	✗	✗	✓	✗
Zdroj	[1]	[1]	[2]	[2]	[3]	[3]	[1]	[1]	[1]

Tabulka 5.2: Porovnání grafických šablon

Zdroje:

- [1] http://www.wrapbootstrap.com/
- [2] http://www.bootswrap.com/
- [3] http://www.gridgum.com/

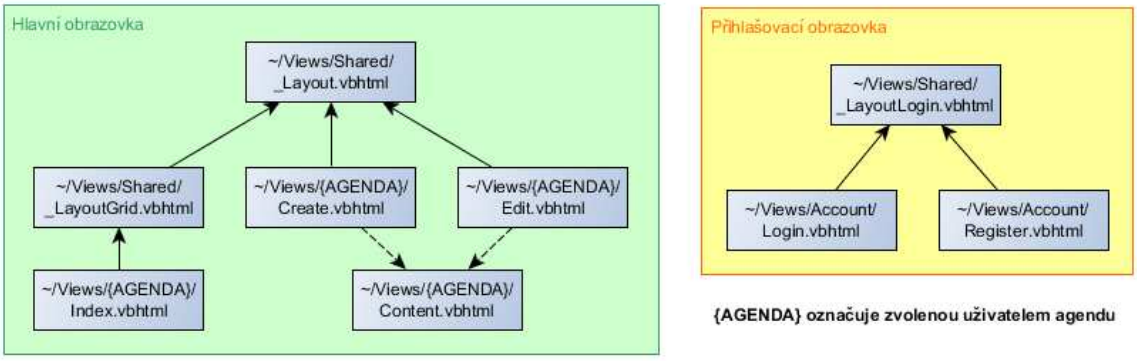
Grafická šablona je distribuovaná ve formě ZIP archivu, který obsahuje vzorové stránky v čistém HTML kódu a k nim stylové CSS předpisy a nezbytný JavaScript pro některé ovládací prvky. Jelikož se jedná o sadu statických webových stránek, vybraná šablona pomáhá pouze s grafickým ztvárněním aplikace. Veškerou logiku i konkrétní rozložení uživatelských prvků je třeba naprogramovat. Použití hotové grafické šablony bylo zvoleno jen a pouze za účelem dosažení

profesionálního vzhledu výsledného produktu, nikoli z důvodu usnadnění práce, jak by se na první pohled mohlo zdát.

Rozložení prvků ve stránce je lehce odlišné od původně navrhovaného schématu a to následujícím způsobem. Nabídka agend není umístěna v horní části okna aplikace, nýbrž po její levé straně. V mobilním zobrazení je nabídka agend zmenšená a na malých displejích úplně skrytá a dostupná po kliknutí na tlačítko, v poslední době známé jako *hamburger menu*. [26] Další změnou je nabídka pro jednotlivé agendy původně navrhovaná v levém postranním panelu a která se přesunula do hlavní středové části obrazovky. Logo aplikace zůstalo nezměněno v levém horním rohu. Navíc přibyla pravá postranní skrývatelná nabídka, která je momentálně nevyužitá. Její součástí by v budoucnu mohly být různé statistické informace specifické pro jednotlivé agendy. Příkladem může být zobrazení nákupní historie zákazníka při kliknutí na objednávku, procentuální poruchovost zboží při kliknutí na reklamaci, atd.

5.2.4 Struktura pohledů

Technologie ASP.NET MVC obsahuje funkci, která dovoluje generování pohledů, tedy webových stránek zobrazovaných uživateli, na základě tzv. *layoutů*. Layouty (opět z názvosloví ASP.NET MVC) nám definují grafický rámec aplikace, v jehož konkrétní definované části se posléze mění uživatelem požadovaný obsah. Layouty mohou být definovány v hierarchické struktuře, což je také případ naší aplikace. Hierarchii pohledů v aplikaci naznačuje obrázek 5.4.



Obrázek 5.4: Hierarchie pohledů

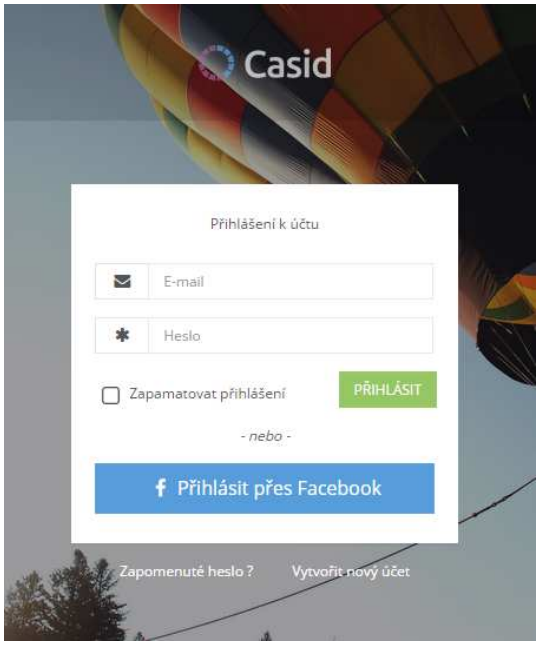
Jak je patrné z obrázku výše, v aplikaci jsou dva druhy grafického rozložení obrazovky. Jedno grafické rozložení je vytvořeno pro proces registrace a přihlašování do systému, druhé poté pro samotný vzhled aplikace.

Vzhled aplikace je dán rozložením v souboru *_Layout.vbhtml*, z něhož je dále odvozen pohled *_LayoutGrid.vbhtml*, který do původního layoutu přidává tabulkový výpis položek zvolené agendy. Pro každou agendu (tedy objednávky, zásilky, reklamace, atd.) je poté definován pouze jednoduchý pohled s názvem *Index.vbhtml*, který obsahuje pouze informace o tom, jaká data do pohledu vložit.

Toto řešení umožňuje vzhled měnit pouze na jediném místě pro všechny agendy, tím výrazně zlepšuje udržovatelnost kódu a bezezbytku naplňuje princip *DRY* v softwarovém inženýrství [27].

Z výchozího rozložení *_Layout.vbhtml* jsou odvozena také rozložení pro zobrazení položek jednotlivých agend, tedy rozložení pro zobrazení objednávky, zásilky, reklamace, atd. Pro každou agendu existuje zvláštní pohled určený pro vytváření nové položky v agendě, nazvaný *Index.vbhtml*, a pohled pro editaci existující položky nazvaný *Edit.vbhtml*. Oba tyto pohledy však využívají společnou část v podobě definice vstupních polí a jiného obsahu ze souboru *Content.vbhtml* a sami definují pouze specifické odlišnosti pro vytváření resp. upravování položek agendy.

Zvláštní rozložení je definováno pro přihlašovací a registrační obrazovku v systému. V rámci zjednodušení a zpřehlednění procesu registrace a přihlašování bylo vhodné vytvořit rozložení specifické pouze pro tyto činnosti. Rozložení reprezentuje soubor s názvem *_LayoutLogin.vbhtml*, ze kterého jsou odvozeny pohledy *Login.vbhtml* a *Register.vbhtml* pro přihlašování, resp. registraci do systému.



Obrázek 5.5: Přihlašovací obrazovka systému

5.3 Správa uživatelů

Správa a zabezpečené uživatelských účtů je kamenem úrazu mnoha volně dostupných informačních systémů. Je třeba zajistit nejen samotné zapsání uživatelů do databáze systému a následné ověřování přihlašovacích údajů, ale také zabezpečit celý přenos citlivých údajů od uživatele a jejich bezpečné uložení. Trendem poslední doby je poté přihlašování k webovým aplikacím skrze aplikace třetích stran, jako jsou například různé sociální sítě. Přihlašováním přes

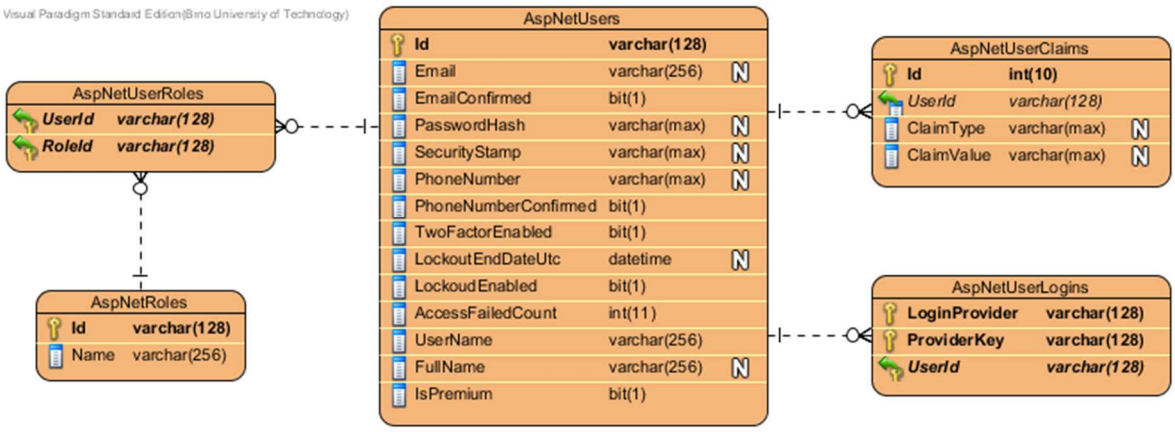
aplikace třetích stran vzniká v systému další potenciální slabina v zabezpečení, o níž se vývojář musí postarat.

Naštěstí však existují některá hotová komplexní řešení, která dokážou výše zmíněné problémy vyřešit takřka zcela ve své režii. Jedním z takových řešení je framework *ASP.NET Identity*, použitý i v tomto projektu.

5.3.1 ASP.NET Identity

ASP.NET Identity je nástupcem technologie *ASP.NET Membership* [28] z roku 2005, která sloužila ke správě uživatelů ve webových aplikacích založených na technologii WebForms. Postupem času a se stoupající provázaností internetových aplikací právě se sociálními sítěmi se tento framework stal limitujícím a společnost Microsoft přišla s řešením v podobě nového frameworku s názvem Identity. [32]

Základním prostředkem pro identifikaci uživatele samozřejmě zůstává uživatelské jméno a heslo volené uživatelem. Na tuto základní funkčnost nabaluje Identity framework právě onu potřebnou provázanost se sociálními sítěmi. Dalším inovativním prvkem je správa rolí, která umožňuje mít v systému uživatele s různými právy přístupu, v našem případě to budou uživatelé placené a volně dostupné verze. Framework používá technologii Entity Framework pro objektově-relační mapování a perzistenci uživatelů do relační databáze. Použití této technologie zajišťuje vysokou míru abstrakce nad fyzickou databází a tedy možnost zvolit takřka libovolnou relační databázi, kde budou informace o uživatelích uloženy. Schéma databáze je definováno ASP.NET Identity frameworkem. Schéma je možné libovolně rozšiřovat o další entitní množiny nebo atributy těch stávajících. Pro správnou funkci je však nezbytné původní entitní množiny a jejich atributy ponechat.



Obrázek 5.6: Schéma databáze pro správu uživatelů

Co se týká samotného zabezpečení, framework se stará o ukládání hesla v podobě hashe, podporuje ověření zadaných uživatelských údajů skrze SMS bránu či e-mail, implementuje protokol

OAuth pro přihlašování skrze služby třetích stran, blokování přístupu po několika neúspěšných přihlášeních, atd. [29]

Informace o uživatelích systému, tedy o prodejcích, jsou v systému uložena odděleně od uživatelských dat, je pro ně vyhrazena zvláštní databáze. V našem případě posloužila pro ukládání dat o uživatelích databáze *MSSQL Express*, standardně dodávaná s instalací Visual Studia. Výsledné schéma databáze pro správu uživatelů vidíme na obrázku 5.7.

Zabezpečení přenosu citlivých informací mezi webovým prohlížečem uživatele a informačním systémem zajišťuje šifrovaný SSL kanál.

5.3.2 Proces registrace uživatelů

Registrace uživatelů do systému probíhá skrze registrační formulář, v němž zájemce o používání programu vyplní svoji e-mailovou adresu a heslo. Volitelně může vyplnit také své jméno, které se následně zobrazuje v pravém horním rohu okna aplikace. Po odeslání registračního formuláře a validaci zadaných dat dojde k vytvoření databáze pro nově registrovaného uživatele. Databáze nese jméno ve tvaru *casid-user-GUID*, přičemž *GUID* reprezentuje unikátní identifikátor uživatele v systému. Během této doby je uživateli zobrazen vyčkávací dialog a po skončení akce je uživatel automaticky přesměrován do aplikace. Proč je vytvářena zvláštní databáze pro každého uživatele v systému, je objasněno v kapitole 4 – Návrh.

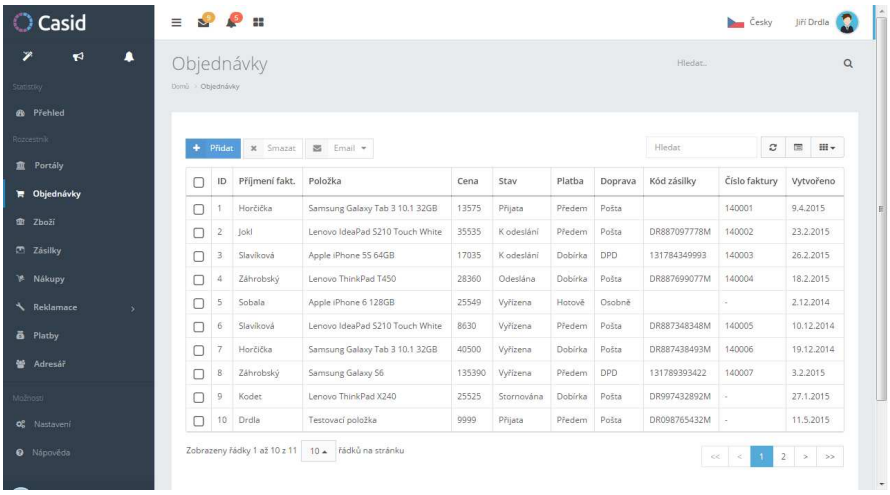
Obrázek 5.7: Uživatel přihlášený v systému

5.4 Agendy v systému

Položky jednotlivých agend evidovaných ve webové aplikaci, tj. objednávky, zásilky, reklamace, zboží, nákupy, zákazníci a platby, jsou implementovány velice podobným způsobem. To zajišťuje jednoduchost práce se systémem napříč agendami. Kliknutím na název agendy v levém postranním panelu se přechází na tabulkový výpis položek evidovaných v rámci této agendy. Tabulkový výpis je patrný z obrázku 5.8.

Podobný styl výpisu se používá v mnoha dalších podobných systémech a uživatelé s nimi pracující jsou na něj úspěšně přivyklí. V horní části prostoru pro tabulkový výpis je pro každou agendu uveden její název a pod ním navigační lišta, tzv. *breadcrumb*. [30] Největší část okna však vyplňuje zmiňovaný výpis položek. Tabulkový výpis zajišťuje webové rozšíření s názvem *Bootstrap Table*, více o tomto rozšíření je uvedeno v kapitole 5.2.2.1. Pro účely této aplikace bylo rozšíření lehce přizpůsobeno. Na řádky tabulky lze nyní kliknout a fungují jako odkaz na konkrétní objednávku. Řádky tabulky je možné označovat pomocí zaškrtávacích polí a pomocí horní tlačítkové

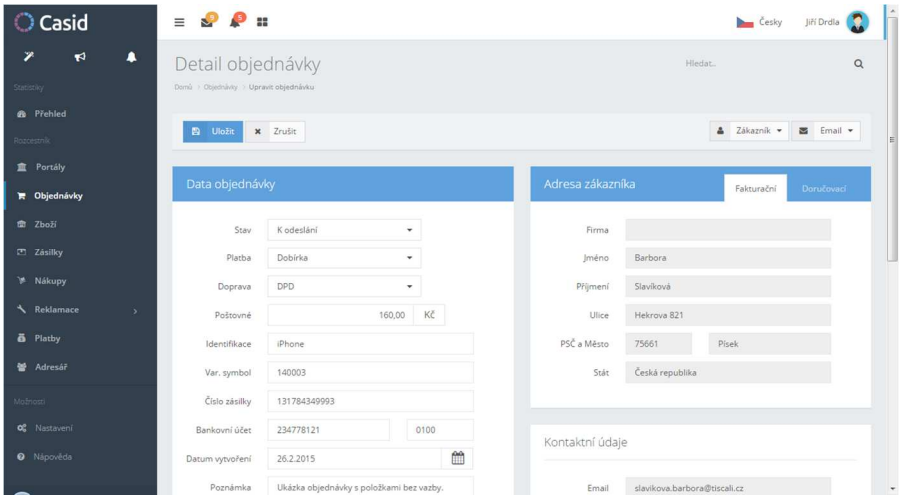
lišty s nimi hromadně pracovat. Pokud není označen žádný řádek, tlačítka v horní liště zůstávají neaktivní, což lépe informuje uživatele o aktuálně proveditelných akcích.



Obrázek 5.8: Tabulkový výpis položek agendy

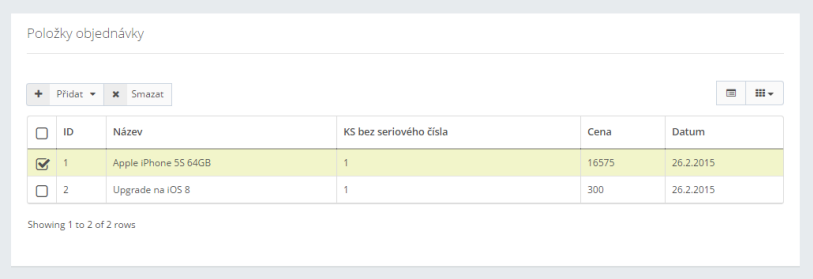
Data do tabulky jsou získávána pomocí technologie AJAX, nejsou tedy součástí webové stránky již při načtení. Tím je umožněno například provádět refresh dat v tabulce bez nutnosti nahrávat celou stránku.

Rozšíření Bootstrap Table nabízí mnoho funkcí, z nichž některé jsou použité i v tomto projektu. Zejména se jedná o vyhledávací pole, které na základě uživatelem zadaného textu vyhledává mezi aktuálně zobrazenými objednávkami. Dále je to možnost řádky tabulky konvertovat do seznamového zobrazení, které je zvláště vhodné pro malé displeje mobilních telefonů. V projektu byl použit i uživatelský výběr sloupců a stránkování tabulky.



Obrázek 5.9: Zobrazení položky agendy

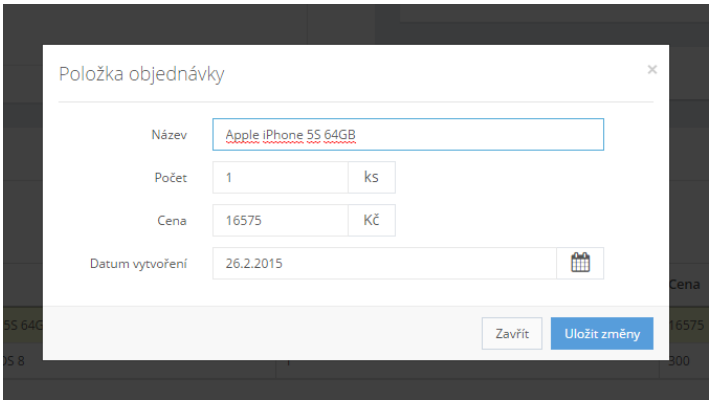
Kliknutím na řádek tabulky se přechází na zobrazení konkrétní položky, které můžeme vidět na obrázku 5.9. Zobrazení položky agendy využívá stejné rozložení jako tabulkový výpis, pouze se liší svým obsahem. Horní část středového prostoru opět obsahuje název aktuálně zobrazené položky a pod ním navigační lištu. Následuje lišta tlačítek pro práci s aktuálně zobrazenou položkou, přičemž v její levé části jsou umístěny volby pro uložení a návrat zpět, na pravé straně jsou to již akce pracující v dané položce.



Obrázek 5.10: Položky objednávky

Nyní si vezměme pro příklad objednávku, jakožto položku z agendy objednávek. Pro zobrazení objednávaného zboží je využita opět tabulka implementovaná rozšířením Bootstrap Table. Zajímavým způsobem bylo nutné vyřešit naplnění tabulky daty, tedy informacemi o objednaném zboží a rovněž předávání těchto informaci zpět serveru při uložení objednávky. Za tímto účelem je ve formuláři přítomno skryté textové pole, v němž jsou informace o všech objednaných položkách uloženy ve formátu JSON. Po načtení stránky se text uložený v tomto poli naparsuje do podoby kolekce objektů představujících objednané zboží a ta předá tabulce pro zobrazení. Při uložení objednávky, případně při jiné akci, se kolekce z tabulky zpátky serializuje do formátu JSON, uloží se do avizovaného skrytého pole a je odeslána jako standardní položka formuláře. Tímto řešením se také předchází problému s funkcí model binder, popsaném v kapitole 3.1.1.

Editace položek objednávky probíhá pouze na straně klienta, za pomocí modálních oken z Bootstrap frameworku. Na obrázku 5.10 vidíme výpis položek objednávky a na obrázku 5.11 příklad editace jedné položky.



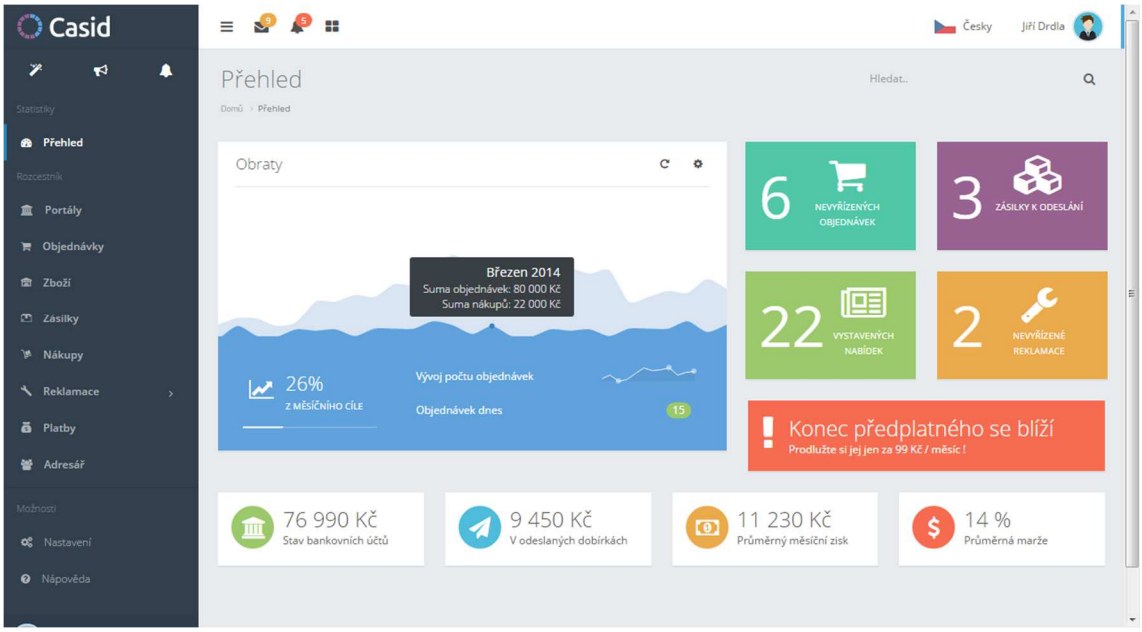
Obrázek 5.11: Editace položky objednávky

5.5 Dashboard

Jednou z obrazovek informačního systému je dashboard. Je to první obrazovka, kterou uživatel uvidí po vstupu do systému a slouží k poskytnutí základního přehledu o aktuálním stavu prodejů.

> *"A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance."* [34]

Jak už bylo zmíněno v návrhu, webová aplikace cílí především na malé internetové prodejce všech kategorií. Z tohoto důvodu by i dashboard měl být jednoduchý, na první pohled srozumitelný a měl by poskytovat pouze takovou škálu informací, ve které se lze rychle zorientovat. Ukázku dashboardu z implementované webové aplikace nalezneme na obrázku 5.12.



Obrázek 5.12: Dashboard

Hlavní část dashboardu zaujímá graf nákupů a prodejů v měsících za poslední rok. Pod ním lze nalézt vývoj počtu objednávek za stejné období a počet objednávek realizovaných zákazníky v aktuálním dni. K přehlednosti a čistotě celého řešení přispívá fakt, že konkrétní hodnoty se zobrazují po přejetí kurzoru myši. Vpravo od grafu jsou umístěny 4 widgety informující o nedokončených činnostech evidovaných v systému. Jsou to informace o počtu nevyřízených objednávek, počtu

neodeslaných zásilek, počtu aktuálně vystavených nabídek na prodejních portálech a počtu nevyřízených reklamací. Na widgety je možné kliknout, přičemž uživatel je přesměrován na výpis položek dané agendy.

Obsahem spodní části dashboardu je opět několik widgetů, tentokrát bez funkce odkazu, které dávají základní přehled o finanční stránce obchodů. Jejich obsahem je celkový stav bankovních účtů, celková suma objednávek odeslaných na dobírku, u nichž očekáváme přijetí dobírkové částky, dále průměrný zisk za měsíc a průměrná marže za poslední měsíc.

Oproti dashboardům jiných informačních systémů může tento dashboard působit poněkud stroze a dojmem, že poskytuje malé množství informací. Jednoduchost dashboardu je však primárním účelem.

5.6 Zajímavé části implementace

V této podkapitole si podrobněji popíšeme některé zajímavé části z implementace projektu.

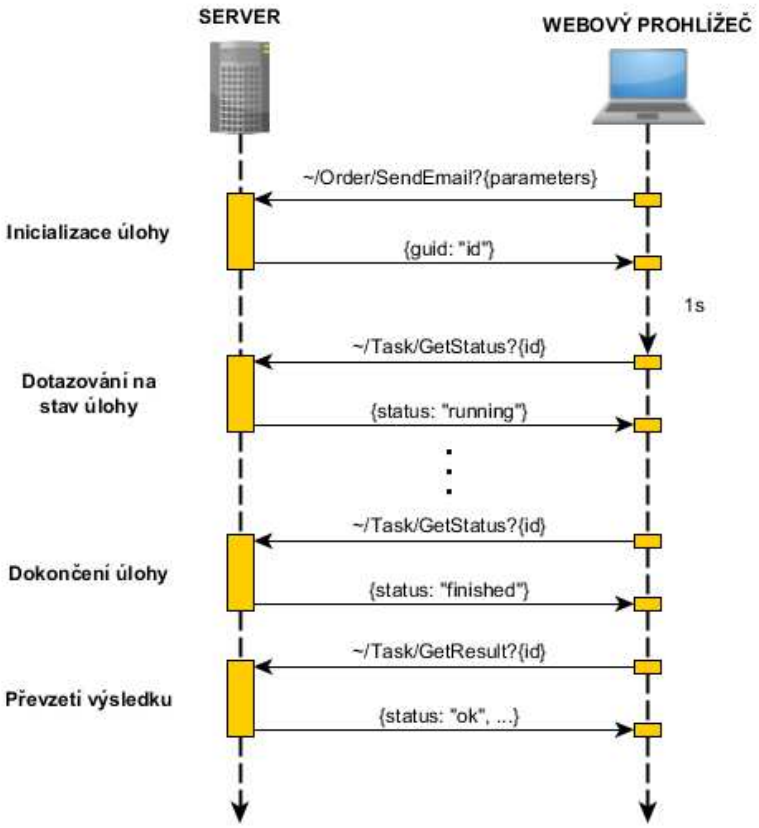
5.6.1 Akce vykonávané na pozadí

Akce jako import objednávek, export zboží nebo i odesílání emailů zákazníkům mohou trvat déle než je odezva systému na běžné požadavky a jejich sekvenční provádění by mohlo mít neblahý vliv na uživatelskou přívětivost systému. Z tohoto důvodu byl v aplikaci navržen a implementován systém pro zpracovávání akcí běžících na pozadí.

Tento systém dovoluje uživateli spustit požadovanou akci, zobrazuje mu její průběh a také informuje o dokončení akce. Specialitou je poté možnost vyžádat si od uživatele vstup v průběhu prováděné akce. Systém se skládá ze serverové a klientské části a blíže si jej popíšeme v této podkapitole.

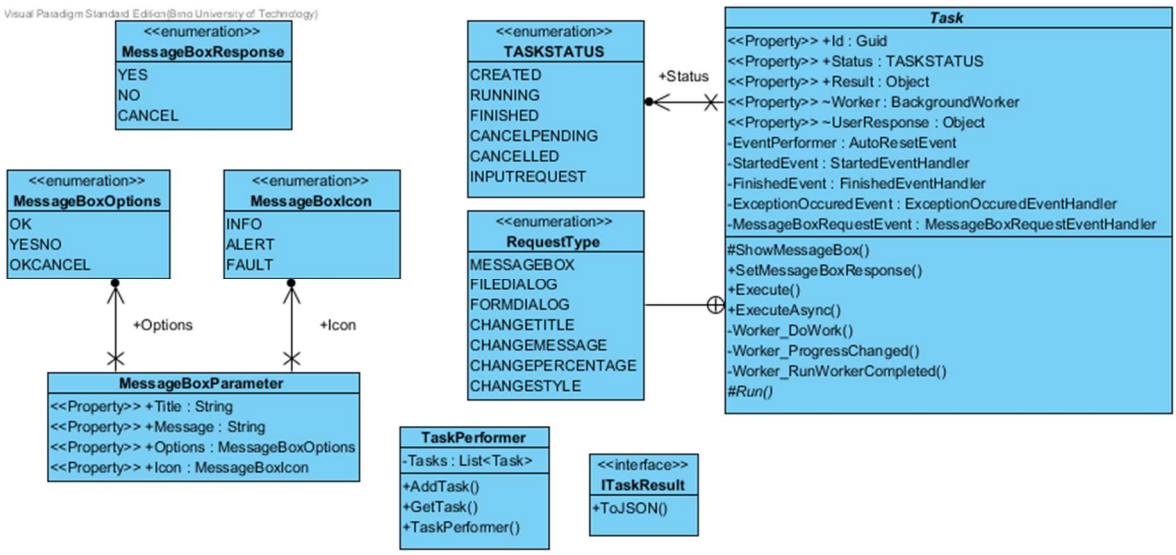
Klientská část systému pro vykonávání akcí na pozadí je knihovna *Tasks.js* napsaná v jazyce JavaScript a sada informačních dialogů v jazyce HTML využívající Bootstrap framework. Zmiňovaná knihovna implementuje speciální objekt *$task*, jehož atribut *start()* dovoluje zahájit akci vykonávanou na pozadí, přičemž zobrazí modální dialogové okno pro informování uživatele o průběhu. Samotná metoda vyžaduje jediný parametr a tím je objekt specifikující požadavek na prováděnou akci. Tento požadavek zahrnuje adresu pro spuštění úlohy, dále seznam parametrů zasílaných serveru. Jeho součástí je také definice obslužných rutin pro případ úspěchu i neúspěchu prováděné akce a také dvojice informačních textů, které se uživateli po dokončení akce zobrazí v závislosti na stavu dokončení úlohy.

Spuštění následně probíhá zasláním požadavku na server pomocí technologie AJAX. Následuje periodické dotazování serveru na stav úlohy a po jejím dokončení převzetí výsledku. Přenos dat probíhá ve formátu JSON. Následující obrázek ilustruje sekvenci zasílaných zpráv při provádění akce na pozadí včetně jejich konkrétní podoby.



Obrázek 5.13: Sekvenční diagram odesílání emailů

Na obrázku 5.13 vidíme příklad sekvenčního diagramu pro úlohu zasílání emailů. Každá úloha je inicializována na straně klienta akcí uživatele. S pomocí výše zmínění knihovny *Task.js* je vytvořen AJAXový požadavek na server a požadavek je odeslán. Mezitím je uživateli zobrazen vyčkávací dialog. Odpovědí serveru se zpráva ve formátu JSON obsahující unikátní identifikaci běžící úlohy. Každá úloha v systému je identifikována svým unikátním GUID, což je 32 znaků dlouhý řetězec zasílaný klientovi po vytvoření úlohy. Klient se následně v periodických intervalech dotazuje na stav úlohy a server mu tyto informace poskytuje. V odpovědi serveru může být kromě aktuálního stavu úlohy také například požadavek na změnu textu zobrazovaného uživateli v dialogovém okně, dále aktualizace progresu (pokud to úloha požaduje) nebo například požadavek na vstup od uživatele. Periodické dotazování na stav úlohy probíhá až do té doby, dokud odpověď ze serveru neobsahuje stav *finished* nebo *failed*. Stav *failed* indikuje, že úloha skončila předčasně vyvoláním výjimky. Tento stav by neměl v úlohách nastávat. Po skončení úlohy zašle knihovna *Task.js* požadavek na vrácení výsledku úlohy. Výsledek úlohy se opět předává ve formátu JSON a v případě zasílání emailů tento objekt obsahuje informaci, zda bylo odesílání úspěšné a pokud nikoli, obsahem je také důvod selhání. U emailů to může být chybně vyplněná emailová adresa, neodpovídající emailová šablona nebo nesprávné nastavení SMTP serveru.



Obrázek 5.14: Diagram tříd pro akce běžící na pozadí

Serverová část je kompletně implementována v kódu VB.NET. Diagram tříd, které implementují tuto funkčnost, najdeme na obrázku 5.14. Hlavní komponentou je abstraktní třída *Task*, jenž reprezentuje úlohu vykonávanou na pozadí. Samotný kód úlohy nese abstraktní metoda *Run()* a kód je vykonáván ve zvláštním vlákně, jehož vytvoření a správu zajišťuje knihovní objekt *BackgroundWorker*. [6] Třída umožňuje vznášet události, například v případě nutnosti informovat uživatele o průběhu akce, při odstartování nebo ukončení úlohy atd. Výsledkem prováděné akce je objekt třídy implementující rozhraní *ITaskResult*. Toto rozhraní obsahuje jedinou metodu s názvem *ToJSON()*, která zajišťuje zformátování výsledku do formátu JSON určeného pro zaslání zpět webovému prohlížeči.

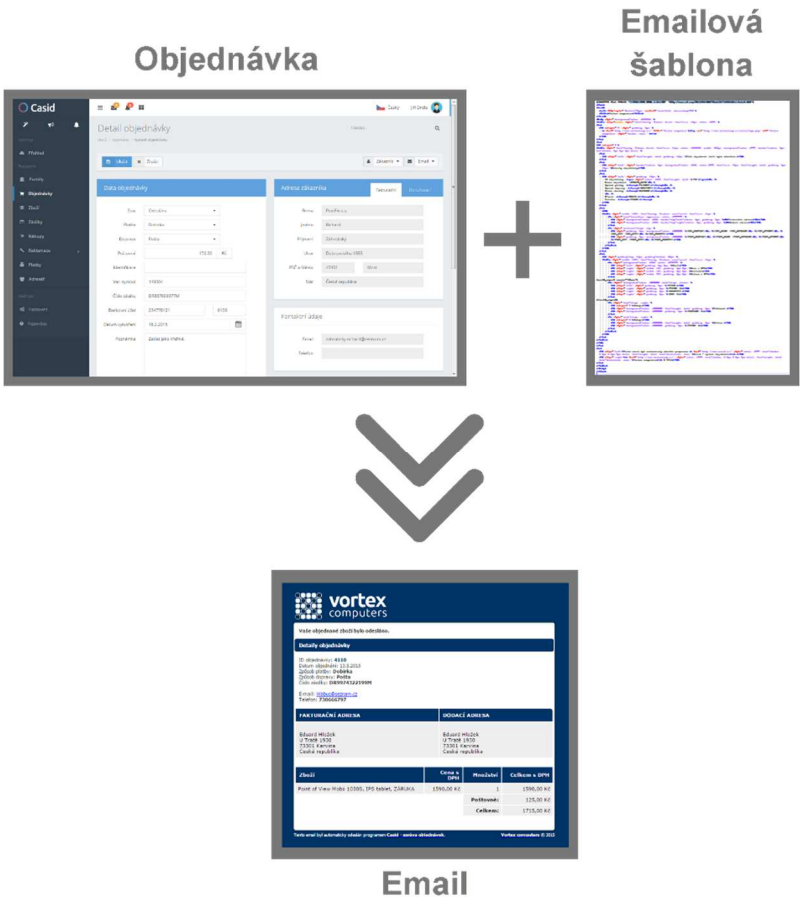
Správu všech úloh zajišťuje objekt třídy *TaskPerformer*. Tento objekt je v aplikaci přítomen pouze v jediné instanci a obsluhuje tak všechny úlohy všech uživatelů. Jeho úkolem je však pouze přijímat požadavky na nově zpracovávané úlohy a ty řadit do fronty a spouštět, dále na žádost klienta vracet stav požadované úlohy a v případě dokončení opět na žádost vrátit její výsledek a úlohu z fronty odstranit.

5.6.2 Generování emailových zpráv a nabídek zboží

Jednou z vlastností určené už při návrhu webové aplikace je dát prodejci možnost, aby svým zákazníkům zasílal e-mailové zprávy a to jak čistě textově, tak i ve formátu HTML. Další vlastností systému má být generování popisů evidovaného zboží na základě jeho parametrů. Obě tyto činnosti spolu zdánlivě nesouvisí, avšak opak je pravdou.

V obou případech jde o generování dokumentů ve formátu prostého textu nebo HTML na základě dat evidovaných v aplikaci. Pro tento účel byl v aplikaci navržen systém šablonování, který

nám dovoluje na základě zvolené šablony a evidované položky v rámci agendy vygenerovat prakticky libovolný textový dokument popisující danou položku. Třída položky, k níž je možné dokument takto generovat musí implementovat rozhraní *IExportable*. Agendy, jejichž položky toto kritérium splňují, jsou objednávky, zboží, reklamace a zásilky. Konkrétní podobu rozhraní *IExportable* nalezneme v diagramu tříd v příloze 2. Proces generování emailových správ je zobrazen na obrázku 5.15. Podobně probíhá proces generování popisů zboží.



Obrázek 5.15: Proces vytváření emailů

Aby bylo možné emaily na základě emailových šablon vytvářet, je zapotřebí v emailové šabloně definovat místa, do kterých dosadit požadovaný text z evidované objednávky, reklamace, atd. Místa pro vkládání označíme speciálními tagy. Tagy budou celkem třech druhů a všechny si je popíšeme v následujících podkapitolách. Tyto tagy řeší prakticky 3 problémy, které při generování textových zpráv na základě šablony nastávají. Těmito základními problémy jsou:

- Jak do šablony vložit konkrétní informaci?
- Jak určitou část textu šablony zopakovat?
- Jak určitou část textu šablony za určitých podmínek vynechat?

V příloze 4 poté najdeme ukázku celé emailové šablony, jak je použita v reálném nasazení. Nyní si představme jednotlivé tagy podrobněji.

5.6.2.1 Jednoduché tagy

Jednoduchými tagy se rozumí jednoduché textové hodnoty jako je například název zboží, cena objednávky, jméno zákazníka atd. Tyto tagy jsou vždy ohraničeny hranatými závorkami, například *[title]* pro název zboží nebo *[id]* pro číslo objednávky. Seznam veškerých tagů, použitelných pro jednotlivé agendy, najdeme v příloze číslo 3 této práce.

Následuje ukázka použití jednoduchých tagů na příkladu vkládání názvu zboží a jeho krátkého popisu při generování popisu ve formátu HTML.

```
...
  <div id="obsah00in">
    <h2>
      Prodám [title]
    </h2>
    <p>
      [parameters:short_description]
    </p>
...
```

5.6.2.2 Vkládací tagy

Někdy je zapotřebí některé části šablony ve výsledném dokumentu zobrazit nebo naopak skrýt v závislosti na hodnotě některého atributu. K tomuto slouží speciální tag *casid_insert*. Příklad použití vidíme zde:

```
...
  <casid_insert ifdef="parameters:description">
    <p>
      [parameters:description]
    </p>
  </casid_insert>
...
```

Výše uvedený kód nám do výsledného dokumentu vloží odstavec s popisem zboží (*[parameters:description]*), ale pouze pokud je tento parametr pro dané zboží definován (*ifdef="parameters:description"*). Není-li parametr *description* ve zboží přítomen nebo je prázdný, odstavec se do výsledné šablony vůbec nevloží.

5.6.2.3 Opakovací tagy

Pokud potřebujeme určitou část kódu ve výsledném dokumentu opakovat, například pro všechny parametry zboží nebo pro všechny položky objednávky, které chceme vypsat jako jednotlivé řádky tabulky, použijeme speciální tag *casid_repeat*. Použití může vypadat podobně jako kód uvedený níže.

```
...
  <table>
    <tbody>
      <casid_repeat source="parameters" tag="param" hideonempty="true">
        <tr>
          <td>
            <strong>
              [name]
            </strong>
          </td>
          <td style="width: 100%;">
            [value]
          </td>
        </tr>
      </casid_repeat>
    </tbody>
  </table>
...
```

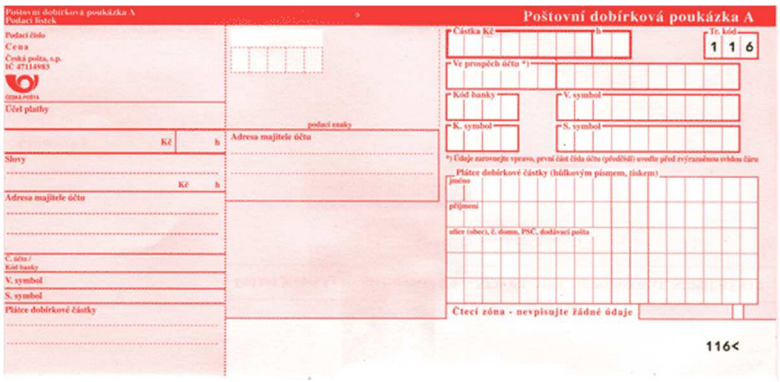
Výše uvedený kód nám do výsledného dokumentu vkládá všechny námi definované parametry zboží (*source="parameters"*) označené jako *param* (*tag="param"*). Vkládá je jako jednotlivé řádky HTML tabulky. Konkrétní místo vložení názvu parametru je pak označeno pomocí *[name]* a místo pro vložení hodnoty jednoduchým tagem *[value]*.

5.6.3 Tisk dokumentů

Dalším aspektem implementované webové aplikace je tisk. Jedná se zejména o tisk dokumentů souvisejících se zásilkami, jako jsou balíkové štítky, poštovní poukázky, podací archy, atd. Kromě nich je však možné tisknout dokumenty také k položkám jiných agend. V tomto případě se jedná například o faktury k objednávkám či reklamační protokoly k evidovaným reklamacím. Bylo by možné namítnout, že tyto dokumenty je možné generovat stejně jako výše uvedené emailové zprávy a zvláštní zacházení s nimi tedy není zapotřebí. Problémem u tiskových položek je ovšem formát těchto dokumentů, u nichž běžný HTML kód interpretovaný prohlížečem nevyhovuje.

Důvodem je lehce rozdílná interpretace HTML kódu napříč prohlížeči. Tento problém je zcela zanedbatelný například při tisku faktur nebo v reklamačních protokolech. Komplikace však nastává v případě tisku poštovních dokumentů, které zpravidla tiskneme na předem připravené formuláře.

Podívejme se na obrázek 5.16, kde je vyobrazena *Poštovní dobírková poukázka A* jako příklad jednoho z takových předtištěných formulářů.



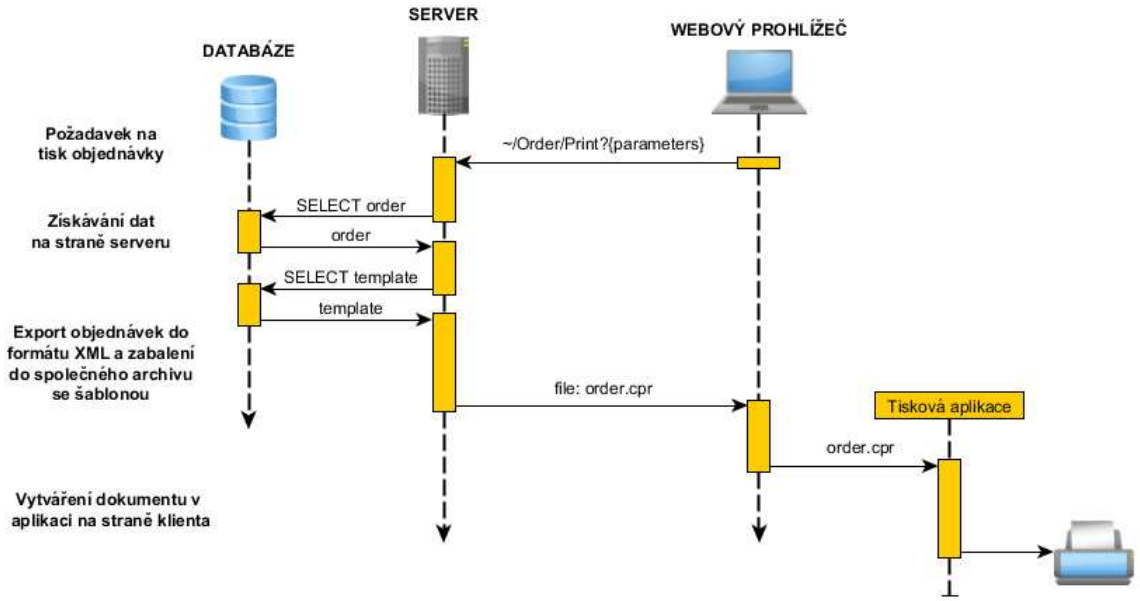
Obrázek 5.16: Poštovní dobírková poukázka A

Z obrázku je patrné, že tisknutí textu jako je číslo účtu, variabilní symbol a další musí být velice přesné proto, aby jednotlivé znaky byly přesně zacíleny do svých jednotlivých polí. Dosáhnutí takové přesnosti je takřka nemožné, pokud je text generovaný pomocí obyčejné HTML šablony a tisknutý na připravený formulář skrze webový prohlížeč. Experimentálně bylo zjištěno, že ne každý prohlížeč interpretuje například rozteč znaků u čísla účtu stejně a prohlížeče se taktéž rozchází v celkovém umístění textových řádků ve vytištěném dokumentu. Rovněž byla zvažována možnost vytvoření dokumentu ve formátu PDF a jeho následný tisk v aplikaci na straně klienta. Tato varianta však také nepřinesla požadované výsledky.

5.6.3.1 Princip funkce

Z důvodů popsaných výše bylo upuštěno od snahy využívat k tisku funkce internetového prohlížeče a byl navržen systém tiskových šablon ve formátu XML, pomocí nichž se výsledné dokumenty vytváří. Stejně jako u emailových šablon probíhá generování výsledného dokumentu na straně serveru. Další zpracování je ale od zasílání emailů rozdílné. Po vygenerování dokumentu je výsledný tiskový formulář, také ve formátu XML, zaslán zpět klientovi jako soubor. Uživatel si tento soubor stáhne skrze webový prohlížeč a otevře pomocí speciální malé desktopové aplikace, která je taktéž součástí projektu. Desktopová aplikace následně pouze vezme soubor, vytvoří z něj dokument a ten odešle k tisku. Proces tisku dokumentu ukazuje sekvenční diagram na obrázku 5.17.

Toto řešení se může zdát poněkud komplikované, ale pro přesný tisk do předpřipravených formulářů je opravdu nezbytné. V budoucím nasazení do aplikace přibude možnost tisknout dokumenty, u nichž tato přesnost není vyžadována přímo z okna webového prohlížeče. V současné verzi projektu však tuto funkci neuvažujeme.



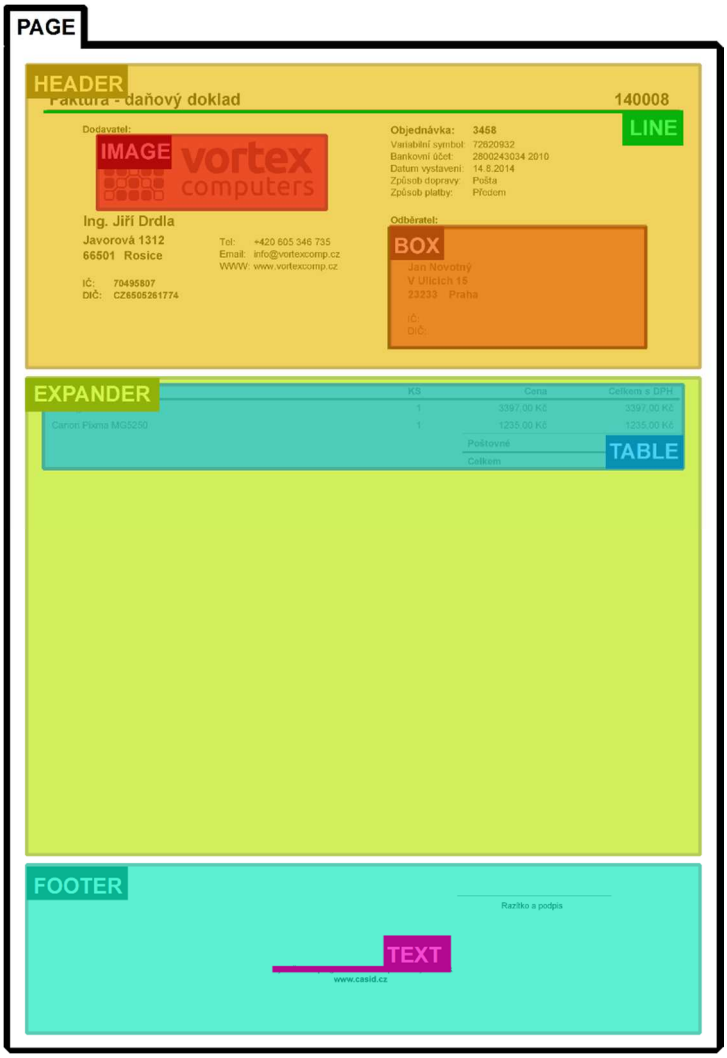
Obrázek 5.17: Proces tisku dokumentů

5.6.3.2 Struktura XML formuláře

Nyní si pojďme blíže představit strukturu tiskové šablony ve formátu XML. Popsány zde budou všechny elementy, které mohou být obsaženy v tiskové šabloně. Pro lepší představu fungování se však nejprve podívejme na obrázek 5.18, který nám graficky znázorňuje strukturu tiskové šablony.

Kořenovým elementem tiskové šablony je vždy element *Page*. Tento element reprezentuje vždy celý tištěný dokument, definuje jeho rozměry, název, agendu, pro kterou je implementován, atd. Jeho obsahem mohou být elementy s názvem *Header*, *Expander* a *Footer*. Elementy *Header* a *Footer*, jak u nich název napovídá, tvoří záhlaví a zápatí tištěného dokumentu. Element *Expander* reprezentuje samotné tělo dokumentu. V případě faktury, jak je vidět na obrázku 5.18, obsahuje tabulkové zobrazení jejích položek. Pokud obsah elementu *Expander* přeteče velikost svého bloku, automaticky je vytvořen dokument o více stranách a obsah volně pokračuje v rozsahu elementu *Expander* na další straně. Obsah elementů *Header* a *Footer* je v takovém případě na všech stranách tištěného dokumentu shodný.

V obrázku 5.18 jsou naznačeny některé typy elementů, které mohou bloky *Header*, *Expander* a *Footer* obsahovat. Výčet všech elementů, jejich možných atributů a přijatelných dceřiných elementů, které mohou být jejich obsahem, nalezneme v příloze 3.



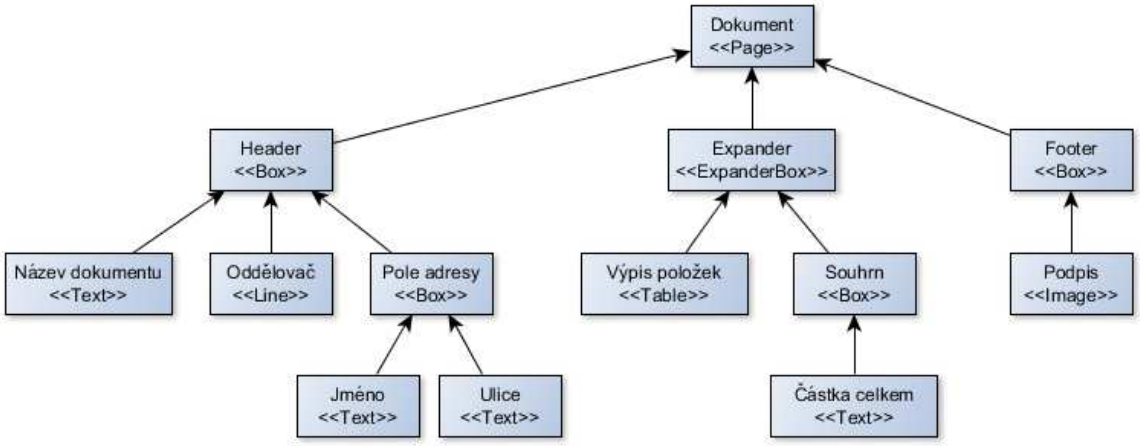
Obrázek 5.18: Struktura tiskové šablony

5.6.3.3 Zpracování dokumentu na straně klienta

Nyní se pojďme zaměřit na zpracování dokumentu na straně klienta. Jak jsme se dozvěděli výše, do tiskové aplikace přichází archív s uživatelem zvolenou tiskovou šablonou ve formátu XML a datovou reprezentací tiskových položek, například objednávek, rovněž ve formátu XML. Přijetím tohoto archívu dojde k naparsování tiskové šablony do hierarchické struktury objektů, které reprezentují jednotlivé elementy XML stromu. Podívejme se na diagram tříd, ze kterých hierarchie objektů vychází, najdeme jej v příloze 2g. Na obrázku 5.19 vidíme velice zjednodušenou hierarchickou strukturu objektů pro tištěný dokument.

Po naparsování XML dokumentu jsou objekty naplněny daty z přiložených objednávek a následuje procedura tisku. Při tisku se prochází hierarchická struktura objektů od kořenového uzlu a tím je objekt *Page*. Stromovou strukturou se prochází algoritmem Pre-order. [31] Při průchodu

uzlem je tento uzel vytisknut na interní canvas a pokračuje se na jeho prvního následníka. Po průchodu celého stromu je připravený canvas odeslán k tisku na uživatelem zvolené tiskové zařízení. Všechny tyto operace již probíhají mimo webovou aplikaci, tedy na počítači uživatele pomocí jednoduchého desktopového rozhraní.



Obrázek 5.19: Hierarchická struktura objektů popisující dokument

Zmiňovaná desktopová aplikace má velice spartánské grafické rozhraní, zahrnující pouze volbu pro načtení archívu z aplikace, která je následně okamžitě odeslána na tisk. Zamýšleným rozšířením pro budoucí vývoj je v každém případě integrace aplikace do operačního systému. To zahrnuje registraci přípony .CPR souborů stahovaných prostřednictvím webového prohlížeče do systému a jejich automatické otevírání tiskovou aplikací.

5.6.4 Import a export dat

V této podkapitole si popíšeme, jakým způsobem probíhá napojení implementované webové aplikace na externí systémy. Těmito systémy jsou služby třetích stran jako portály pro nabízení evidovaného zboží nebo systémy internetového bankovnictví.

Problém, který řeší import dat z různých zdrojů, je agregace těchto nehomogenních dat s různou strukturou a uložení do námi implementovaného systému. Data z externích systémů jsou na požadovaný formát transformována procedurálně, tedy přímo v kód u programu. Obecně lze říci, že data z různých zdrojů přicházejí do webové aplikace v tak rozdílných formátech, že vytvářet jim nějaký sjednocující adaptér by pouze zvýšilo složitost bez dalšího užitku. Kupříkladu data z internetového bankovnictví přicházejí do aplikace v čistém XML, portál Aukro.cz nabízí přístup ke svým datům prostřednictvím protokolu SOAP a například Česká pošta vyžaduje data o zásilkách v prostém CSV formátu. V rámci budoucího vývoje při rozšiřování počtu podporovaných externích systémů bude však vhodné nějakou vhodnou míru abstrakce nad procesy importu a exportu dat v aplikaci implementovat.

Export dat, v našem případě export zboží na prodejní portály, řeší problém opačný. Tedy, kterak zajistit transformaci dat ze struktur používaných v naší webové aplikaci na struktury požadované prodejním portálem. Tato transformace rovněž probíhá procedurálně, stejně jako proces importu popsaný výše.

5.6.4.1 Import objednávek z Aukro.cz

Aukro.cz je prodejní portál, ze kterého navrhovaná aplikace umí stahovat objednávky zákazníků. Základem pro komunikaci s Aukro.cz je jím poskytované webové API, viz [8]. Webové API je vlastně SOAP služba implementovaná nad protokolem HTTP. Po přihlášení do této služby může prodejce spravovat svůj účet stejně jako skrze webové rozhraní, avšak čistě datově orientovaně.

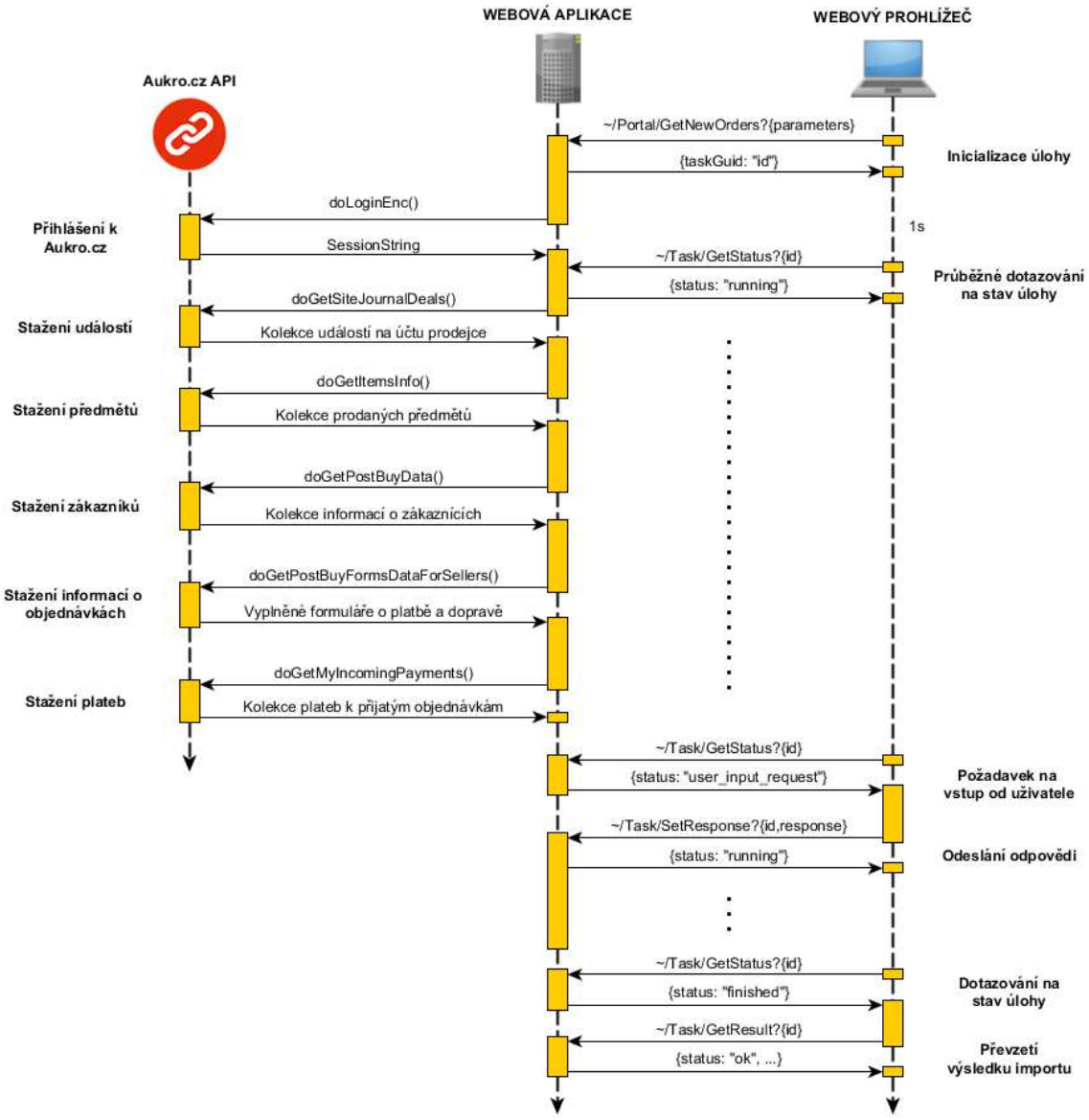
Přistupme tedy ke konkrétnímu popisu implementace importu objednávek z tohoto portálu. K portálu je nutné se nejdříve přihlásit, přihlášení nám zajistí vzdálená metoda *doLoginEnc()*. Návratovou hodnotou této metody je tzv. *SessionString* s platností 3 hodiny. Ten slouží při používání dalších metod jako identifikace přihlášeného uživatele. Po úspěšném přihlášení je potřeba stáhnout seznam událostí uskutečněných v našem portálu od minulého importu objednávek a to pomocí metody *doGetSiteJournalDeals()*. Tato metoda vrací seznam událostí jako je prodej zboží, vyplnění *Formuláře voleb přepravy* a zaplacení zboží. Jsou to však pouze metadata a informace o konkrétních prodaných produktech a zákaznících je nutné stáhnout zvlášť.

Začněme stažením informací o prodaných předmětech. K tomu slouží metoda *doGetItemsInfo()*, která vrací název prodaného zboží, cenu, počet kusů a další volitelné informace. Takto stažené produkty by se měly ideálně namapovat na evidované zboží v naší databázi. To se však nemusí stát například v tom případě, že prodejce předmět na portál vystaví bez naší aplikace, tedy ručně. V tomto případě se tedy prodávané zboží uloží do naší databáze.

Dále provedeme stažení dat zákazníků metodou *doGetPostBuyData()*. V naší webové aplikaci budou staženými daty buď aktualizována data již existujících zákazníků, nebo budou vytvoření zákazníci noví.

Zákazník může při prodeji libovolně vyplnit tzv. *Formulář voleb přepravy*, který informuje prodejce o zvoleném způsobu dopravy a platby zakoupeného zboží. Data vyplněných formulářů stáhneme z webové služby metodou *doGetPostBuyFormsDataForSellers()* a pomocí nich vyplníme potřebné údaje v námi vytvořených objednávkách.

Poslední položkou, kterou v importu využijeme, je seznam přijatých plateb za zboží zakoupené zákazníky. Aukro.cz využívá služeb platební brány PayU a skrze webovou službu je možné informace o přijatých platbách stáhnout. Na základě stavu zaplacení zboží se bude měnit stav importované objednávky. Pokud například již byla zaplacena, bude označena stavem *K odeslání* a naopak, pokud zaplacená stále nebyla, zůstane ve stavu *Přijata*. Pro stažení plateb k objednávkám zákazníků slouží metoda *doGetMyIncomingPayments()*. Na obrázku 5.20 vidíme sekvenční diagram průběhu importu dat z Aukro.cz. Diagram zahrnuje kromě komunikace webové aplikace se serverem Aukro.cz také komunikaci s webovým prohlížečem.



Obrázek 5.20: Sekvenční diagram importu dat z Aukro.cz

Po stažení všech těchto údajů je uživateli zobrazena souhrnná informace o realizovaných prodejích a je vyzván k potvrzení uložení stažených dat. Potvrzením dojde k transformaci všech přijatých dat na formát potřebný pro uložení v databázi aplikace a data jsou v ní uchována.

5.6.4.2 Export zboží na Aukro.cz

Základem exportu zboží na portál Aukro.cz je metoda *doNewAuctionExt()* poskytovaná webovým API portálu. Pro vystavení nové nabídky je zapotřebí se opět na portál přihlásit, přihlášení

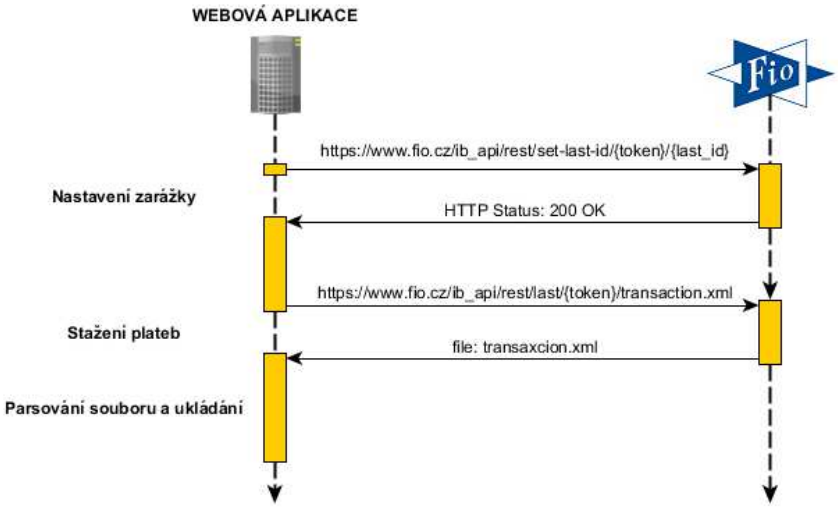
probíhá metodou *doLoginEnc()* a její návratovou hodnotou je již dříve uvedený *SessionString*. Tím je zapotřebí se identifikovat při volání dalších metod.

Po přihlášení je zapotřebí si zjistit, které parametry je nutné vyplnit v nově vystavované nabídce. Seznam parametrů získáme metodou *doGetSellFormFieldsForCategory()*. Návratovou hodnotou této metody je seznam parametrů, včetně jejich popisu, minimálních a maximálních hodnot, atd. Pokud máme seznam vyplňovaných parametrů, voláme metodu *doNewAuctionExt()*, jíž předáme kolekci struktur reprezentující parametry nově zveřejňované nabídky. Pokud jsou zadané parametry korektní, nabídka je vystavena na prodejní portál. Před samotným vystavením nové nabídky můžeme zavolat metodu *doCheckNewAuctionExt()*, která zkontroluje správnost zadaných údajů.

Správnost vystavení nabídky můžeme ověřit voláním metody *doVerifyItem()*, jíž předáme identifikátor námi zvolený při vystavování nabídky.

5.6.4.3 Import bankovních plateb z FIO banky

Import plateb z internetového bankovnictví FIO banky probíhá prostřednictvím bankou poskytovaného aplikačního rozhraní. Toto rozhraní je založeno na architektuře REST [41] postavené nad protokolem HTTP. Jedná se o jednoduché rozhraní, které dovoluje zadávat systému internetového bankovnictví příkazy, přičemž systém odpovídá formou textu s předem zvolenou strukturou. V našem případě byl zvolen formát XML, dostupné jsou však i další formáty jako CSV, JSON, GPC, atd. Celá komunikace mezi webovou aplikací a serverem banky je zabezpečena pomocí protokolu SSL s minimálně 128 bitovým šifrováním.



Obrázek 5.21: Sekvenční diagram importu plateb z FIO banky

Přístup k datům z internetového bankovnictví prostřednictvím API musí být nejprve povolen majitelem účtu nebo osobou s patřičným oprávněním. Tato osoba musí ve svém internetovém

bankovnictví vygenerovat 64 znakový token, kterým se při následné komunikaci se serverem webová aplikace identifikuje. Tato činnost musí být provedena manuálně. V dalším textu předpokládejme, že tento token již byl vygenerován.

Aplikační rozhraní banky poskytuje systém s takzvanou zarážkou. Představme si, že platby evidované bankou jsou uloženy v jednom dlouhém seznamu. Vyžádáním seznamu plateb prostřednictvím aplikačního rozhraní dojde k nastavení zarážky na poslední v seznamu plateb, které byly aplikaci vypsány. Každým dalším stažením plateb poté dochází ke stažením pouze takových, které jsou umístěny pod zarážkou. To jest takových, které od posledního stažení v systému přibyly.

Zarážku je však možné v případě potřeby posunout na libovolnou jinou transakci na základě znalosti jejího identifikátoru.

Nyní však přistupme k samotnému importu plateb. Prvním krokem je nastavení zarážky na poslední námi evidovanou platbu. To provádíme z toho důvodu, že pokud by býval při posledním importu nastal nějaký problém, zarážka by mohla být nastavena na jinou platbu než je námi poslední evidovaná. Po nastavení zarážky následuje stažení seznamu posledních plateb, a pokud proběhne úspěšně, obdržený XML dokument je naparsován a informace o platbách jsou uloženy do databáze webové aplikace. Na obrázku 5.21 vidíme proces importu bankovních plateb včetně přesné podoby zasílaných příkazů.

5.6.5 Zálohování a obnova

Webová aplikace přináší uživateli možnost zálohovat si svá data, tedy přesněji veškeré svoje evidované objednávky, zboží, nákupy atd. Zálohu si uživatel může stáhnout ve formě souboru přímo z prostředí webové aplikace. Tento soubor může později sloužit pro návrat ke starším verzím dat prodejce, může být použit pro přenos dat různými uživatelskými účty v aplikaci, ale především bude sloužit k migraci mezi desktopovou a webovou verzí aplikace.

Jak jsme se dozvěděli výše, při registraci se pro každého uživatele vytváří zvláštní databáze, v níž jsou uložena výlučně jeho data. Schéma této databáze je shodné se schématem desktopové verze této aplikace. Prodejce tak má možnost pomocí zálohy přenést data z lokální desktopové verze aplikace do webového rozhraní nebo opačně. Toto přináší zejména pro podniky klíčovou vlastnost, kdy prodejce má možnost mít svá data zcela pod svojí kontrolou, pokud si to bude přát.

Záloha i obnova databáze je řešena jako úloha běžící na pozadí, tyto úlohy byly popsány v podkapitole 5.6.1. Nyní se blíže podíváme na proces zálohování. Jelikož veškerá data uživatele jsou obsažena pouze v jemu vyhrazené databázi, záloha probíhá formou vytvoření skriptu reprezentujícího obsah této databáze. Skript je samozřejmě vytvářen na straně serveru a to pomocí utility *mysqldump.exe*, která je standardní součástí instalace MySQL serveru. Utilita je běžnou konzolovou aplikací, která po zadání příslušných přístupových údajů a specifikování jména databáze vytvoří textový SQL skript reprezentující databázové schéma a data. K vytvořenému skriptu je následně přiložen soubor určující verzi aplikace, ve které záloha vznikla. Oba tyto soubory jsou zkomprimovány metodou ZIP a ve formě archívu nabídnuty ke stažení uživatelem.

Nyní si ve stručnosti představíme také proces obnovy. Jeho základem je vytvořený archív zálohy popsaný výše. Nahráním tohoto archívu do webového rozhraní aplikace dojde k jeho rozbalení na straně serveru. Jelikož máme zálohu připravenou ve formě skriptu, použijeme utilitu *mysql.exe*, opět z balíku instalace MySQL serveru, pro nahrání tohoto skriptu do databáze. Následně ještě zkontrolujeme, zda souhlasí verze nahrané databáze s aktuální aplikací. Pokud tomu tak není a verze databáze je starší, aplikujeme potřebné rozdílové skripty, aby schéma databáze odpovídalo aktuální verzi webové aplikace. Tímto jsou veškerá předchozí data přepsána zálohou a proces obnovy je dokončen.

Kapitola 6

Propagace produktu

V této kapitole si shrneme možné možnosti propagace navrhované webové aplikace tak, aby se snáze dostala ke svým zákazníkům – maloobchodním prodejcům. Dále si představíme metody, které byly k propagaci produktu zvoleny a provedeme zhodnocení jejich účinnosti. Popíšeme si rovněž detaily nasazení aplikace v komerčním prostředí, což bylo také jedním z cílů této diplomové práce.

> *"Internetový marketing je tady proto, že je tady internet."* [2]

6.1 Možnosti propagace

V následující podkapitole si blíže představíme některé možnosti, které můžeme využit k propagaci výsledného produktu. Omezíme pouze na metody propagace produktu cestou internetu. Propagace produktu tradičními metodami, jako je například inzertní kampaň v médiích, bývá pro jednotlivce vyvíjející internetovou aplikaci zpravidla finančně nerealizovatelná.

6.1.1 Webová prezentace

Zcela nezbytnou položkou při propagaci navrhované webové aplikace je její internetová prezentace. Na webové prezentaci musejí potenciální zákazníci být informování především o účelu poskytovaného produktu a zpravidla je také vstupní branou do samotné aplikace. Prezentace by měla být laděna v podobném grafickém duchu jako finální produkt, takové řešení zajistí dojem větší komplexnosti a lepšího uživatelského zážitku.

Kromě klasické textové specifikace funkcí by hlavní stránka webové prezentace měla obsahovat interaktivní schéma fungování aplikace, které by mělo na první pohled poskytnout povědomí o její samotné funkci. Důležitou součástí webové prezentace jsou také materiály, které případného zájemce blíže seznámí s funkcemi aplikace a poskytnou informace, zda produkt splňuje očekávané požadavky. Moderním přístupem v poskytování těchto informací jsou online videa. Divákovi poskytují mnohem snazší cestu k požadovaným informacím než klasické textové popisy a zjednodušování je, možná naneštěstí, obecným trendem dnešní doby.

6.1.2 Google AdWords

Aby byla webová prezentace produktu v záplavě webových stránek na internetu vůbec dohledatelná, je nutné ji zaregistrovat do běžně používaných katalogů internetových stránek. Samotná registrace do katalogů však povětšinou zvýšení návštěvnosti webových stránek nezajistí. V dnešní době přichází na webové servery největší procento unikátních návštěvníků, tj. takových, kteří přichází na web poprvé, skrze internetové vyhledávače. [2] Cílem provozovatele webu je tak zobrazit svoji prezentaci co možná nejvýše ve výsledcích vyhledávání. K dosažení toho zpravidla vede cesta prostřednictvím předplacení proklikové reklamy v internetových vyhledávačích.

V českých poměrech jsou to vyhledávače Seznam.cz [38] a Google [39], kteří společně zastávají zcela dominantní podíl na trhu. V případě tohoto projektu by pak placená reklama měla být přednostně zobrazována ve výsledcích vyhledávání klíčových slovních spojení, jako je *správa objednávek*, *tisk poštovních poukázek*, apod.

6.1.3 Bannerová inzerce

Poměrně rozšířenou formou propagace aplikace je využití reklamních ploch na webech třetích stran k umístění grafických reklamních odkazů, jinak řečeno bannerů. Vzhledem k zaměření aplikace na poměrně užší skupinu internetových prodejců, by tato reklama měla být umístěna zejména na webové portály zaměřené nějakým způsobem na internetový prodej, setkávání maloobchodníků, apod. Takovým webem je například Webtrh.cz nebo Lupa.cz.

Umístění banneru na takové weby je pak otázkou individuální dohody s provozovatelem a je obecně těžké o takové formě spolupráce předem odvodit její obecný průběh.

6.1.4 Sociální sítě

Fenoménem posledních deseti let je prezentace jednotlivců, firem i zájmových skupin na sociálních sítích. Pro aplikace, jako je ta vytvořená v rámci diplomové práce, je hlavním důvodem prezentace v této oblasti především možnost udržování trvalého styku se zákazníky. Primárně tedy neslouží k přímému získávání nových uživatelů, ale spíše k vytváření komunity okolo projektu, informování současných zákazníků o novinkách v aplikaci a udržování kontaktu v podobě diskuze.

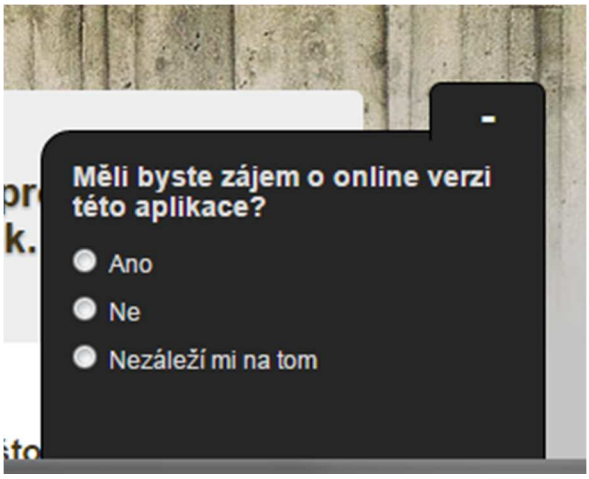
Opět je však potřeba zvážit potřebu takové prezentace vzhledem k zaměření konkrétní aplikace. Prezentace na sociálních sítích může být vhodná pro projekty určené k masovému použití, což navrhovaná aplikace není. Správa profilu na sociální síti rovněž vyžaduje poměrně velké úsilí a zpravidla každodenní monitorování reakcí uživatelů a odpovídání na jejich příspěvky.

Z pohledu provozovatele aplikace je hlavní slabinou celého systému prezentace na sociálních sítích především nemožnost moderovat a regulovat uživateli psané příspěvky. Toto může vyústit v ne zcela příznivou situaci pro provozovatele, kdy negativní obsah několika málo příspěvků může vážným způsobem poškodit reputaci celého projektu.

6.2 Zvolený způsob propagace

Původním záměrem zamýšleným v době návrhu webové aplikace byla realizace propagace výsledného produktu prostřednictvím webové prezentace a její zviditelnění v podobě reklamní kampaně ve vyhledávači Google. Neplánované prodloužení doby práce na samotné webové aplikaci však nedovolilo tento časově poměrně náročný plán realizovat. Výsledkem byla volba způsobu propagace, který by byl realizovatelný ještě v průběhu tvorby samotného produktu.

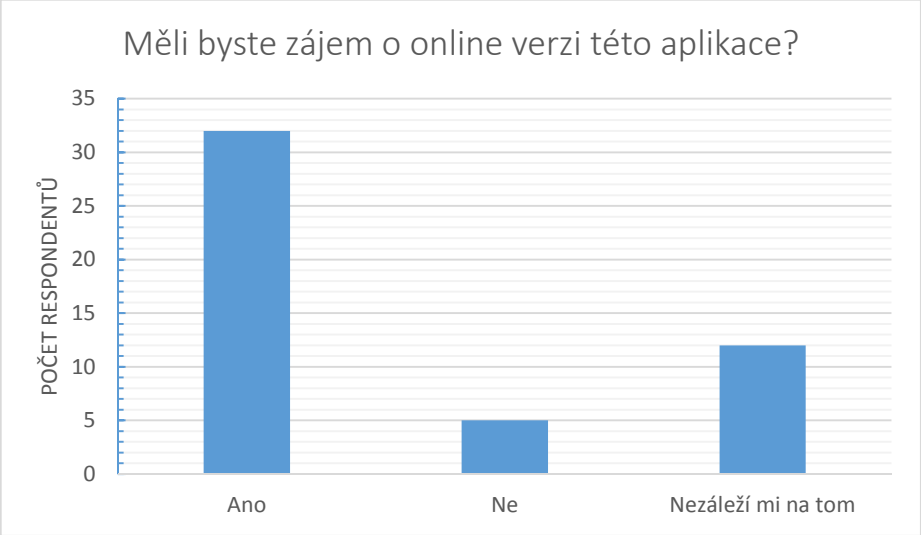
Variantou, která by splňovala tuto podmínku, byla zvolena metoda anketního dotazníku na webové prezentaci současné desktopové verze aplikace. Prezentace desktopové verze se nachází na doméně www.casid.cz a v anketní otázka byla na tomto webu umístěna po dobu dvou měsíců od 1. 3. 2015 do 30. 4. 2015. Anketní otázka se zobrazovala v pravém spodním rohu webové stránky a její podobu nalezneme na obrázku 6.1.



Obrázek 6.1: Anketní otázka na webové stránce

Anketní box byl realizován prostřednictvím služby Feedbackdaddy.com [35] v její zdarma dostupné variantě. Služba nabízí automatické vytvoření dotazníkového pole, které lze vložit do webové stránky formou vygenerované knihovny JavaScriptu. Zároveň umožňuje průběžně vyhodnocovat výsledky ankety prostřednictvím svého administračního rozhraní. Během zmíněných dvou měsíců zodpovědělo otázku celkem 49 respondentů, přičemž celkový počet unikátních návštěvníků webu za toto období byl 366. Na anketní otázku tedy odpovědělo 7,47% návštěvníků. Výsledky ankety lze nalezneme na grafu 6.1.

Pojďme se tedy zaměřit na krátké zhodnocení nasbíraných dat. Z grafu je patrné, že majoritní skupina respondentů má zájem na tvorbě webové verze aplikace, konkrétně 65,3 % dotázaných. Druhá největší skupina čítající 24,5 % dotázaných odpověděla neutrálně a posledních 10,2 % respondentů vyjádřilo nesouhlas nad tvorbou online verze.



Graf 6.1: Výsledky ankety na webu www.casid.cz

Respondentům odpovídajícím na anketní otázku kladně byl po zaškrtnutí volby nabídnut vstupní box pro zadání emailové adresy, na kterou mohou dostávat informace o novinkách ohledně přípravy webové verze. Z celkového počtu 32 kladně hlasujících vyplnilo svou emailovou adresu 26 dotázaných. Všem z nich byl automaticky odeslán email jako poděkování za poskytnutý hlas a následně v průběhu dvou měsíců dvě emailové zprávy informující o průběhu implementace a předpokládaném datu dokončení. Celkem 3 uživatelé následně kladli dotaz v odpovědi na zasílaný email, čímž dále potvrdili svůj zájem o online verzi.

6.3 Reálné nasazení

Jedním z bodů zadání bylo otestovat webovou aplikaci v reálném nasazení. Toto nasazení aplikace proběhlo v internetovém obchodě Vortex Computers, zabývajícím se prodejem výpočetní techniky a jiné elektroniky [36] a dále ve společnosti OnStyle s.r.o., zaměřující se na výrobu a prodej rytých šperků [37]. V obou těchto případech byla aplikace používána paralelně s již existující instalací desktopové aplikace, přičemž pracovala nad stejnými daty. Cílem nasazení bylo především získání zpětné vazby z reálného použití.

Obrázek 6.2: Společnosti testující webovou aplikaci

V obou případech proběhlo nasazení bez obtíží a bylo získáno velké množství podnětů pro aktuální i budoucí vývoj celé webové aplikace. Výhodou byla možnost použití stávajícího ověřeného desktopového systému, pokud by z jakéhokoli důvodu webová verze nefungovala.

Níže nalezneme stručný výpis některých podnětů pro další vývoj webové aplikace. Přijímány byly podněty ze všech možných oblastí a pro prezentaci vybíráme ty zajímavější:

- Přizpůsobit zobrazení detailů položek agend pro monitory s velmi vysokým rozlišením. V současné době dochází na extrémně velkých displejích k nepříjemnému roztažení ovládacích prvků přes celou obrazovku.
- Implementovat agendu pro evidenci vypůjčeného zboží.
- Na hlavní stránce zobrazit komponentu pro přímé spojení s technickou podporou.
- Implementovat napojení na další prodejní portály a bankovní systémy.
- Tisk přehledu zboží, které je aktuálně skladem.

Kapitola 7

Závěr

Cílem této práce bylo navrhnout informační systém pro podporu maloobchodního prodeje. V úvodní části byl problém analyzován, dále byly specifikovány požadavky na navrhovaný informační systém, požadavky byly zkonfrontovány s jinými existujícími řešeními a dále bylo přistoupeno k návrhu. V návrhu byly podrobně popsány jednotlivé funkce programu, doplněny grafickými návrhy obrazovek a diagramy vytvořenými v konvenci jazyka UML. V poslední části návrhu se zaměřujeme na návrh databáze a tříd v aplikaci. Následuje obsáhlá kapitola popisující implementaci webové aplikace a dále kapitola o možnostech propagace produktu.

V tomto diplomovém projektu se podařilo vytvořit webovou aplikaci implementující až na výjimky všechnu požadovanou funkčnost. Těmito výjimkami, které se do doby odevzdání projektu nepodařilo zdárně nasadit je evidence sériových čísel a evidence parametrů u zboží.

V průběhu testování při reálném nasazení v obou společnostech byly zaznamenávány četné podněty pro rozšíření funkčnosti aplikace nad rámec této diplomové práce. Z důvodu pozitivních ohlasů na tento projekt by bylo velice vhodné tuto práci dále rozvíjet a přivézt k nasazení u většího počtu klientů.

Literatura

[1] MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně. Vyd. 1. Brno: Zoner Press, 2011, 2 sv. (880, 700 s.). Encyklopedie Zoner Press. ISBN 978-80-7413-131-8.

[2] JANOUCH, Viktor. Internetový marketing: prosaďte se na webu a sociálních sítích. Vyd. 1. Brno: Computer Press, 2010, 304 s. ISBN 978-80-251-2795-7.

[3] Data Developer Center: Entity Framework. Microsoft Developer Network [online]. United States: Microsoft, 2015, [vid. 2014-12-18]. Dostupné z https://msdn.microsoft.com/en-us/data/ef.aspx.

[4] MySQL: Documentation. MySQL: The world's most popular open source database [online]. United States: Oracle Corporation, 2015, [vid. 2014-11-13]. Dostupné z http://dev.mysql.com/doc/.

[5] HALVORSON, Michael. Microsoft Visual Basic 2010: krok za krokem. Vyd. 1. Brno: Computer Press, 2010, 480 s. Krok za krokem (Computer Press). ISBN 978-80-251-3146-6.

[6] Visual Studio: .NET Framework and .NET SDKs. Microsoft Developer Network [online]. United States: Microsoft, 2015, [vid. 2015-01-12]. Dostupné z https://msdn.microsoft.com/en-US/vstudio/aa496123.aspx.

[7] Visual Studio: Visual Basic resources. Microsoft Developer Network [online]. United States: Microsoft, 2015, [vid. 2015-01-11]. Dostupné z https://msdn.microsoft.com/en-us/vstudio/hh388573.aspx.

[8] Allegro WebAPI: Documentation. Aukro [online]. Praha (Česká Republika): Allegro Group CZ, 2014, [vid. 2014-11-02]. Dostupné z http://aukro.cz/webapi/documentation.php.

[9] Manažer Prodeje. Aukro [online]. Praha (Česká Republika): Allegro Group CZ, 2014, [vid. 2014-11-21]. Dostupné z http://mp.aukro.cz.

[10] Money S3: Vlastnosti systému. Money S3 [online]. Brno (Česká Republika): Cígler software, 2014, [vid. 2014-12-10]. Dostupné z http://www.money.cz/money-s3/vlastnosti-systemu/.

[11] OpenCart: Features. OpenCart [online]. Tuen Mun (Hong Kong): OpenCart Limited, 2015, [vid. 2015-01-09]. Dostupné z http://www.opencart.com/index.php?route=feature/feature.

[12] PrestaShop: Features. PrestaShop [online]. Levallois-Perret (France): PrestaShop SA, 2015, [vid. 2015-01-10]. Dostupné z https://www.prestashop.com/en/ecommerce-templates.

[13] Česká pošta: Balík Do ruky. Česká pošta [online]. Praha (Česká Republika): Česká pošta, 2014, [vid. 2014-11-12]. Dostupné z https://www.ceskaposta.cz/sluzby/baliky/cr/balik-do-ruky.

[14] Česká pošta: Poštovní poukázka A. Česká pošta [online]. Praha (Česká Republika): Česká pošta, 2014, [vid. 2014-11-12]. Dostupné z https://www.ceskaposta.cz/sluzby/platebni-a-financni-sluzby-cr/postovni-poukazka-a.

[15] Microsoft Virtual Academy: Introduction to ASP.NET MVC. Microsoft Virtual Academy: Bezplatné školení od společnosti Microsoft pod vedením odborníků [online]. United States: Microsoft, 2014-08-08, [vid. 2014-11-20]. Dostupné z http://www.microsoftvirtualacademy.com/training-courses/introduction-to-asp-net-mvc.

[16] Wikipedia: Reflexe (programování). Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 2002, naposledy editováno 27. 2. 2015 v 00:05, [vid. 2015-05-12]. Dostupné z http://cs.wikipedia.org/wiki/Reflexe_(programování).

[17] w3schools.com: ASP.NET MVC - HTML Helpers. w3schools.com: The world's largest web developer site [online]. Norsko: Refsnes Data, 2015, [vid. 2015-01-30]. Dostupné z http://www.w3schools.com/aspnet/mvc_htmlhelpers.asp.

[18] ASP.NET: Views and ViewModels. ASP.NET [online]. United States: Neudesic, 2015, [vid. 2015-02-22]. Dostupné z http://www.asp.net/mvc/overview/older-versions/mvc-music-store/mvc-music-store-part-3.

[19] Bootstrap: Getting started. Bootstrap [online]. komunitní projekt, 2015, [vid. 2015-02-07]. Dostupné z http://getbootstrap.com/getting-started/.

[20] w3schools.com: The XMLHttpRequest Object. w3schools.com: The world's largest web developer site [online]. Norsko: Refsnes Data, 2015, [vid. 2015-03-28]. Dostupné z http://www.w3schools.com/xml/xml_http.asp.

[21] Bootstrap Table: Documentation. Bootstrap Table [online]. China: Scutech Corporation, 2015, [vid. 2015-02-15]. Dostupné z http://bootstrap-table.wenzhixin.net.cn/documentation/.

[22] Bootbox.js: Bootbox API. Bootbox.js: Bootstrap modals made easy [online]. Guiseley (England): Nick Payne, 2015, [vid. 2015-03-12]. Dostupné z http://bootboxjs.com/documentation.html.

[23] Pace: Automatic page load progress bar. HubSpot Dev & Design: We build software to help businesses succeed [online]. Cambridge (England): HubSpot, 2015, [vid. 2015-02-02]. Dostupné z http://github.hubspot.com/pace/.

[24] Datepicker for Bootstrap. Stefan Petre [online]. Brasov (Romania): Stefan Petre, 2015, [vid. 2015-03-31]. Dostupné z http://www.eyecon.ro/bootstrap-datepicker/.

[25] Bootawrap: Bootstrap templates. Bootswrap [online]. Scottsdale (Arizona): Bootswrap, 2014, [vid. 2014-12-23]. Dostupné z http://www.bootswrap.com/.

[26] ANTONIO. A Brief History of the Hamburger Icon. In: Placeit blog [online]. 29. 10. 2014 [vid. 2015-05-13]. Dostupný z: http://blog.placeit.net/history-of-the-hamburger-icon/.

[27] Wikipedia: Don't repeat yourself. Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 2013, last modified on 3 May 2015, at 18:44 [vid. 2015-05-05]. Dostupné z http://en.wikipedia.org/wiki/Don%27t_repeat_yourself.

[28] Microsoft Developer Network: Introduction to Membership. Microsoft Developer Network [online]. United States: Microsoft, 2015, [vid. 2015-04-18]. Dostupné z https://msdn.microsoft.com/en-us/library/yh26yfzy%28v=vs.140%29.aspx.

[29] OAuth: About OAuth 2.0. OAuth [online]. komunitní projekt, 2015, [vid. 2015-04-18]. Dostupné z http://oauth.net/2/.

[30] Wikipedia: Breadcrumb (navigation). Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 2014, last modified on 28 December 2014, at 04:00 [vid. 2015-05-10]. Dostupné z http://en.wikipedia.org/wiki/Breadcrumb_%28navigation%29.

[31] Wikipedia: Tree traversal. Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 2005, last modified on 2 May 2015, at 19:21 [vid. 2015-05-12]. Dostupné z http://en.wikipedia.org/wiki/Tree_traversal#Pre-order.

[32] ASP.NET: Getting Started with ASP.NET Identity. ASP.NET [online]. United States: Neudesic, 2015, [vid. 2015-02-27]. Dostupné z http://www.asp.net/mvc/overview/older-versions/mvc-music-store/mvc-music-store-part-3.

[33] FREEMAN, Adam. Pro ASP.NET MVC 4. 4th ed. Berkeley, California: Apress, 729 stran, 2012. ISBN 9781430242376.

[34] FEW, Stephen. Information dashboard design: the effective visual communication of data. 1st ed. Cambride [MA]: O'Reilly, 2006, 211 stran. ISBN 9780596100162.

[35] FeedbackDaddy: What we do. FeedbackDaddy [online]. Toronto (Canada): feedbackdaddy, 2014, [vid. 2015-05-15]. Dostupné z https://www.feedbackdaddy.com.

[36] Aukro: Vortex Computers – stránka O Mně. Aukro [online]. Praha (Česká Republika): Allegro Group CZ, 2014, [vid. 2015-05-15]. Dostupné z http://aukro.cz/my_page.php?uid=6072332.

[37] OnStyle: O nás. OnStyle: dává víc [online]. Karlovy Vary (Česká Republika): ONstyle s.r.o., 2015, [vid. 2015-05-15]. Dostupné z http://www.onstyle.cz/#whoweare.

[38] Sklik.cz: PPC reklama. Sklik.cz [online]. Praha (Česká Republika): Seznam.cz, 2015, [vid. 2015-05-16]. Dostupné z https://www.feedbackdaddy.com.

[39] Google AdWords: Princip. Google AdWords [online]. Mountain View (United States): Google Inc., 2015, [vid. 2015-05-15]. Dostupné z http://www.google.cz/adwords/start/how-it-works/.

[40] Balsamiq: Balsamiq Mockups. Balsamiq [online]. Sacramento (United States): Balsamiq Studios, 2015, [vid. 2015-05-20]. Dostupné z https://balsamiq.com/products/mockups/.

[41] Wikipedia: Representational State Transfer. Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 2008, last modified on 22 May 2015, at 16:19 [vid. 2015-05-22]. Dostupné z http://en.wikipedia.org/wiki/Representational_state_transfer.

Seznam použitých zkratek a symbolů

Zkratka	Celý název	Vysvětlení
API	Application Programming Interface	Rozhraní pro programování aplikací
AJAX	Asynchronous JavaScript and XML	Technologie pro interaktivní webové stránky
ASP	Active Server Pages	Technologie pro tvorbu webových aplikací
CRUD	Create-Read-Update-Delete	Zkratka pro 4 základní operace nad daty
CSV	Comma-separated values	Hodnoty oddělené čárkami, formát souboru
HTML	Hypertext Markup Language	Značkovací jazyk pro webové stránky
MSSQL	Microsoft SQL	Databázový server
MVC	Model-View-Controller	Vzor pro vytváření webových aplikací
SOAP	Simple Object Access Protocol	Protokol pro výměnu strukturovaných dat
SQL	Structured Query Language	Dotazovací jazyk pro práci s databází
UML	Unified Modeling Language	Grafický jazyk pro vizualizaci
XML	Extensible Markup Language	Rozšiřitelný značkovací jazyk

Seznam příloh

Příloha 1 ER diagram databáze uživatele

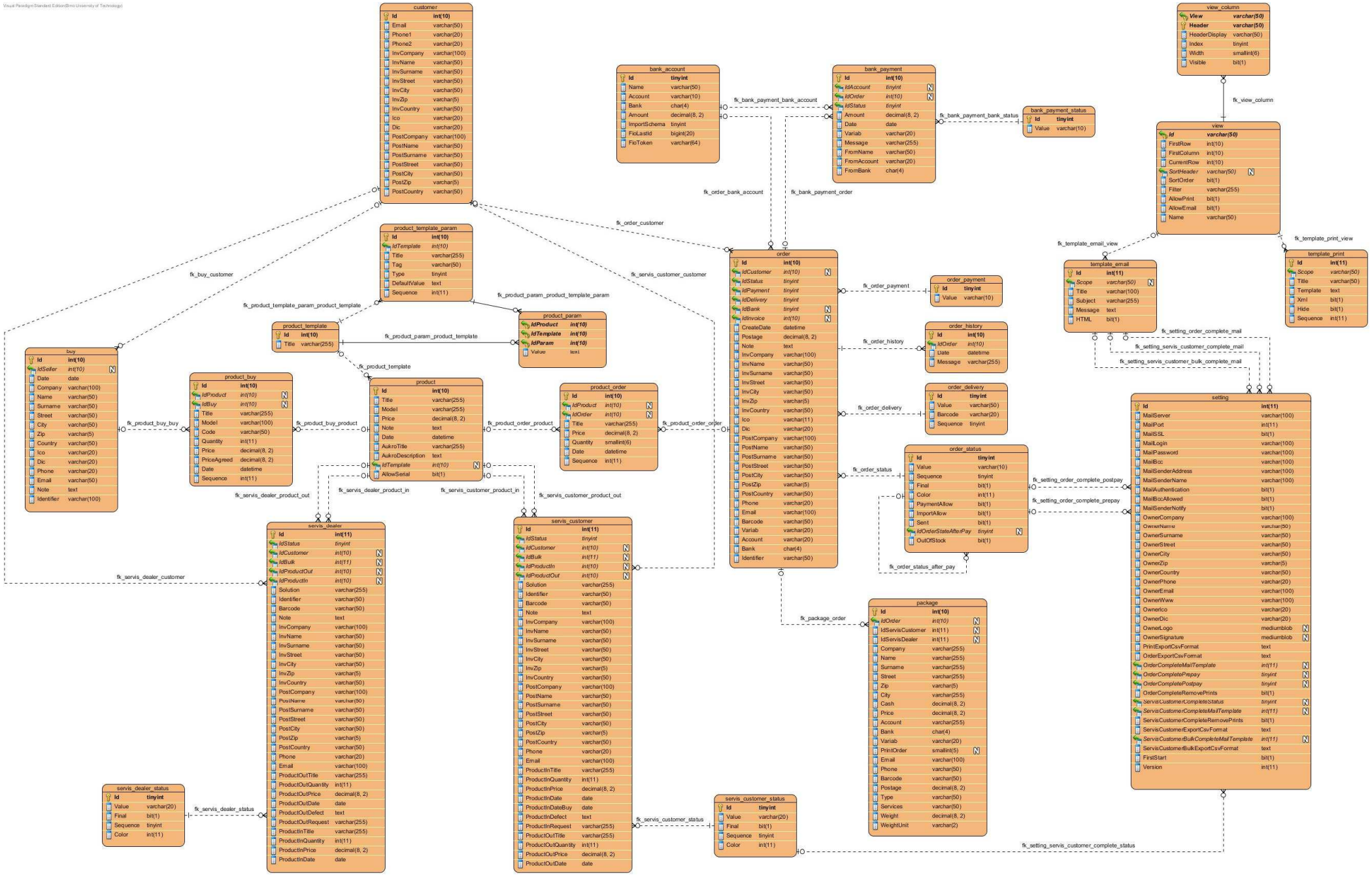
Příloha 2 Konceptuální diagramy tříd

- Příloha 2a Konceptuální diagram tříd pro objednávky
- Příloha 2b Konceptuální diagram tříd pro zboží
- Příloha 2c Konceptuální diagram tříd pro reklamace
- Příloha 2d Konceptuální diagram tříd pro bankovní platby
- Příloha 2e Konceptuální diagram tříd pro portál Aukro.cz
- Příloha 2f Konceptuální diagram tříd pro nastavení a pohledy
- Příloha 2g Konceptuální diagram tříd pro tisk dokumentů

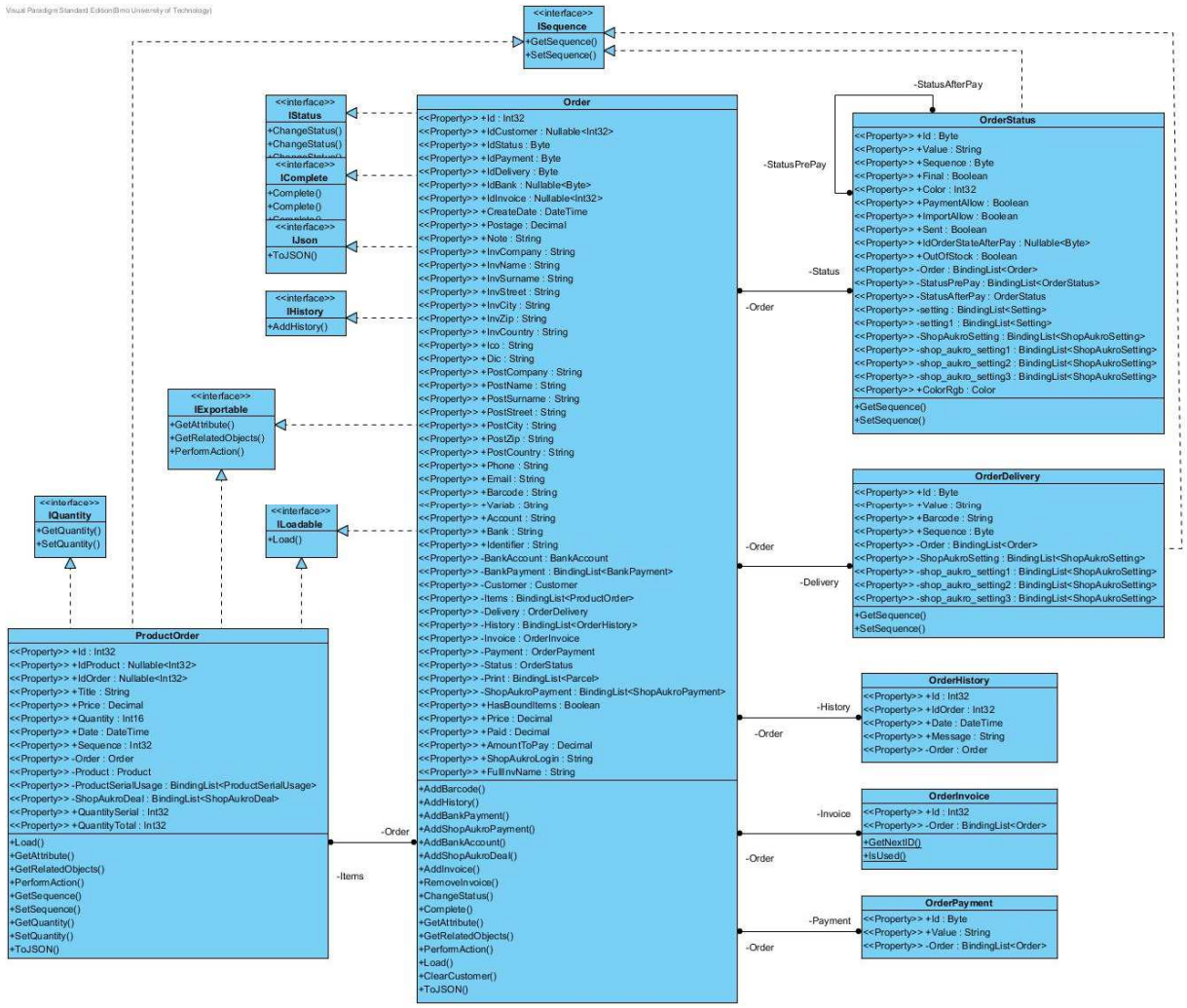
Příloha 3 Seznam XML elementů pro vytváření tiskových šablon

Příloha 4 Příklad emailové šablony

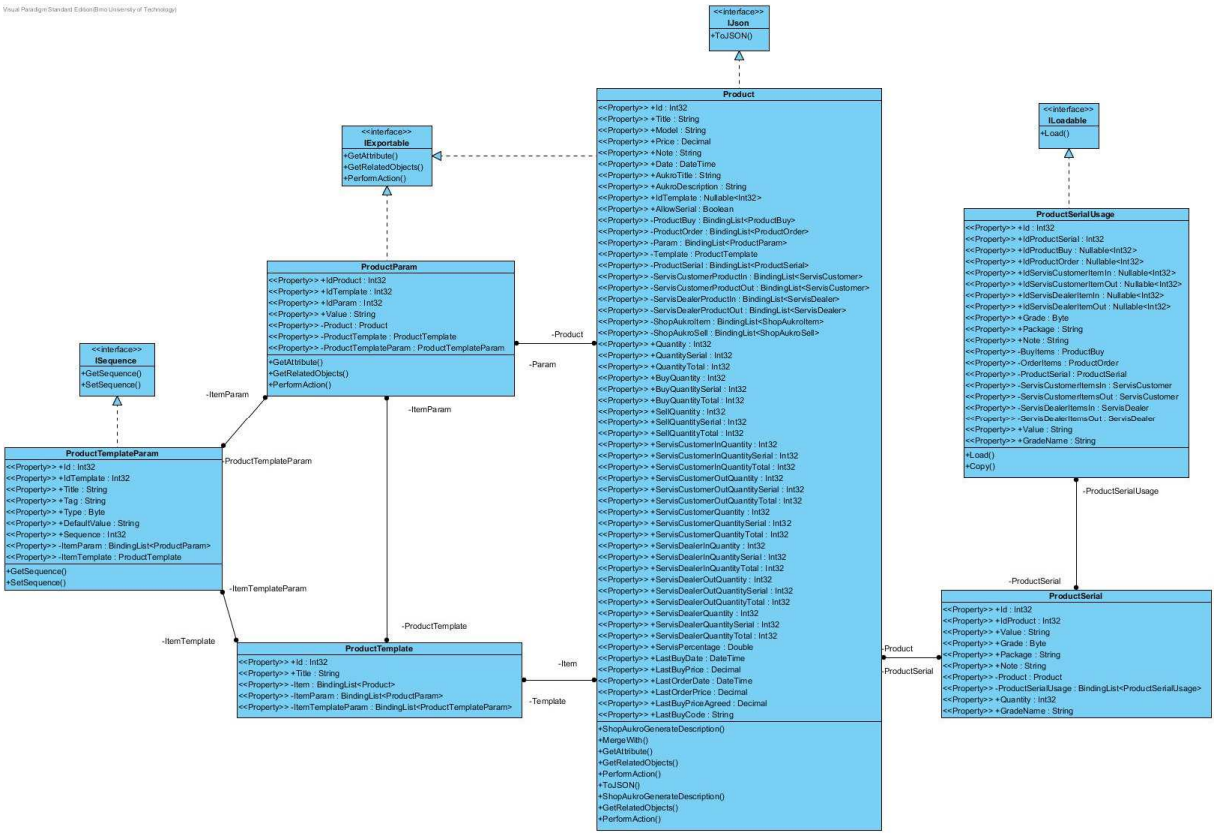
Příloha 1 – ER diagram databáze uživatele



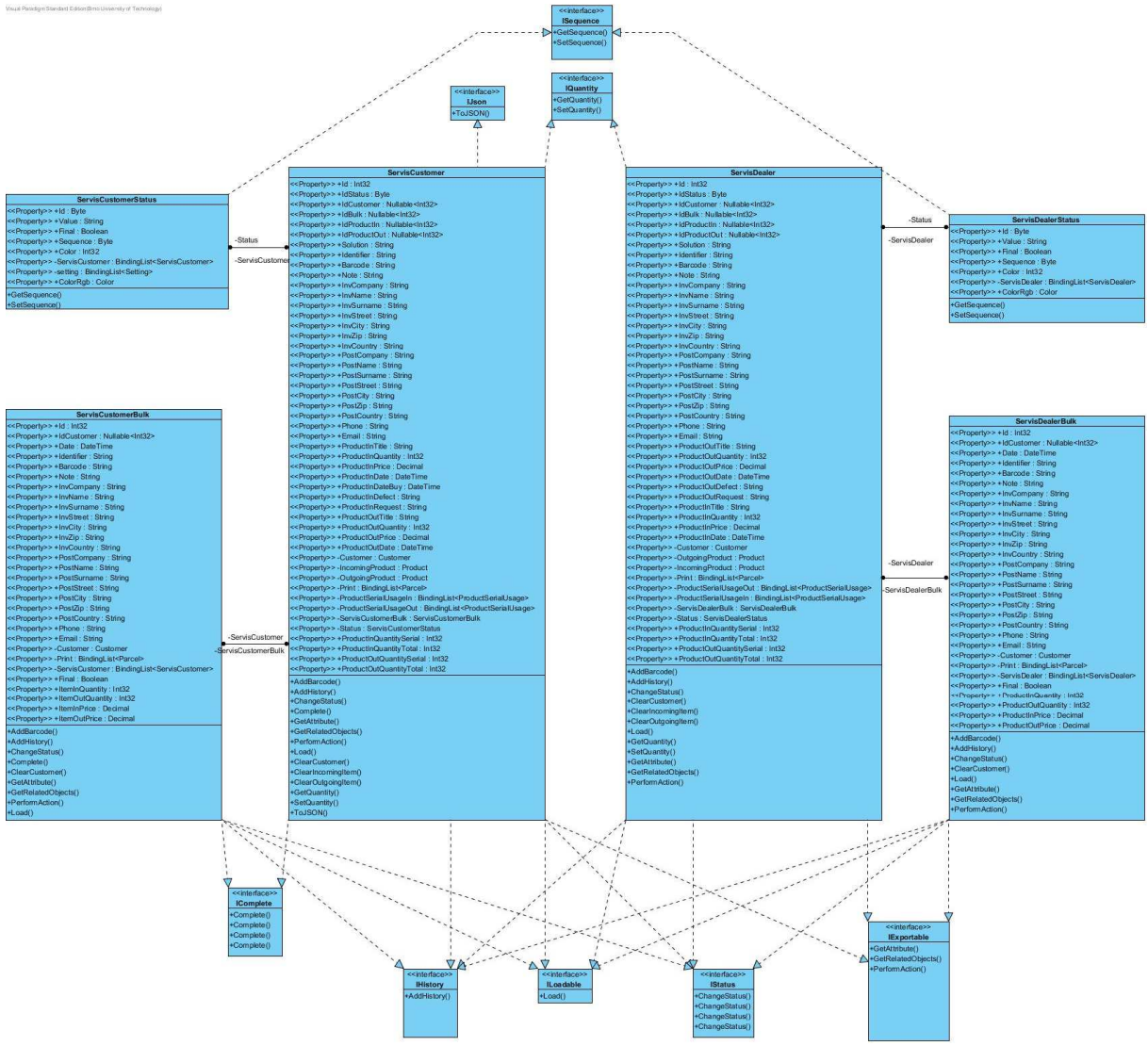
Příloha 2a – Konceptuální diagram tříd pro objednávky



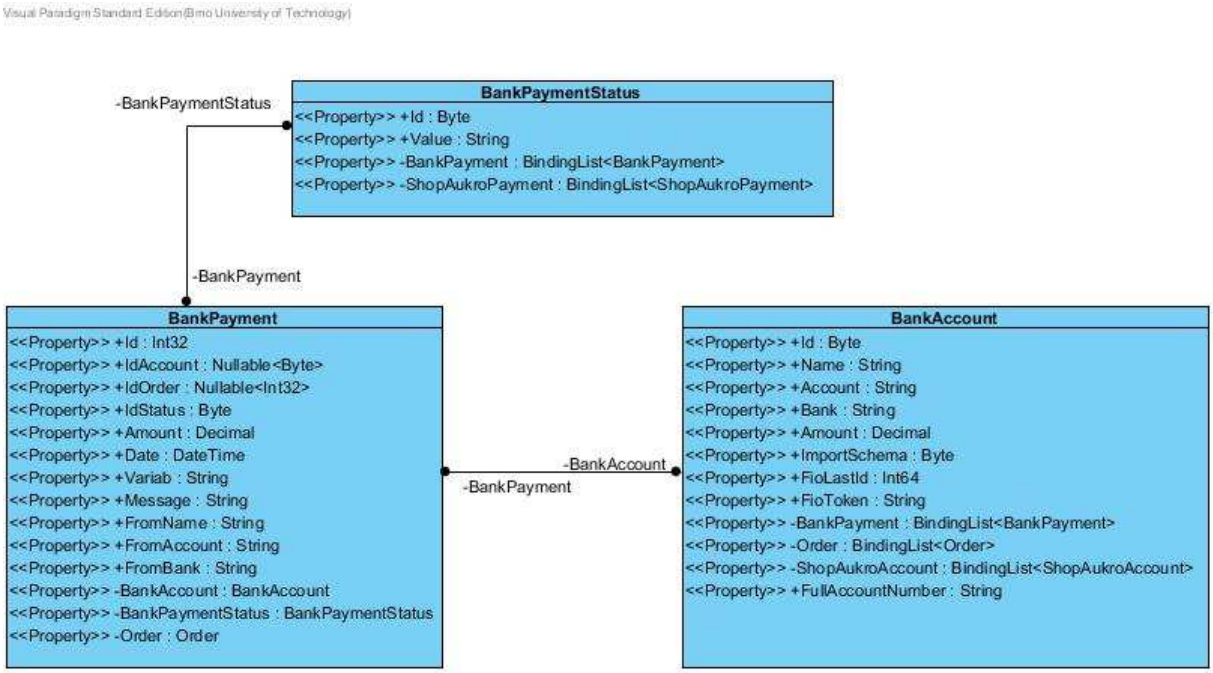
Příloha 2b – Konceptuální diagram tříd pro zboží



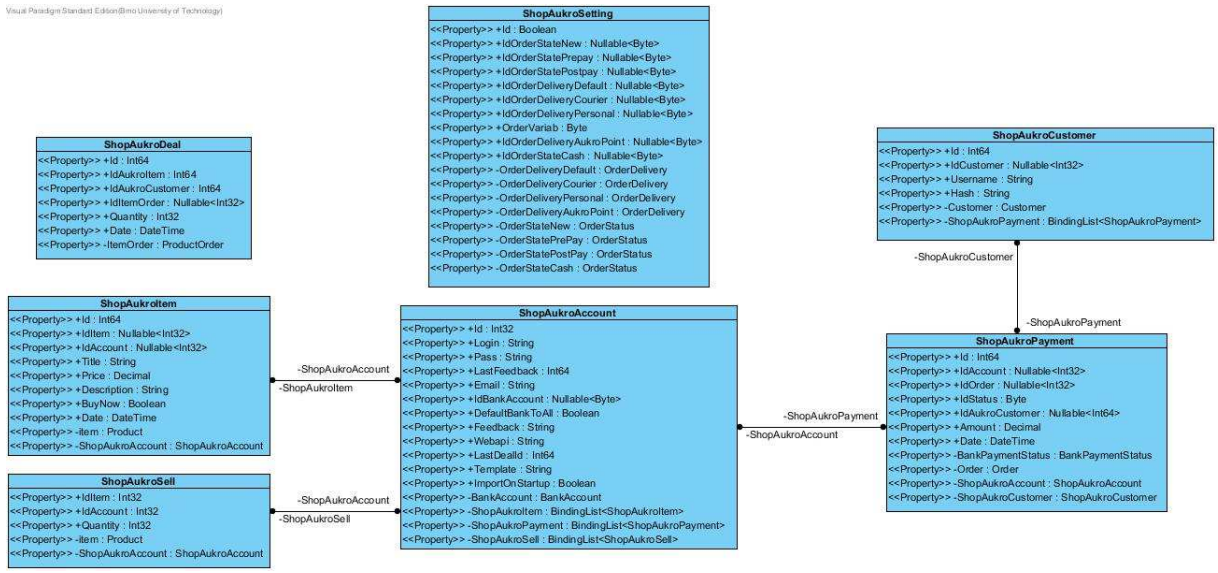
Příloha 2c – Konceptuální diagram tříd pro reklamace



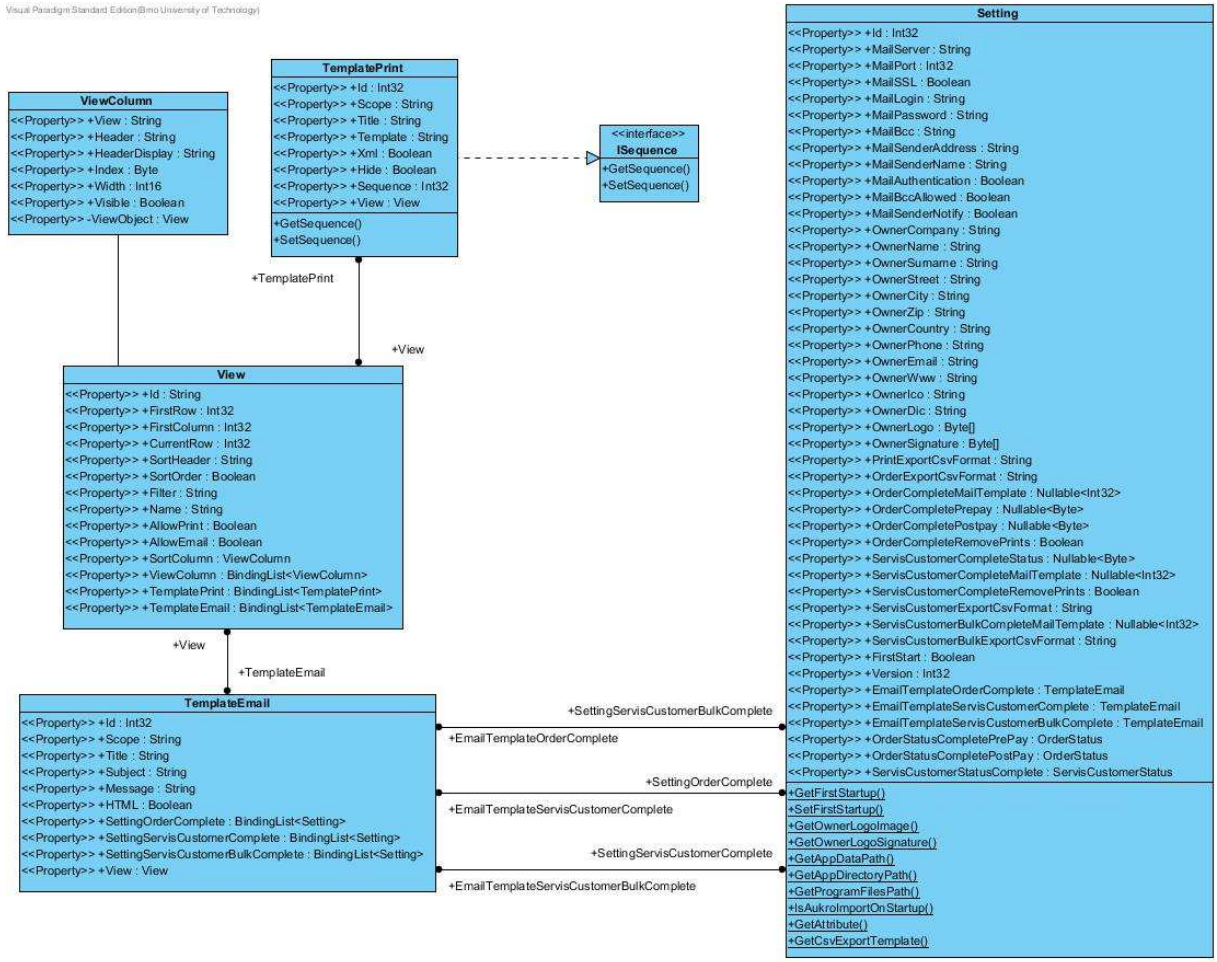
Příloha 2d – Konceptuální diagram tříd pro bankovní platby



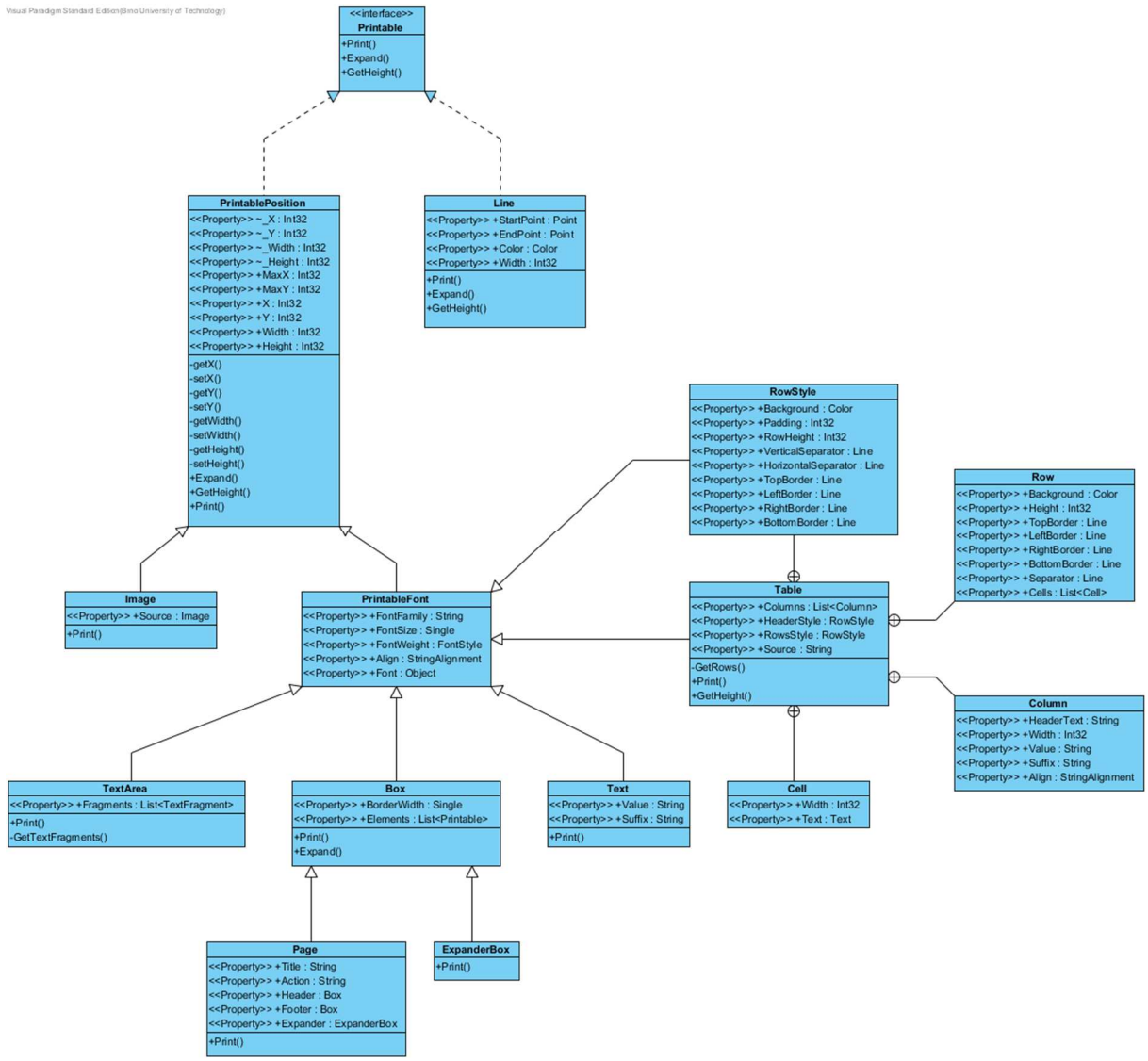
Příloha 2e – Konceptuální diagram tříd pro portál Aukro.cz



Příloha 2f – Konceptuální diagram tříd pro nastavení a pohledy



Příloha 2g – Konceptuální diagram tříd tisk dokumentů



Příloha 3 – Seznam XML elementů pro vytváření tiskových šablon

Název elementu	Povolené atributy		Povolené vnitřní elementy
Page kořenový element dokumentu	Title	Název dokumentu	Header, Footer, Expander
	Action	Akce prováděné před tiskem (např. vygenerování čísla faktury)	
Header	Stejné s elementem Box		
Footer			
Expander			
Box oblast pro vkládání dalších elementů, je základem hierarchie	BorderWidth	Šířka elementu	Text, Line, Box, Table, Image, Textarea
	FontSize	Velikost písma	
	Font	Typ písma	
	FontWeight	Styl písma: {bold|italic|normal}	
	Align	Zarovnání textu	
	Positon	Pozice: souřadnice X;Y	
	Size	Velikost: šířka;výška	
	Height	Šířka	
	Width	Výška	
Line čárový oddělovač	Start	počáteční bod	-
	End	koncový bod	
	Width	šířka čáry	
Text textová hodnota	Value	zobrazovaný text	-
	Suffix	textové zakončení (např. „ Kč“)	
	FontSize	Velikost písma	
	Font	Typ písma	
	FontWeight	Styl písma: {bold|italic|normal}	
	Align	Zarovnání textu	
	Positon	Pozice: souřadnice X;Y	
	Size	Velikost: šířka;výška	
	Height	Šířka	
	Width	Výška	
Image vložený obrázek	Source	Zdroj obrázku: {“[owner_logo]“| ”[owner_signature]“}	-
	Positon	Pozice: souřadnice X;Y	
	Size	Velikost: šířka;výška	
	Height	Šířka	
	Width	Výška	
Textarea textová oblast, dovoluje vložit formátovaný text	FontSize	Velikost písma	-
	Font	Typ písma	
	FontWeight	Styl písma: {bold|italic|normal}	
	Align	Zarovnání textu	
	Positon	Pozice: souřadnice X;Y	
	Size	Velikost: šířka;výška	
	Height	Šířka	
	Width	Výška	

Table tabulkové zobrazení položek	Source	Zdroj dat	TableHeader, TableRows, TableColumns
TableHeader definice stylu hlavičky tabulky	Background	Barva pozadí	TableLine
	Padding	Odsazení od kraje	
	FontSize	Velikost písma	
	Font	Typ písma	
	FontWeight	Styl písma: {bold|italic|normal}	
	Align	Zarovnání textu	
	Positon	Pozice: souřadnice X;Y v bodech	
	Size	Velikost: šířka;výška v bodech	
	Height	Šířka	
	Width	Výška	
TableRows definice stylu řádků tabulky	FontSize	Velikost písma	TableLine
	Font	Typ písma	
	FontWeight	Styl písma: {bold|italic|normal}	
	Align	Zarovnání textu	
	Positon	Pozice: souřadnice X;Y v bodech	
	Size	Velikost: šířka;výška v bodech	
	Height	Šířka	
	Width	Výška	
TableColumns	-		TableColumn
TableLine ohraničení buněk tabulky	Type	Typ čáry: {border, separator}	-
	Orientation	Orientace čáry: {top, left, right, bottom, horizontal, vertical}	
TableColumn definice obsahu sloupce tabulky	HeaderText	Text v hlavičce sloupce	-
	Width	Šířka sloupce	
	Value	Typ hodnoty ve sloupci	
	Suffix	Textový dodatek k hodnotám	
	Align	Zarovnání ve sloupci	

Příloha 4 – Příklad emailové šablony

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/1999/REC-html401-19991224/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=cp1250">
  <title>Vortex computers</title>
</head>
<body style="background-color: #003366;">
<table align=center style="font-family: Tahoma, Arial; font-size: 10px; color: #FFF;">
<tr>
  <td colspan="2" style="padding: 5px;">
    <a href="http://www.vortexcomp.cz/" title="Vortex computers"><img src="http://www.vortexcomp.cz/aukro/logo.png" alt="Vortex computers" style="border: none;"></a>
  </td>
</tr>
<tr>
<td colspan="2">
<table style="font-family: Tahoma, Arial; font-size: 11px; color: #333333; width: 600px; background-color: #FFF; border-radius: 5px; box-shadow: 0px 0px 5px black;">
  <tr>
    <td align="left" style="font-weight: bold; padding: 10px;">Vaše objednané zboží bylo odesláno.</td>
  </tr>
  <tr>
    <td align="left" style="border-radius: 5px; background-color: #036; color:#FFF; font-size: 12px; font-weight: bold; padding: 8px 10px;">Detaily objednávky</td>
  </tr>
  <tr>
    <td align="left" style="padding: 10px;">
      ID objednávky: <span style="color: #036; font-weight: bold;">[ID]</span><br />
      Datum objednání: [CREATE_DATE]<br >
      Způsob platby: <strong>[PAYMENT]</strong><br />
      Způsob dopravy: <strong>[DELIVERY]</strong><br />
      Číslo zásilky: <strong>[BARCODE]</strong><br />
      <br />
      E-mail: <strong>[EMAIL]</strong><br />
      Telefon: <strong>[PHONE]</strong>
    </td>
  </tr>
  <tr>
    <td>
            <table style="width: 100%; font-family: Verdana, sans-serif; font-size: 11px;">
        <tr style="text-transform: uppercase; color: #FFFFFF;">
          <td style="background-color: #036; border-top-left-radius: 5px; padding: 8px;"><b>Fakturační adresa</b></td>
          <td style="background-color: #036; border-top-right-radius: 5px; padding: 8px;"><b>Dodací adresa</b></td>
        </tr>
        <tr style="vertical-align: top;">
          <td style="padding: 8px; background-color: #EEEEEE;">[INV_COMPANY]<br />[INV_NAME] [INV_SURNAME]<br />[INV_STREET]<br />[INV_ZIP] [INV_CITY]<br />[INV_COUNTRY]</td>
          <td style="padding: 8px; background-color: #EEEEEE;">[POST_COMPANY]<br />[POST_NAME] [POST_SURNAME]<br />[POST_STREET]<br />[POST_ZIP] [POST_CITY]<br />[POST_COUNTRY]</td>
        </tr>
      </table>
    </td>
  </tr>
  <tr>
    <td style="padding-top: 15px; padding-bottom: 30px;">
            <table style="width: 100%; font-family: Verdana, sans-serif; font-size: 11px;">
        <tr style="background-color: #036; color: #FFFFFF;">
          <th align="left" style="padding: 5px 8px;">Zboží</th>
          <th align="right" style="width: 15%; padding: 5px 8px;">Cena s DPH</th>
          <th align="right" style="width: 15%; padding: 5px 8px;">Množství</th>
          <th align="right" style="width: 20%; padding: 5px 8px;">Celkem s DPH</th>
        </tr>
<casid_repeat source="items">
```

```
        <tr style="background-color: #EEEEEE; text-align: center;">
          <td align="left" style="padding: 5px;">[TITLE]</td>
          <td align="right" style="padding: 5px;">[PRICE] Kč</td>
          <td align="right" style="padding: 5px;">[QUANTITY]</td>
          <td align="right" style="padding: 5px;">[SUM] Kč</td>
        </tr>
</casid_repeat>
        <tr style="text-align: right;">
          <td colspan="2"> </td>
          <td     style="background-color:     #EEEEEE;     font-weight:     bold;     padding:
5px;">Poštovné:</td>
          <td style="background-color: #EEEEEE; padding: 5px;">[POSTAGE] Kč</td>
        </tr>
        <tr style="text-align: right;">
          <td colspan="2"> </td>
          <td style="background-color: #EEEEEE; font-weight: bold; padding: 5px;">Celkem:</td>
          <td style="background-color: #EEEEEE; padding: 5px;">[PRICE] Kč</td>
        </tr>
      </table>
    </td>
  </tr>
</table>
</td>
</tr>
<tr>
  <td align="left">Tento email byl automaticky odeslán programem <a href="http://www.casid.cz/"
style="color: #FFF; text-shadow: 0.5px 0.5px 2px black; font-weight: bold; text-decoration:
none;">Casid - správa objednávek</a>.</td>
  <td align="right"><a href="http://www.vortexcomp.cz/" style="color: #FFF; text-shadow: 0.5px
0.5px 2px black; font-weight: bold; text-decoration: none;">Vortex computers</a> © 2015</td>
</tr>
</table>
</body>
</html>
```