VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

Fakulta informačních technologií
Faculty of Information Technology

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

Brno, 2017 Jakub Filípek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS++

WEBOVÝ INFORMAČNÍ SYSTÉM PRO VINAŘSKOU FIRMU

WEB INFORMATION SYSTÉM FOR WINE COMPANY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jakub Filípek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Pavel Očenášek, Ph.D.
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20209/2016/xfilip34

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Filípek Jakub**
Obor: Informační technologie
Téma: **Webový informační systém pro vinařskou firmu**
Web Information System for Wine Company
Kategorie: Web

Pokyny:

1. Seznamte se s webovými technologiemi e-shopů.
2. Analyzujte požadavky na vlastní e-shop prodávající vinařské produkty.
3. Navrhněte vlastní e-shop podporující: administraci uživatelů a produktů, administraci článků a fotografií, šablonovací systém pro generování faktur, automatizaci objednávek atd. Pokuste se e-shop napojit na některý z existujících ekonomických systémů.
4. Navržený e-shop implementujte dle instrukcí vedoucího práce.
5. Implementovaný e-shop otestujte na reálných datech.
6. Zhodnoťte realizovanou aplikaci a navrhněte další rozšíření.

Literatura:

- Liu Bing, Totty Brian. Web data mining: exploring hyperlinks, contents, and usage data. Springer, 2007. ISBN 978-3-540-37881-5.
- Gourley David, Totty Brian. HTTP: the definitive guide. O'Reilly, 2002. ISBN 15-659-2509-2.
- Mitchell Ryan, Holmes James. Instant web scraping with Java, Packt Publishing, 2013. ISBN 978-184-9696-883.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Očenášek Pavel, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Hlavním cílem této bakalářské práce je analýza, návrh a implementace webového informačního systému pro vinařskou firmu. Tento informační systém umožňuje internetové obchodování, administraci webového portálu a propojení s ekonomickým systémem. Další součástí je webové uživatelské rozhraní. Informační systém je navržen, jako webová aplikace. Pro webové uživatelské rozhraní jsou využity technologie: HTML, CSS, JSP, JavaScript. Logická část aplikace je implementována v jazyku Java 7 s využitím frameworku Spring. Pro databázi je využita technologie PostgreSQL. Webová aplikace je připravena pro spuštění na serveru Apache Tomcat.

Abstract

The aim of this bachelor thesis is to analyze, design and implement a web information system for a wine company. This information system enables internet trading, administration of the web portal and connection with the economic system. Another part is the web user interface. The information system is designed as a web application. Technologies used for frontend are: HTML, CSS, JSP, JavaScript. The backend part of the application is implemented in Java 7 using the Spring framework. The database uses PostgreSQL technology. The web application is ready to run on Apache Tomcat server.

Klíčová slova

Informační systém, internetový obchod, webová aplikace, Java, HTML, CSS, databáze, Spring Framework, PostgreSQL, Apache Tomcat

Keywords

Information system, Eshop, web appliction, Java, HTML, CSS, database, Spring Framework, PostgreSQL, Apache Tomcat

Citace

Filípek Jakub: *Webový informační systém pro vinařskou firmu*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Očenášek Pavel

Webový informační systém pro vinařskou firmu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Pavla Očenáška, Ph.D. Další informace mi poskytl majitel společnosti Vinařství Gertner pan Marek Gertner. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……..

Jakub Filípek

17. května 2017

Poděkování

Chtěl bych poděkovat majiteli společnosti Vinařství Gertner panu Marku Gertnerovi, který mi poskytl zadání práce a pomoc při analýze a testování. Dále bych chtěl poděkovat mému vedoucímu panu Ing. Pavlu Očenáškovi, Ph.D. za odborné konzultace a také za poskytnutí užitečných rad při návrhu informačního systému.

Obsah

1 Úvod

Cílem této bakalářské práce je navrhnout webový informační systém pro vinařskou firmu. Součástí je také implementace daného návrhu. Výsledný systém by měl sloužit zejména k prodeji vinných produktů a zároveň také majiteli usnadnit prodej a správu objednávek. Snahou práce je také systém co nejvíce automatizovat. Výsledný návrh a implementace by měli vycházet z analýzy požadavků zákazníka.

Obsah dokumentu je rozdělen do několika částí. V druhé kapitole je uvedena již zmíněná analýza a studium podobných internetových obchodů, kde je popsán průzkum konkurenčních portálů. Kromě průzkumu je v analýze popsán kompletní seznam požadavků na systém. Tyto požadavky byly předem konzultovány s majitelem vinařství a výsledný systém je musí splňovat. V další kapitole jsou informace o zvolených technologií, které se budou používat při implementaci.

Čtvrtá kapitola se pak dále zabývá návrhem informačního systému. Součástí tohoto návrhu je databázový model ERD a model případů užití (Use case diagram). Kromě databázového návrhu je popsána i integrace s ekonomickým systémem Pohoda. Dále také administrace systému a návrh uživatelského rozhraní.

V páté kapitole se práce zabývá implementací. V implementaci je pak podrobněji popsána architektura výsledné aplikace a její jednotlivé komponenty, které usnadňují práci při vývoji. Nechybí zde ani popis využitých knihoven.

V posledních kapitolách je popis procesu testování, jak průběžného, tak i testování výsledné aplikace. V průběhu práce na celém systému vznikly i možnosti na rozšíření, které jsou popsány v poslední kapitole.

2 Analýza

Tato kapitola se zabývá zpracováním a analýzou, která je vytvořena na základě požadavků zákazníka.

Analýza nám slouží k lepšímu pochopení implementovaného systému. Před samotným vývojem je nezbytné se ohlédnout nad přesnými požadavky na systém, aby se zamezilo chybám a nejasnostem při vývoji.

2.1 Studium podobných internetových obchodů

Vlastním průzkumem internetových obchodů, které nabízí víno, lze zjistit, že doposud většina vinařů nevyužívá jako primární zdroj prodeje webové stránky. Inspirace pro tuto bakalářskou práci pramenila zejména z všeobecných internetových obchodů, které nabízí různé produkty.

Většina z těchto zkoumaných portálů obsahuje komponentu nákupní košík, která je vždy dostupná ze všech stránek a zobrazuje množství produktů v košíku, případně celkovou cenu. Dále je také zákazníkovi umožněn přístup do tohoto košíku, kde je podrobněji uveden celkový nákup, případně objednávkový formulář pro vyřízení objednávky. Na těchto stránkách je také možnost odebírat nebo měnit množství produktů.

Objednávkové formuláře u všech zkoumaných obchodů obsahují jméno a příjmení, adresu a kontaktní údaje zákazníka, způsob dopravy a souhlas s obchodními podmínkami, který je nutný uvádět dle zákona o ochraně spotřebitele (zákon č. 634/1992 Sb.) a občanského zákoníku (zákon č. 89/2012 Sb.).
Občanský zákoník a Zákon o ochraně spotřebitele stanovuje okruh povinností, které je povinen prodejce dodržovat při právním styku se spotřebiteli.

Některé internetové obchody umožňují registraci zákazníka. Po registraci se zákazník může přihlásit a při nákupu pak nemusí vyplňovat objednávkový formulář. To slouží zejména pro zákazníky, kteří nakupují v daném internetovém obchodu častěji.

Přidávání produktů do košíku je u většiny portálů rozdílné. Internetové obchody, které se zabývají čistě prodejem, nabízí své zboží na úvodní stránce. Další z mnoha rozdílů je v celkové nabídce. Některé obchody mají vetší sortiment různého typu, a proto mají spoustu filtrů, aby se zákazník snadno dostal k tomu, co hledá. Internetové obchody, které nabízí víno, ve většině případů mají podstránku nazvanou „Naše vína“. Na této stránce je obvykle nabídka s informačním popisem a možností přidávat produkty do košíku.

Při dokončení objednávky je obvykle vygenerována e-mailová zpráva se shrnutím nákupu a s doručovacími, případně fakturačními údaji. Tato zpráva je pak zaslána zákazníkovi. Podobná zpráva se zasílá i prodejci. Po odeslání objednávek je už studium technologií složitější, z pochopitelných důvodů je přístup k administraci chráněný. Zde je tedy kladen důraz na požadavky zákazníka.

2.2 Analýza požadavků

Požadavek je něco, co musí systém pro uživatele umět vykonat. Může to být specifická funkce, kterou uživatel vyžaduje a systém poskytuje. Přidáním požadavku se typicky vývoj softwaru prodlouží, odebráním zkrátí [1].

Při analýze je nejdůležitější, aby poskytovatel přesně porozuměl požadavkům zákazníka. Tímto se ve většině IT firem zabývá analytik, což je subjekt, který má klíčovou roli při zpracovávání požadavků a přípravě modelů, tedy schémat realizace dalšího postupu při vývoji – informačních systémů, jednoduše řečeno se jedná o prostředníka mezi vývojářem a zákazníkem. Při analýze se často využívají různé tipy diagramů, které slouží vývojářům a pro zákazníka jsou nepodstatné [1].

2.3 Požadavky zákazníka

Požadavky a kritéria na konečný webový systém po konzultaci se zákazníkem se mohou rozdělit do následujících bodů:

- Přehledná nabídka produktů s možností vložit produkt do košíku.
- Možnost registrace a přihlášení uživatelů do systému.
- Dynamický design.
- Možnost administrace:
    - přidávání a editace aktualit,
    - přidávání fotografií do galerie,
    - administrace uživatelů,
    - administrace produktů,
    - výpis a potvrzení objednávek.
- Propojení internetového obchodu se systémem Pohoda.
- Generování faktur.
- Automatizace objednávek.

Tyto požadavky vznikly na základě konzultace se zákazníkem a finální systém by je měl splňovat. Každý z těchto požadavků je možné rozdělit do větších detailů.

Přehlednou nabídkou produktů je myšlen seznam zboží, které bude pomocí internetového obchodu prodáváno. V případě této práce se jedná o víno. Seznam bude ve formě tabulky a musí zobrazovat všechny informace o daném produktu. Dalším požadavkem je, aby po najetí myši na produkt byl zobrazen detail s fotkou a popisem.

Možnost registrace a přihlášení uživatelů do systému má sloužit zejména pro koncové uživatele, kde registrovaní budou mít výhodu v tom, že při objednání produktu nemusí vyplňovat objednávkový formulář a postačí jim pouhé přihlášení. Po přihlášení uživatele bude objednávkový formulář automaticky vyplněn údaji, které byly zadány uživatelem při registraci.

U dynamického designu je požadováno, aby byl použit ve značné míře a působil na uživatele příjemným a také seriózním dojmem. Konkrétně je to například banner na úvodní stránce nebo interaktivní fotogalerie.

Možnost administrace zahrnuje více položek. Všechny tyto položky se musí týkat pouze jedné role uživatele a tou je administrátor. V této roli budou další rozšířené možnosti pro práci se systémem.
Jednou z nich je přidávání a editace aktualit. Pro tuto část bude na webu speciální rubrika pro zobrazování aktuality v podobě článků, které může přidávat pouze administrátor. Při přihlášení bude mít administrátor navíc od obyčejného uživatele editační okno k článkům. Zde je požadováno, aby šly články přidávat, upravovat a případně mazat. Podobně to bude u fotogalerie.

Administrace uživatelů by měla zobrazovat seznam všech uživatelů systému. Tyto uživatele bude možnost editovat, přidávat aktivitu, případně je odstraňovat ze systému.

Další požadovanou součástí je administrace produktů. Stejně, jako u uživatelů zde bude možnost editace, přidávání a odstraňování produktů z nabídky.

Výpis a potvrzení objednávek by mělo sloužit administrátorovi pro přehled uskutečněných objednávek, kde se budou zobrazovat informace o zákazníkovi, jako například jméno, adresa, telefon a další údaje včetně nákupu daného zákazníka. Administrátor dostane práva na potvrzení a odeslání zásilky a tím se zašle zákazníkovi informační e-mail. Všechny tyto objednávky se budou současně přeposílat do e-mailové schránky administrátora.

U webové aplikace dojde také k propojení s ekonomickým systémem Pohoda. Při potvrzení objednávky se veškeré informace uloží, jak do databáze, tak i do systému Pohoda. V tomto systému budou využity funkce pro vytváření objednávek a celkové propojení internetového obchodu. Administrátor bude mít možnost kontrolovat objednávky a zároveň je spravovat, jak ze svého informačního systému, tak i z programu Pohoda.

3 Webové technologie

U dnešních webových portálů je kladen důraz na dynamičnost a maximální funkcionalitu celého webu. Z toho důvodu jsou v práci zvoleny oblíbené a také často používané technologie, které jsou uvedeny níže. Hlavním aspektem je, aby tyto technologie šly bez problémů společně propojit a umožňovaly komunikaci frontendu s backendem a databází.

3.1 Webové uživatelské rozhraní

Webové uživatelské rozhraní je dále popisováno jako frontend a pochází z oblasti programování webových aplikací, kde slouží k označení části webu viditelné běžným návštěvníkům. U internetového obchodu může být například frontend katalog zboží, nákupní košík a objednávkový formulář [16].

K administraci webové aplikace slouží opak frontendu, který se nazývá backend. Zde se určuje a ovlivňuje obsah, který pak frontend zobrazuje. Na rozdíl od backendu, frontend bývá většinou mnohem propracovanější ze všech stránek, zejména z hlediska přístupnosti, použitelnosti a vzhledu [16].

3.2 JavaServer Pages (JSP)

JavaServer Pages (JSP) je technologie pro vývoj webových stránek, které podporují dynamický obsah, který pomáhá vývojářům v jazyce Java do HTML stránek, s využitím speciálních JSP tagů, z nichž většina začínají <% a končí %>. Komponenta JSP stránky je typ Java servlet, který je navržen tak, aby plnil roli uživatelského rozhraní pro webovou aplikaci Java. Weboví vývojáři píšou JSP jako textové soubory, které kombinují s HTML nebo XHTML kódy. JSP může sbírat informace od uživatelů prostřednictvím různých forem: webové stránce přítomných záznamů z databáze nebo z jiného zdroje a vytvořit webové stránky dynamicky [10].

3.3 PostgreSQL

PostgreSQL je relační databázový systém s otevřeným zdrojovým kódem. Má za sebou více než 15 let vývoje a zakládá si na spolehlivost a na bezpečnost. Funguje pod rozšířenými operačními systémy včetně systémů Windows a Linux. Hlavní nevýhodou je malá rozšířenost na hostingových serverech [11].

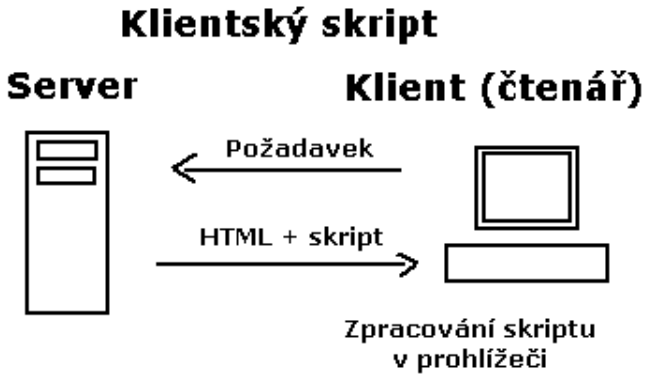
3.4 CSS

CSS (Cascading Style Sheets) neboli česky „kaskádové styly“ se používají k formátování webových stránek napsaných v HTML, XHTML a XML. Pomocí CSS stylů můžeme navrhnout design celého webu a ovládat ho jedním souborem. Tím zpřehledníme kód stránek. Tento způsob je efektivnější, než každou stránku upravovat zvlášť, kde většinou hrozí riziko, že při upravování desítek stránek můžeme snadněji udělat chybu a tudíž nebudou vypadat stejně. CSS nahrazuje používaný tag v HTML `<font>` za `<style>`, kterým kromě upravování fontu, velikosti i barvy textu dále umožňuje definovat rámce, neviditelný text, rolovací lištu nebo pozadí. Nevýhodou CSS je to, že vykreslování stránek v prohlížečích se může navzájem lišit [12].

3.5 JavaScript

JavaScript je programovací jazyk, který se používá pro vývoj uživatelského rozhraní internetových stránek. Umožňuje vytvářet interaktivní prvky uživatelského rozhraní. Implementuje se přímo do HTML kódu [14].

JavaScript je skript ze strany klienta. Program odesílá se stránkou do prohlížeče tedy na klienta a teprve tam je vykonáván. Na rozdíl od klientských skriptů existují ještě skripty serverové, které jsou vykonávány na serveru, na klienta pak jdou už jen samotné výsledky [14].



Obrázek 1: Komunikace JavaScriptu

3.6 Administrační webové rozhraní

Administrační webové rozhraní dále popisováno jako „backend“, se označuje část webové aplikace, která slouží k administraci webové aplikace nebo také ke zpracování dat. Například u redakčního systému umožňuje backend vkládání a úpravy článků, zakládání uživatelských účtů nebo dalších administrátorů nebo třeba také možnost manipulace se strukturou celého webu. U internetového obchodu backend slouží především ke vkládání nových produktů a k úpravám jeho vlastností, hlavně cen a dostupností [17].

Pojem backend pochází z anglických slov *back* a *end*, backend znamená něco jako „zadní část“. Koncept backend --> frontend je nejčastější schéma na základě kterého se dnes staví webové aplikace s dynamickým obsahem [17].

Backend této práce je psán v jazyce Java Enterprise Edition, s použitím technologie Spring Framework.

3.7 Java

Java je jedním z nejdůležitějších a nejrozšířenějších počítačových programovacích jazyků na světě, neboť jde o přední jazyk Internetu. Být profesionálním webovým vývojářem dnes znamená ovládat jazyk Java [6].

Jazyk Java je jednoduchý, má stručnou kompaktní sadu rysů, jež usnadňují jeho osvojování a používání. Ztělesňuje moderní objektově orientovanou filozofii programování [6].

Java podporuje kód nezávislý na platformě pomocí bajtového kódu Javy. Bajtový kód Javy je vysoce optimalizovaný na rychlost provádění. Není svázaná s architekturou určitého stroje či operačního systému [6].

3.8 Java EE

Java EE (Java EE) je standard v komunitním řízení podnikového softwaru. Java EE je vytvořena pomocí Java Community Process s příspěvkem od odborníků v průmyslu, komerčních a open source organizací, cílových skupin a také bezpočtem jednotlivců. Každá verze integruje nové funkce, které se ztotožní s potřebami průmyslu, zlepšuje přenositelnost aplikací a zvyšuje produktivitu vývojářů [7].

Aplikace na platformě Java EE jsou vyvíjeny jako API [1] a dalších fragmenty definované v jednotlivých specifikacích. Pro běh těchto aplikací slouží tzv. Aplikační Server. Tyto aplikační servery jsou dodávány různými dodavateli. Aplikace je tedy provozu schopná na jakémkoliv aplikačním serveru kteréhokoliv dodavatele, který implementuje příslušnou verzi specifikace nazývané také jako koncept přenositelnosti. Většina aplikačních serverů doplňuje některé vlastnosti nad rámec specifikace a aplikace využívající těchto vlastností poté neumožňují přenositelnost [7].

3.9 Spring Framework

Framework Spring poskytuje komplexní programování a konfigurace modelu pro moderní javovské podnikové aplikace - na jakékoliv platformě pro nasazení. Klíčovým prvkem springu je infrastrukturní podpora na úrovni aplikace: Spring se zaměřuje na "instalaci" podnikových aplikací, takže týmy se mohou soustředit na obchodní logiky na úrovni aplikací, bez zbytečné vazby na specifickém prostředí pro nasazení [8].

3.10 Hibernate Framework

Nástroj Hibernate[2] se používá k zajištění **perzistence** dat, což je vlastnost umožňující uchovávat data nebo informace, i když program skončí.

Hibernate je vysoce výkonný objektově relační nástroj pro perzistenci objektů v aplikaci a správu databázových dotazů nad platformou jazyka Java. Umožňuje vývoj persistentních objektů v jazyce Java, splňující základní vlastnosti, jako jsou asociace, dědičnost, polymorfismus, kompozice a práce s kolekcemi objektů [13].

Hibernate odstraňuje generování kódu v době překladu systému nebo zpracování bitového kódu. Místo toho jsou použity principy reflexe a generování bitového kódu za běhu programu, kdy generování SQL dotazu se provádí až v době startu systému. Tento přístup zajišťuje, že neovlivňuje překlad nebo inkrementální kompilaci [13].

Hibernate obsahuje i vlastní typový systém, který je mapován na specifické datové typy databázových strojů. Přesto není omezen pouze na své základní typy, ale jeho typy mohou být mapovány i na jednoduché typy a primitivy jazyka Java (například String, Char nebo Float), zahrnující třídy z

[1] API - Označuje v informatice rozhraní pro programování aplikací.
[2] Hibernate – Více informací k frameworku Hibernate na: http://hibernate.org/ .

rámcového systému kolekcí jazyka Java. Tyto typy mohou být mapovány jako hodnoty, kolekce hodnot nebo asociace k jiným entitám. Identifikační atribut persistentních objektu (id) je zvláštním typem, který reprezentuje databázové identifikátory daných tříd. Tyto identifikátory odpovídají v databázovém prostředí primárním klíčům [13].

3.11 Apache Tomcat

Apache Tomcat [3]slouží jako webový server. Je vyvíjen jako otevřený software. Jeho implementace je v jazyce Java. Základ je založen na javovských servletech Java Server Pages a Enterprise JavaBeans. Srovnávat ho můžeme s komerčními produkty, například "WebSphere Application Server" od IBM nebo "WebLogic Application Server" od Oracle a další. Tyto komerční produkty na rozdíl od Tomcatu poskytují větší robustnost a lépe propracovanou bezpečnost. Oproti tomu Tomcat vyniká svou jednoduchostí, transparentností a má menší náročností na výpočetní výkon a proto se v něm také rychleji vyvíjí [9].

[3] Více o Apache Tomcat na: http://tomcat.apache.org/ .

4 Návrh řešení

V této kapitole je popsán koncepční návrh celého informačního systému. Součástí tohoto návrhu je diagram případu užití, návrh databáze, propojení s ekonomickým systémem a administrace portálu. Poslední částí této kapitoly je popis návrhu webové prezentace.

4.1 Návrh databáze

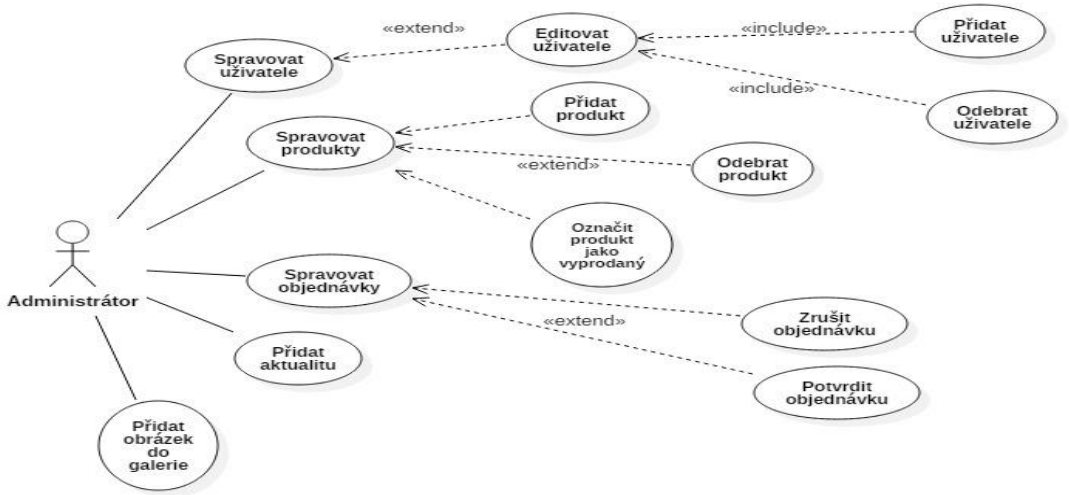
U návrhu databáze bylo nutné nejdříve zajistit strukturu ukládání informace. Pro návrh této struktury bylo použito již zmíněné ERD, neboli diagram tříd. Z analýzy požadavků je patrné, že zákazník bude chtít spravovat produkty, uživatele a jednotlivé objednávky.

V návrhu databáze hrály také roli případy užití aktérů systému. Bylo nutné si stanovit dvě role, které budou systém využívat, případně měnit jeho chování. K porozumění tohoto problému byl využit diagram případů užití.

4.1.1 Diagram případů užití

Diagram případu užití vychází z jazyka UML a reprezentuje interakci aktéra se systémem. Těmito interakcemi jsou myšleny různé způsoby použití systému v závislosti na roli aktéra.
Diagram případů užití může identifikovat různé tipy aktérů (uživatelů) systému a různé operace, které mohou provádět. Zatímco samotný případ použití by mohl sahat do velkých podrobností o každé možnosti, diagram případů užití může pomoci poskytnout pohled na vyšší úroveň celého systému. [2]
Na obrázku. *Obrázek 2*: Ukázka use case diagram, je zobrazen úsek z diagramu, kde aktérem je administrátor. Z diagramu je jasně patrné jaké akce v systému může daný aktér provádět. Dále je zde u některých akcí závislost popsána výrazem “<<extend>>”, což podrobněji určuje operaci na které je závislá. Dalším výrazem je “<<include>>”, což značí, že jednotlivá akce zahrnuje další operace nebo je blíže specifikuje.

Diagram by měl být pochopitelný i z velké části pro zákazníka. Základem je, aby přehledně zobrazoval všechny akce a aktéry systému. Kompletní se diagram vzhledem k rozsahu práce nachází v příloze.

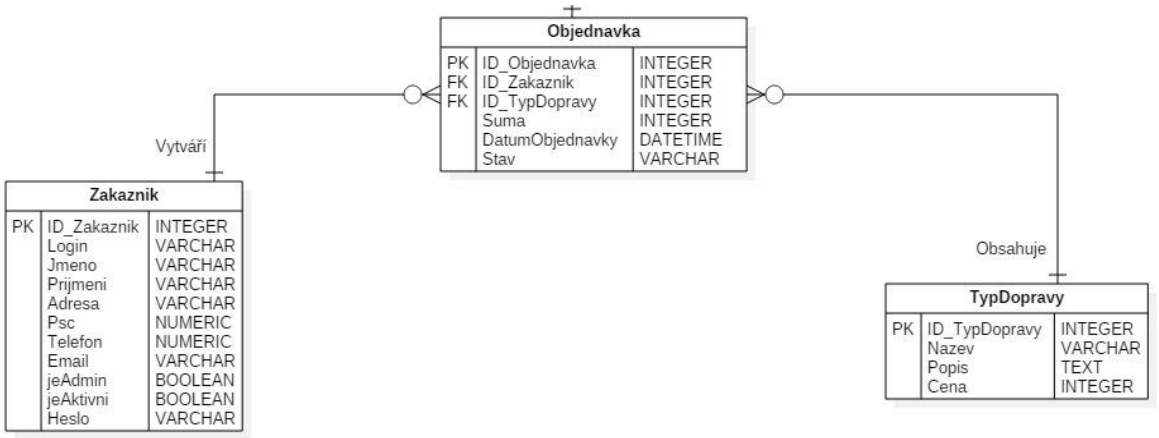


Obrázek 2: Ukázka use case diagram

4.1.2 Entitně relační diagram (ERD)

Název této modelovací techniky, která vznikla v polovině 70. let minulého století, odráží dva základní prvky vytvářeného modelu, kterými jsou entity (E - entity), vztah (R - relationship). Chápe modelovanou aplikační doménu jako množinu entit, mezi nimiž mohou existovat určité vztahy. Hlavním rysem ER modelu je, že popisuje data „v klidu“. Diagram nezobrazuje operace, které budou nad daty prováděny, jen zobrazuje jaká bude struktura uložení těchto dat [3].

Tento diagram se používá pro návrh databázových tabulek a vztahů mezi nimi. U každé tabulky jsou definovány její sloupce, které určují název sloupce, typ sloupce a případně primární nebo cizí klíč. Mezi tabulkami jsou závislosti, které mají z každé strany kardinalitu vztahu a slovní popis dané závislosti. Viz. *Obrázek 3.* Kompletní ER diagram se nachází v příloze.



Obrázek 3: Ukázka ER diagram

4.1.3 Produkty

U produktu se budou ukládat informace o názvu produktu, ceně, informačním popisu produktu a množství, které je k dispozici na skladu. Při objednávce jednoho z těchto produktů se bude množství snižovat, a pokud bude nulové, tak se produkt nastaví do režimu vyprodáno. Tento režim bude identifikovat atribut „k dispozici".

4.1.4 Zákazník

U zákazníka jsou dva druhy reprezentace, kde je to buď zákazník, který nebude vlastnit účet a tudíž se nebude moci přihlašovat k portálu a zákazník, který bude registrován a svůj účet může používat k více objednávkám. Tento zákazník bude mít výhodu, že při objednávce nebude muset vyplňovat své fakturační údaje.

Tabulka pak bude mít atributy: Login - jednoznačné přihlašovací jméno uživatele (bude jedinečné v rámci celé databáze), jméno a příjmení, adresa, poštovní směrovací číslo, telefon, e-mail a heslo. Dále atributy, které budou identifikovat, zda se jedná o administrátora či aktivního uživatele.

4.1.5 Objednávka

Objednávka bude propojovat dvě tabulky: zákazníka a produkty, kde právě zákazník objednává nějaký produkt. Tato informace bude sloužit zejména administrátorovi pro správu objednávek.

Kromě propojení těchto tabulek se budou ukládat informace o stavu objednávky. Tento stav definuje administrátor a zákazník si může zobrazit, zda už je jeho objednávka na cestě případně, jestli se vyřizuje. Dalšími atributy je suma neboli celková cena objednávky a datum objednávky.
Do této tabulky bude napojena další tabulka pro typ dopravy. Při častých změnách v typech dopravy je ideální pro snadnou editaci vytvořit speciální tabulku.

4.1.6 Ostatní položky databáze

Do databáze budou také ukládány i další položky, se kterými bude moci pracovat jen administrátor, a budou sloužit k zobrazení pro uživatele. Jednou z nich je fotogalerie.

Tabulka fotogalerie bude obsahovat atributy: Fotka ve formátu BLOB a její miniatura, komentář k fotce a informace o tom, zda bude fotka vystavena.

Další tabulka „články“ bude sloužit k editaci a vytváření článků v rubrice aktuality. Informace, které zde budou uloženy: Název článku, bude sloužit také jako nadpis, text článku, datum vytvoření a datum expirace článku. Při překročení tohoto data se nebude článek nadále zobrazovat.
Celkový návrh databáze včetně typů atributů a kardinality je v ER diagramu, který se nachází v příloze.

4.2 Ekonomický systém Pohoda

„Pohoda je ekonomický software[4], který se stará o účetnictví a daňovou evidenci podnikajících osob“[4]. Program Pohoda umožňuje posílit svoji funkci o různé programové doplňky z různých oblastí činnosti, což může být například daňové přiznání TAX, kniha jízd, hotelová rezervace nebo provoz restaurace. Dále také lze například rozšířit i o terminálový prodej, který umožňuje využívat GSM nebo WIFI čtečky a zjednodušuje tak například práci externích uživatelů. Kromě toho umožňuje vedení skladového hospodářství. Toto skladové hospodářství lze propojit s e-shopy. Propojení eshopu probíhá pomocí XML komunikace webového systému s programem Pohoda [4].

Všechny zakoupené licence na software Pohoda, TAX, PAMICA, GLX automaticky obsahují legislativní a technický upgrade na rok 2017, včetně stále dostupné podpory. Výrobcem ekonomického programu Pohoda je společnost Stormware s.r.o. [4].

Program Pohoda pro svoji komunikaci využívá rozhraní XML[5], které umožňuje přijímat i odesílat data z různých a do různých aplikací, zejména internetových obchodů a externích softwarových modulů. To slouží hlavně pro zpracování dat, které souvisí s databázemi programu Pohoda [4].

V současné době dává XML komunikace přednost před importy a exporty tabulek nebo přímým přístupem do databáze. Hlavní důvod je vyšší bezpečnost operací, protože při komunikaci pomocí XML dochází ke kontrole a validaci dat tak, jako by šlo o operace vykonávané přímo v prostředí programu. Příkladem může být import dokladů ve formátu XML, kde dochází ke kontrole vstupních údajů, dopočítání chybějících informací nebo u chybně importovaných dat také k upozornění na nutnost doplnit nebo změnit XML soubor bez jeho importování. Bezpečnost této komunikace je možné srovnat s bezpečností a možnostmi ručního zápisu dat přímo v prostředí programu Pohoda [4].

[4] Oficiální stránky k softwaru Pohoda https://www.stormware.cz/pohoda/ .
[5] XML je značkovací jazyk, který slouží pro serializaci dat.

Vývoj XML komunikace se stále zdokonaluje na rozdíl od jiných forem importů a exportů dat, výkonnost a použitelnost tak pořád stoupá. Velkou výhodou této komunikace je také její otevřenost, nezávislost a dostupnost. V případě použití transformačních XSL souborů je možná plná přenositelnost mezi libovolnými systémy. Zpracování XML komunikace lze provádět pomocí dávkového zpracování, toto zpracování plně automatizuje výměnu informací a umožňuje implementaci XML komunikaci s programy společnosti Stormware nebo i jinými společnostmi. Komunikace XML je možné zpracovávat i z prostředí programu Pohoda pomocí jednoduchého uživatelského rozhraní [5].

Všechny potřebné XSD šablony, podle kterých je možné vytvářet objekty, jsou veřejně dostupné na stránkách společnosti Stormware [5].

4.3 Návrh propojení s ekonomickým systémem Pohoda

Ekonomický systém bude propojen s vytvářeným eshopem. U programu Pohoda je možnost využít více služeb. Služba, kterou bude využívána u této práce, je import objednávek.

Program Pohoda umožňuje komunikaci v rámci XML schémat, kde u každé služby jsou atributy pro vyplnění. Po vyplnění těchto atributů se provedené změny zašlou systému Pohoda. Celá tato komunikace bude využívat knihovny jazyka Java Spring.

Komunikační klient programu Pohoda je založen na zasílání zpráv pomocí HTTP protokolu. Díky webové službě je možné rozšířit informační systém o funkcionalitu, dynamičnost a automatizaci. Webové aplikace přijímají zprávy a na tyto zprávy pak můžou odpovídat. Díky tomuto principu může program Pohoda předávat nebo získávat data z webové služby. Webová služba tyto požadavky vyhodnotí, zpracuje a následně vytvoří odpověď v XML formátu a tuto odpověď předá zpět programu Pohoda. Předávání dat probíhá pomocí metody POST [18].

4.3.1 Import dat z webové služby do systému Pohoda

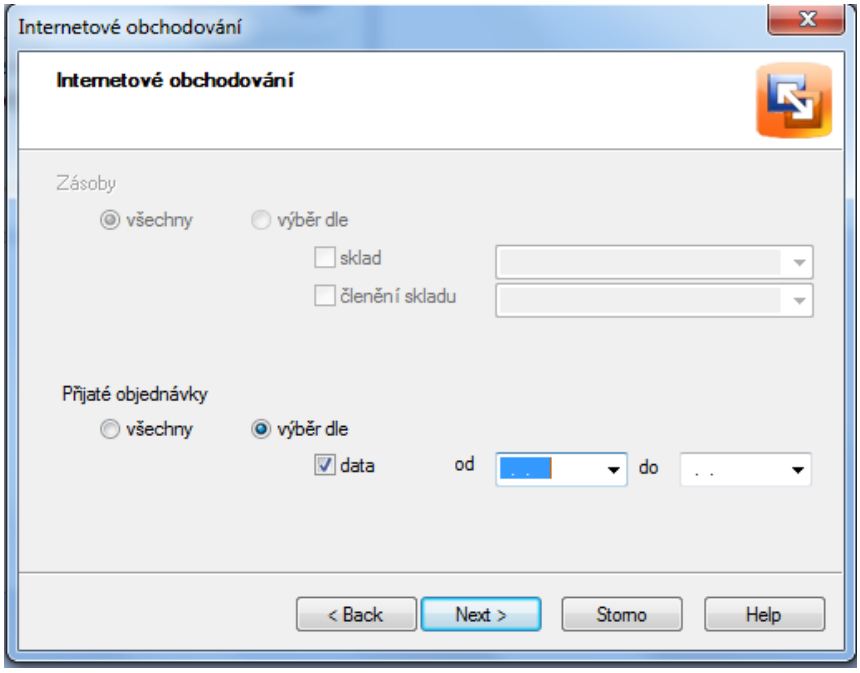
Systém Pohoda umožňuje import dat z webové aplikace. Na rozdíl od exportu je import více omezený. Importovat lze skladové zásoby, adresář a přijaté objednávky.

Jak už bylo popsáno výše, Pohoda využívá komunikace pomocí XML dotazů na server. Tyto dotazy obsahují filtry pro stažení objednávek. Je možné zasílat tři typy požadavků na data:

- Požadavek na stažení všech objednávek z Internetového obchodu

- Požadavek na stažení všech objednávek, které proběhly během data (od - do)
- Požadavek na stažení všech nových objednávek

Zaslání tohoto požadavku přímo z Pohody s možností volby filtru je znázorněno na obrázku. *Obrázek 4.*



Obrázek 4: Ukázka internetového obchodování

4.3.2 Návrh zpracování požadavku na import objednávek

Před tím než bude implementováno zpracování požadavku na objednávky, je nutné navrhnout způsob, jakým budou tyto požadavky zpracovány.

Pohoda nabízí řadu „xsd“[6] schémat, které definuji přesnou strukturu XML souborů, které jsou zasílány nebo přijímány. Na základě těchto schémat bude nutné vytvořit java třídy, aby se výsledné XML požadavky promítly do tříd a bylo možné je používat na úrovni jazyku Java. Vzhledem k velkému obsahu „xsd souborů“ je velmi obtížné vytvořit třídy správně. Pro tyto účely bude využit speciální generátor XJC, který očekává na vstupu soubory formátu „.xsd“ a na výstup vytváří třídy jazyka Java.

[6] XSD schéma je schéma, které popisuje strukturu XML dokumentu.

Po vygenerování už bude možné mapovat požadavky a odpovědi z XML -> Java; Java -> XML, pomocí knihovny JAXB[7].

4.4 Návrh administrace

Důležitou součásti informačního systému a internetového obchodu je administrace. K tomu, aby se předešlo různým problémům a administrátor mohl mít celý systém pod kontrolou, je nutné udělat správný návrh, který bude odpovídat požadavkům. Součástí tohoto návrhu je, jak bude vypadat rozhraní správy administrátora a jakým způsobem budou zajištěna práva k přístupu do administrace.

K přístupu do administrace bude sloužit speciální účet. K identifikaci tohoto účtu bude sloužit databázový sloupec, který určuje, zda se jedná o administrátora. Při přihlášení do tohoto účtu se zobrazí odkaz na stránky, kde bude zajištěna správa informačního systému.

Úvodní stránka by měla sloužit k zobrazení aktuálních informací, jako jsou například: počet registrovaných uživatelů, počet nevyřízených objednávek a množství nabízených produktů.
Další sekce budou sloužit pro administraci jednotlivých databázových tabulek a tedy modelů systému. Administrátor bude mít možnost spravovat produkty, fotogalerii, uživatele, aktuality a objednávky.
Správa aktualit musí obsahovat výpis všech aktivních aktualit, které se zobrazují uživatelům. Tyto aktuality bude pak možné zvlášť editovat nebo odstraňovat. Zajisté bude možnost přidávat nové aktuality, k čemuž bude sloužit speciální formulář. Při vytvoření aktuality se vygeneruje aktuální datum, které se zobrazí mezi titulkem a textem.

U fotogalerie to bude obdobný případ. V administraci se budou fotografie zobrazovat stejně, jako na uživatelské stránce, s tím rozdílem, že vedle každé fotky bude tlačítko pro odstranění. Dále bude na této stránce formulář pro vložení nové fotografie. Před tím než se fotografie uloží do databáze, proběhne validace a vytvoření náhledového obrázku. Do databáze se uloží název, originální obrázek a zmenšený obrázek pro náhled. Toto zmenšení bude zajištěno pomocí knihovny *java.awt.*

[7] JAXB - https://jaxb.java.net/ .

4.4.1 Administrace uživatelů

U uživatelů bude nutné ošetřit přihlašování a jejich práva na administraci a přístup. V administraci se budou zobrazovat všichni uživatelé portálu s možností editace a změny práv. Systém se má skládat ze tří typů uživatelů, a to aktivní, administrátor a neaktivní.

Každý nově registrovaný uživatel se nastaví do režimu aktivní bez práv administrátora. To znamená, že bude mít možnost se přihlašovat do systému, ale nebude mít možnost měnit stav systému. Takovému uživateli se při přihlášení předvyplní objednávkový formulář. To se zajistí tak, že při načtení košíku se současně načtou i jeho fakturační údaje z databáze.

Neaktivní uživatel bude sloužit pouze k informování majitele informačního systému. Tento uživatel se vytvoří při provedení nákupu. Vygeneruje se náhodný login a heslo a nastaví se do režimu neaktivní. Následně se tyto informace a fakturační údaje uloží do databáze.

Poslední z uživatelů je administrátor, který bude mít ta práva, jež jsou popsána v této kapitole. Jakým způsobem bude ošetřen přístup se zabývá kapitola Implementace.

4.4.2 Administrace objednávek

Hlavním cílem návrhu administrace objednávek je, aby měl prodávající celkový přehled o stavu objednávek a zároveň mohl předávat aktuální informace svým zákazníkům.

Veškeré informace po provedení objednávky se budou promítat do administrační sekce objednávky, jelikož může nastat situace, kdy administrátor dlouho nepřistupuje k systému. Je tedy potřeba ho notifikovat i jiným způsobem a to zasláním e-mailové zprávy. V emailové zprávě se zašlou fakturační údaje a obsah nákupního košíku. Současně se zašle e-mailová zpráva zákazníkovi, aby věděl, že jeho objednávka byla úspěšně dokončena, případně si ještě jednou zkontroloval vyplněné údaje.

V administraci se promítnou objednávky, které budou rozděleny do 3 sekcí: nevyřízené objednávky, odeslané objednávky a doručené objednávky. Jakmile objednávka bude zaslána kurýrem, může administrátor přepnout stav objednávky na odeslaná a tím se provede generování zprávy pro zákazníka s informací, že byl balíček předán přepravní společnosti. Tato zpráva se zašle na e-mailovou adresu, kterou zákazník dříve vyplnil. Po úspěšném doručení už nebude potřeba objednávku zobrazovat. Tomu bude sloužit poslední stav doručené objednávky.

Další součástí administrace objednávek je automatické generování faktur. Z informací uložených v databázi a pomocí knihovny *itext* bude administrátorovi umožněno, kdykoliv provést generování a stažení faktury.

4.5 Návrh internetového obchodu a nákupního košíku

Návrh internetového obchodu vychází zejména z průzkumu podobných portálů a také z osobních zkušeností nákupu po internetu. Cílem tohoto návrhu bylo zajistit správnou strukturu, jak může uživatel zacházet s nákupním košíkem a také do něj přidávat, případně měnit množství produktů. Tento návrh je ideální způsob, jak předejít problémům během implementace nebo testování, v nejhorším případě vystavení aplikace do sítě.

Aby se oddělily dvě struktury, které zákazníka provází při nákupu, byla aplikace rozdělena na výběr produktů a košík pro přehled s vyřízením objednávky. Ve výběru produktů bude potřeba položky načítat z databáze a strukturovat do přehledné tabulky. U každé položky se vloží vstup pro množství, které si může zákazník při objednávce zvolit. Aby se změny projevily v košíku, musí být v této části tlačítko pro potvrzení.

Košíkem se myslí samostatná stránka, kde bude zobrazován aktuální nákup s možností odstranění položek. Pro potvrzení nákupu je nutné získat fakturační a doručovací údaje o zákazníkovi, proto je i součástí košíku objednávkový formulář. Častým problémem při vyplňování formuláře jsou překlepy ze strany uživatele. Aby se předešlo těmto překlepům a prodávající, tak nezískal neplatné informace, bude nezbytné implementovat validátor. Důležité je také určit, které položky se budou validovat. Nejdříve se musí ošetřit, zda uživatel vyplnil všechny hodnoty. Dále jestli e-mailová adresa, telefon a poštovní směrovací číslo odpovídá správnému formátu, jako je počet, typ a rozložení znaků.

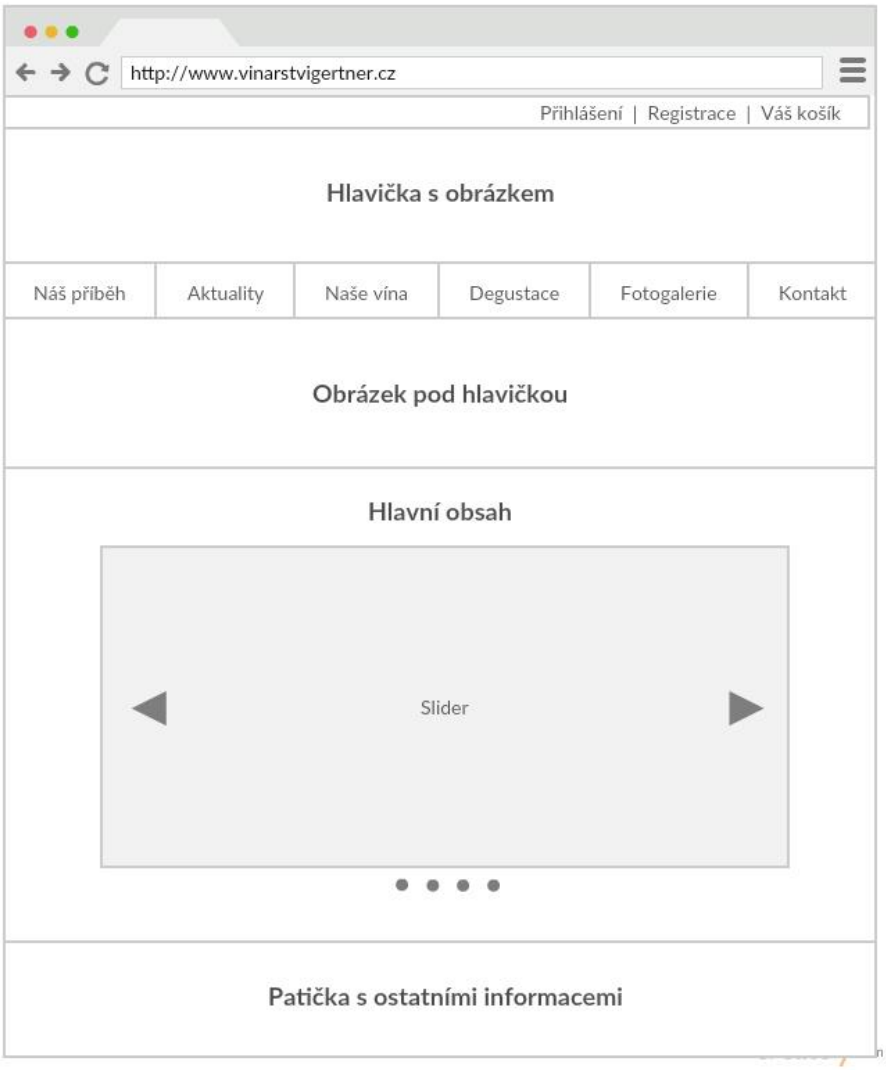
Výše popsané aspekty jsou důležitou součástí každého internetového obchodu a je dobré si je stanovit před samotnou implementací.

4.6 Návrh webové prezentace

Z analýzy požadavků jasně vyplývá, že webová stránka by měla být co nejlépe ovladatelná a přehledná pro uživatele. Také je důležité, aby vzhled stránky byl poutavý a dobře strukturovaný.

Z průzkumu ostatních webových stránek s podobnou tématikou je vhodné využít strukturování, na které jsou již uživatelé zvyklí. Například menu nabídka v hlavičce stránky, přihlašování a registrace na speciálním panelu, který je umístěn nahoře nebo celkový obsah uprostřed stránky.

Při procházení jednotlivých rubrik by měl mít uživatel vždy přístup k rubrikám ostatním, aby mohl pohodlně přecházet k tomu, co na webu hledá, případně se jednoduše vracet zpět.
Návrh webové prezentace je strukturován podle obrázku. *Obrázek 5.*



Obrázek 5: Návrh struktury rozložení stránky

Návrh struktury se skládá z těchto částí: horní panel, hlavička s obrázkem, hlavní nabídka, obrázek pod hlavičkou, hlavní obsah, patička.

Horní panel stránky umožňuje přistupovat uživateli k registraci, přihlášení a nákupnímu košíku. U košíku se bude navíc zobrazovat jeho celkový součet ceny aktuálního obsahu.
Hlavní nabídka neboli menu, slouží k přístupu a přepínání mezi rubrikami, na základě čehož se mění hlavní obsah stránky.

Poslední součástí rozložení stránky je patička, která má sloužit pouze k zobrazování aktualit, případně reklamních bannerů.

4.7 Návrh generování faktur

Na základě požadavků zákazníka je nutné do systému implementovat generátor faktur na základě objednávek. Důležitou otázkou u tohoto návrhu je, kdy se budou dané faktury generovat. Jestli při zaslání objednávky nebo se faktury současně zašlou jako příloha e-mailu a, nebo automatické vytváření v administraci, odkud je pak bude možné kdykoliv stáhnout. Vzhledem k tomu, že generování bude nějakou dobu trvat a vyřízení objednávky by pak bylo delší, tak je nejvhodnější způsob generování v administraci na vyžádání. To znamená, že u každé nevyřízené případně odeslané objednávky bude speciální tlačítko „generovat fakturu“. Při kliknutí na toto tlačítko se daná objednávka vyhledá v databázi a z potřebných informací se spustí generování faktury ve formátu PDF a výsledné PDF se zobrazí v nové záložce prohlížeče.

Na generování souborů ve formátu PDF se nabízí knihovna *itextpdf* [8].Tato knihovna umožňuje vytvářet a zpracovávat dokumenty PDF. Dále je možná sazba tabulek, obrázků. Funkcí obsahuje celou hromadu a je pro ni i dobrá podpora, proto je pro tento případ ideálním nástrojem.

[8] Více o *itextpdf* na: http://developers.itextpdf.com/ .

5 Implementace

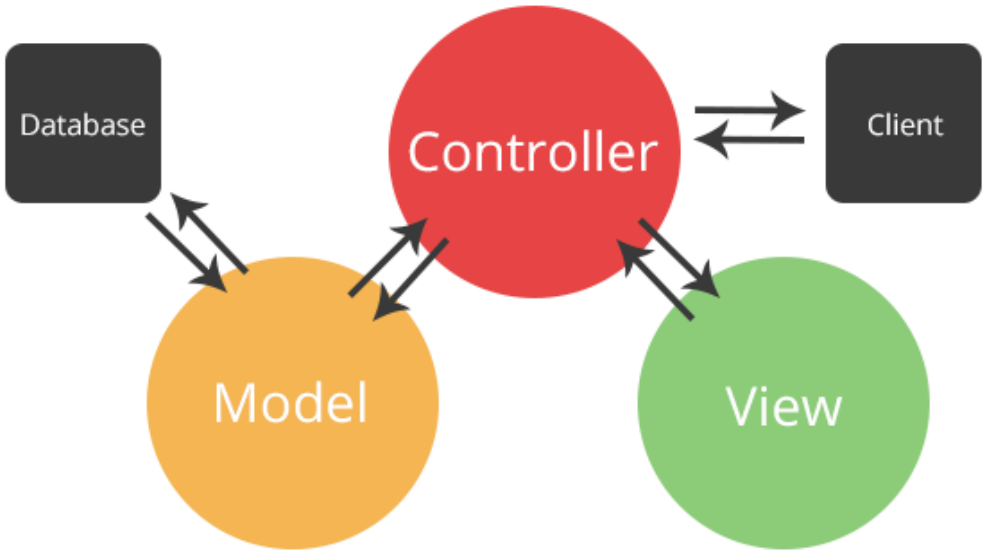
Tato kapitola pojednává o implementaci informačního systému pro vinařskou společnost. Jsou zde detailněji popsány použité technologie. Teoretický popis těchto technologií se nachází v kapitole 2. Webové technologie.

Dále se kapitola zabývá způsobem implementace a využití návrhového vzoru model-view-controller, podle kterého byl back-end informačního systému vyvíjen.

5.1 Architektura MVC

MVC je oblíbený architektonický vzor, který se používá při vývoji webových aplikací, i přestože vznikl na desktopech. Tento vzor zahrnují dnes populární webové frameworky, jakými jsou např. Zend nebo Nette pro PHP, Ruby On Rail pro Ruby nebo MVC pro ASP .NET. a Spring MVC pro Javu, který byl u práce použit. [15]

Hlavním cílem MVC architektury je oddělení logické části aplikace od výstupu. Řeší tedy problém tzv. „špagetového kódu“, kdy můžeme mít v jednom souboru (třídě) logické operace a zároveň renderování výstupu. Soubor tedy obsahuje práci nad databází, logikou a HTML, případně JSP tagy. [15]
Struktura architektury je znázorněna na obrázku. *Obrázek 6.*



Obrázek 6: Architektura MVC

5.1.1 Model

Model má na starosti veškerou logiku aplikace. Například databázové dotazy, validace, výpočty, práce se soubory a podobně. Hlavním úkolem modelu je předávat data na základě parametrů, které jsou modelu předávány. Nezajímá se o to, odkud se data v parametrech získávají a ani jakým způsobem bude probíhat reprezentace a formátování dat.

5.1.2 View (Pohled)

View nebo pohled má za úkol zobrazit data získaná z modelu na výstup pro uživatele. Příkladem může být HTML stránka s tagy značkovacího jazyka, které umožňují vkládat proměnné nebo používat řídící konstrukce, jako jsou cykly, podmíněné výrazy nebo často využívaná věc procházení seznamem. View se stejně jako Model nestará, odkud data přišla. Jeho úkolem je pouze zobrazit data pro uživatele.

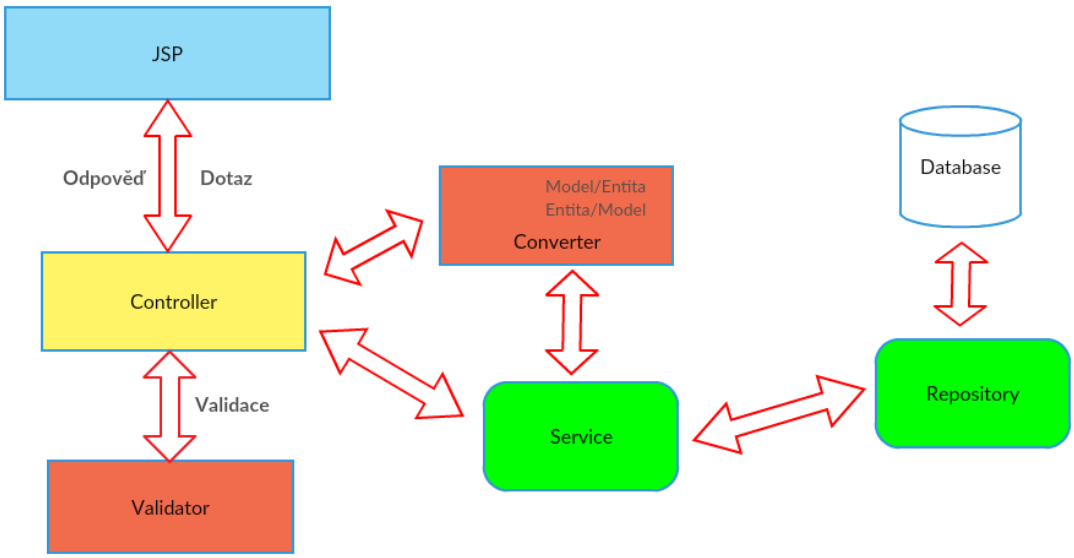
5.1.3 Kontroler

Kontroler je možné chápat jako prostředníka, který komunikuje mezi modelem, pohledem a také i s uživatelem systému. Stará se o propojení předchozích prvků a uživatele. Získává požadavky a zasílá je modelu. Model je zpracovává a předává zpět kontroleru a ten tyto data předá pohledu.

5.2 Architektura aplikace

Aby implementace odpovídala architektuře MVC, je důležité, aby byla také dodržena konvence mezi jednotlivými třídami, rozhraním a celkově balíčky. Pro tyto účely byla předem vytvořena struktura zdrojových souborů a architektura aplikace.

Systém se skládá z několika komponent, které drží vrstvu a každá z nich má jistý úkol a práva komunikovat s některými komponentami, které si mezi sebou zasílají data. Tyto data zpracují a předávají další vrstvě. Tato architektura aplikace je znázorněna na obrázku. *Obrázek 7*.



Obrázek 7: Architektura aplikace

5.2.1 Validátor

Tato komponenta slouží pro validace dat zadaná uživatelem, případně administrátorem. V momentě, kdy je zaslán vyplněný formulář, tak se ve validátoru ověřují jednotlivé položky formuláře. Tato akce probíhá na začátku vstupu do kontroléru. Tato akce zabrání případné manipulaci se špatnými daty. Jakmile nastane chyba, vypíše se chybová hláška do logu serveru a zároveň se předá pohledu do jsp, pro informování uživatele.

Validátor je implementován pomocí frameworku Spring. Každá validační třída implementuje rozhraní `Validator`. A přepisuje metodu `validate`, která je vstupní metodou pro validaci. V této metodě jsou volány metody privátní. Ty řeší už správnost jednotlivých položek a vrací hodnotu `boolean`. Tyto validátory se pak jednotlivě volají v metodách kontroléru pomocí parametru `BindingResults`. Ten zajišťuje, aby získaný model byl ihned validován.

5.2.2 Konvertor

Pro zajištění správného ukládání a strukturování dat, jsou oddělena data přijímaná uživatelem a databází. Data od uživatele jsou reprezentována jako **model.** S modelem se manipuluje v kontroléru a případně v servisních třídách. Před vstupem do repozitáře se konvertují na **entity**, které mohou být

vloženy do databáze a totéž platí i opačným směrem. Tuto konverzi z entity na model a z modelu na entitu zajišťuje komponenta konvertor.

Konvertor je rozhraní z frameworku Spring. Tomuto rozhraní jsou při implementaci předány v generických závorkách dvě třídy: zdrojová třída, ze které bude probíhat konverze a cílová, na kterou bude probíhat konverze. Implementace tohoto rozhraní je označeno anotací `@Component`, aby bylo umožněno vkládání závislostí. Přetěžovaná metoda se nazývá `convert` a v ní probíhá už samotné chování konverze. Pro konvertor je také nutná konfigurace, kde se vkládají jednotlivé konvertory.

5.2.3 Servisní vrstva

Servisní nebo také logická vrstva aplikace zpracovává veškerou logiku, jaká data se budou kdy a kam posílat. Dále také zde probíhají případné výpočty nebo manipulace s daty. Tato vrstva je něco jako prostředník mezi kontrolérem a repozitářem. Od kontroléru přijímá data ze vstupu, které následně zpracuje. Pokud je potřeba zpracovaná data uložit do repozitáře, zavolá konvertor z modelu na entitu a data předá repozitáři, jinak je může vrátit zpět kontroléru. Nebo kontrolér může o data zažádat a to pomocí volání servisní vrstvy.

Servisní vrstva je také použita z knihovny Spring, kdy každá ze servisních tříd je označena anotací `@Service` a podobně, jako u konvertoru je zde zajištěno vkládání závislostí.

5.2.4 Repozitář

Repozitář slouží pro získávání nebo ukládání dat do databáze. Je to vrstva, která je zodpovědná za práci s perzistentními daty. Každá entita má vlastní repozitář, což je v Javě, jako rozhraní s metodami pro dotazy, ukládání nebo aktualizací dat. Komunikace probíhá s databází a servisní vrstvou.

V této aplikaci je pro repozitáře vyžita knihovna *JPA Repositories* z frameworku Spring. Každý repozitář je označen anotací `@Repository` a dědí z třídy `JPARepository`, která má v generických závorkách typ identifikátoru a entitu. I zde je zajištěno vkládání závislostí.

5.3 Webová servisní vrstva

Tato vrstva zajišťuje zpracování požadavků z ekonomického systému Pohoda. Konkrétně jsou to požadavky na import objednávek z informačního systému do Pohody. Požadavky, jak již bylo zmíněno,

se zasílají metodou POST na definovanou URL adresu. O zpracování se stará rest kontroler, který na dané URL očekává požadavek ve správném XML formátu. XML data se následně mapují na Java objekty, se kterýma je pak možné pracovat na této úrovni. Aby mapování proběhlo v pořádku, byl použit speciální generátor tříd XJC. Tento generátor z definovaných xsd schémat vygeneruje třídy jazyka Java.

U importu objednávek bylo potřeba také ošetřit filtrování požadavku. Pohoda nabízí 3 typy filtrování objednávek. Jedním z nich je filtr na stažení všech objednávek, kdy se všechny objednávky z databáze přenesou do ekonomického systému. Dále je to filtr na stažení všech nových objednávek. Zde filtr obsahuje datum, od kterého se budou stahovat objednávky. A poslední je filtr, který obsahuje datum od kterého probíhá stahování objednávek a datum po které probíhá stahování objednávek.

Stažení objednávek z databáze probíhá stejně, jako u ostatních servisních vrstev. Rozdíl je v tom, že webová servisní vrstva má za úkol vytvořit odpověď v XML formátu, kterou už zpracovává program Pohoda. Pro tento případ byly vytvořeny speciální továrny, které pomocí knihovny JAXB vytváří odpověď ve formátu XML nebo JSON. Pokud se podařilo systému nalézt objednávky, tak je pro ně vždy vytvořena hlavička s informacemi, jako je datum, popis a fakturační údaje zákazníka. Pod hlavičkou je další objekt, který obsahuje už konkrétní položky, celkovou cenu a způsob dopravy. Hlavička a detail objednávky jsou pak zabaleny ještě do jednoho objektu, který očekává systém Pohoda. Tímto končí komunikace webové servisní vrstvy a ekonomického systému.

5.4 Konfigurační balíček

Aby bylo možné využívat všechny komponenty správně, je potřeba, aby byly nakonfigurovány. Všechny konfigurace řeší konfigurační balíček. Zde je zajištěno propojení s databází, konverze mezi entitami, inicializace aplikace, zabezpečení přihlašování do systému a nastavení ostatních služeb, které systém využívá.

U konfigurace databáze se nastavuje zdroj, tedy kde databáze běží. K tomu je potřeba knihovna s ovladačem JDBC [9] pro databázi PostgreSQL. Dále se nastavuje *Hibernate adaptér*. Tomuto adaptéru se předá balíček s entitami, na které probíhá mapování databázových objektů.

Důležitou konfigurací je také zabezpečení při přihlašování. Protože systém obsahuje dvě role: administrátora a zákazníka, každá z rolí má jiná práva, je potřeba tento stav zabezpečit. K tomu slouží bezpečnostní adaptér což je třída `WebSecurityConfigurerAdapter`. Tato třída nastavuje dané

[9] JDBC – rozhraní pro přístup k relačním databázím.

roli přístup k jednotlivým stránkám aplikace. Například editace uživatelů, kde má přístup pouze administrátor. Zákazníkovi zobrazí chybový kód *403 - Požadavek byl legální, ale server odmítl odpovědět*. Dále kromě práv, je zde nastaveno odhlašování uživatelů ze systému.
Při vyřizování objednávek jsou vždy zasílány e-mailové zprávy. Také zde byla nutná konfigurace, kterou zajišťuje třída `JavaMailSender`. Této třídě se nastavuje SMTP server[10], port na kterém běží server a e-mailová adresa ze které bude pošta zasílána. Pro tuto aplikaci byl zřízen účet u Googlu, jehož smtp server běží na portu 587. Kromě této konfigurace je zde nastavena také výchozí šablona zprávy, která obsahuje jen adresu odesílatele.

Konfigurační balíček je jádrem celého systému a je nezbytný pro běh celé aplikace.

5.5 Objevení chyby v ekonomickém systému Pohoda

Při implementaci propojení s ekonomickým systémem byla objevena chyba na straně programu Pohoda. Chyba se vyskytuje při zasílání špatného XML požadavku na import objednávek. Na webových stránkách tohoto systému je možno zobrazit, jak takový požadavek vypadá. Ten se ovšem neshoduje s požadavkem, který se zasílá při importu dat. Chyba je jen v jednom slově, a to, že místo klíčového slova „version“ je slovo „vesion“. Jedná se pravděpodobně jen o překlep vývojářů programu Pohody a oprava nebude nijak zvlášť náročná. Ovšem bez této opravy nebude informační systém správně reagovat na požadavky. V případě opravy ze strany Pohody je webový informační systém pro vinařskou firmu již připraven na propojení s ekonomickým systémem. Chyba je detailněji zobrazena na obrázku. *Obrázek 8*.

[10] SMTP server – server na který se zasílá elektronická pošta.

```
<?xml version="1.0" encoding="UTF-8"?>
<dat:dataPack id="00000001" ico="04416678" application="Obecný Internetový obchod -
import" version="2.0" note="Import objednávek"
              xmlns:dat="http://www.stormware.cz/schema/version_2/data.xsd"
              xmlns:ftr="http://www.stormware.cz/schema/version_2/filter.xsd"
             xmlns:lst="http://www.stormware.cz/schema/vesion_2/list.xsd"
              xmlns:typ="http://www.stormware.cz/schema/version_2/type.xsd">
    <dat:dataPackItem id="00000001" version="2.0">
        <lst:listOrderRequest version="2.0" orderType="receivedOrder"
orderVersion="2.0">
            <lst:requestOrder>
                <ftr:filter>
                    <ftr:dateFrom>2017-04-06</ftr:dateFrom>
                    <ftr:dateTill>2017-04-20</ftr:dateTill>
                </ftr:filter>
            </lst:requestOrder>
        </lst:listOrderRequest>
    </dat:dataPackItem>
</dat:dataPack>
```

Obrázek 8: Chyba v požadavku na import objednávek

6 Testování

Tato kapitola se zabývá průběžným testováním informačního systému a také testováním výsledné aplikace. Průběžné testování probíhalo pomocí debuggeru vývojového prostředí Idea[11]. Výsledná aplikace pak byla testována zákazníkem a potenciálním uživatelem systému.

6.1 Průběžné testování programátorem

Průběžné testování probíhalo při vývoji aplikace. Po implementaci nové vlastnosti systému byla tato vlastnost řádně otestována a v případě nalezené chyby opravena. K snadnějšímu nalezení chyby byl použit již zmíněný debugger.

K nasazení testovací verze byl nakonfigurován server Apache Tomcat ve verzi 7.0., který je více popsán v kapitole 3. Webové technologie. Tento server běžel pouze na lokální síti programátorova počítače. Stejně tak byla i vytvořena speciální databáze s testovacími daty. Tabulky této databáze byly vyplněny náhodnými, ale validními hodnotami.

Dále bylo nutné také otestovat bezpečnost při přihlašování uživatelů pod různými rolemi, kdy každá role má různá práva na jednotlivé stránky webu. Pro tuto akci byli vytvořeni v databázi dva uživatelé s různými rolemi, pomocí kterých následně probíhalo přihlašování do systému a provádění akcí v systému.

Toto testování bylo nezbytně nutné, aby byl zajištěn plynulý průběh vývoje aplikace a aby byla aplikace řádně připravena na testování zákazníkem.

6.2 Testování zákazníkem

Testování zákazníkem probíhalo ve dvou etapách. První z nich byla po dokončení administrační části a nákupního košíku a druhá po dokončení webového uživatelského rozhraní.

Po dokončení administrační části bylo připraveno i testovací uživatelské rozhraní, které sloužilo pouze pro vstupní data. Uživatel tak zkoušel zadávat různé hodnoty, případně v administračním režimu spravovat svůj systém. Nalezená chyba byla vždy uživatelem zapsána a následně předána na opravu

[11] Více o vývojovém prostředí IDEA na: https://www.jetbrains.com/idea/ .

programátorem. Například: *Pří zadání jedenáctimístného telefonního čísla do formuláře pro vyplnění objednávky. Objednávka se úspěšně zašle, očekávaný stav: Výpis: chybné telefonní číslo.*

Další etapou bylo testování uživatelského rozhraní. V této části se už nezkoumala práce s daty, ale celkový design a vizualita webu, zda jsou správně zvoleny fonty, barvy, rozmístění textů apod. Z tohoto testování vycházel také delší výstup a seznam chyb, jelikož koncoví uživatelé více vnímají tuto stránku webových aplikací, než tu stránku, kde se manipuluje s daty. Opět stejně jako v první etapě byl výstupem dokument se seznamem chyb. Například: *Špatně zvolená barva písma u registračního formuláře.*

Po opravení chyb z obou etap bylo možné aplikaci předat k finálnímu testování a případnému nasazení na produkci.

6.3 Finální testování

Finální testování probíhalo jak ze strany programátora, tak i ze strany zákazníka. Nejdříve byly opraveny chyby z výstupu testování zákazníkem a následně otestovány programátorem. Jakmile byla úspěšně dokončena oprava chyb s následným testováním, byl opět systém předán zákazníkovi k otestování. Z tohoto testování už byl výstup mnohem kratší a jednalo se pouze o drobnosti, které bylo možné snadno opravit.

Další součástí finálního testování, bylo vyzkoušení aplikace pod různými prohlížeči. K tomu byl využit nástroj BrowserStack[12], který umožňuje bezplatnou zkušební verzi. BrowserStack nabízí asi všechny běžně používané prohlížeče, skoro ve všech verzích. Tomuto nástroji pak byla předána adresa webové aplikace a zvoleny prohlížeče, pod kterýma testování probíhalo. Výstupem byly snímky obrazovky z jednotlivých prohlížečů. Zvolené prohlížeče: Google Chrome, Mozzila Firefox, Internet Explorer, Edge, Opera a Safari.

Finální testování proběhlo, aby se doladily poslední nedostatky systému a na produkci pak běžel systém co nejoptimálněji. Dále se také zamezilo, případným nedostatkům, kterých by si nikdy zákazník jakékoliv webové stránky, portálu nebo informačního systému neměl všimnout. Takový nedostatek by mohl zákazníka odradit při další návštěvě.

[12] Více o BrowserStacku na https://www.browserstack.com/start .

7 Návrh na rozšíření

Tato kapitola se zabývá návrhy na rozšíření aktuálního systému, které budou pravděpodobně časem doplněny. Těchto návrhů je několik a lze je rozčlenit i do více kategorií.

7.1 Rozšíření uživatelského prostředí

Jednou z kategorií je rozšíření části pro uživatele, kde se nabízí více možností, jak informační systém něčím obohatit. V současné době je trendem mít propojenou webovou stránku s facebookem. Pravděpodobně by mohlo jít o nějaký postranní banner, kde by se zobrazovaly příspěvky z facebookové stránky tohoto vinařství. Případně by mohlo jít o propojení i s jinými sociálními sítěmi, jako například Instagram, kdy by se jednotlivé fotografie přidané na Instagram současně uložily do databáze informačního systému a zobrazovaly se už v hotové fotogalerii.

Dalším návrhem je rozšíření uživatelského účtu a možnost spravovat svůj účet. To znamená, že při registraci a přihlášení uživatele, by byla speciální nabídka, kde by uživatel mohl měnit své osobní a přihlašovací údaje. Dále by se mu zde také zobrazovaly statistiky nákupů a stejně, jako administrátor by měl možnost si vygenerovat faktury.

Posledním návrhem na rozšíření uživatelského prostředí je přidání hodnocení k jednotlivým položkám. Jednalo by se například o hodnocení počtem hvězdiček. U každého produktu by se kromě ostatních informací zobrazoval také vstup pro hodnocení. Na hodnocení by měl pouze právo přihlášený uživatel, a který vykonal objednávku daného produktu. Toto rozšíření může sloužit, jako zpětná vazba prodejci a současně doporučení pro nové zákazníky, a který produkt si mají koupit. Místo hodnocení hvězdičkami by se pak mohla přidat slovní recenze produktu.

7.2 Rozšíření administrátorského prostředí

U administrace se nabízí rozšíření o skladové hospodářství, kdy by se jednalo o kontrolu množství produktů na skladu. V případě, že by se množství produktů snížilo na nulovou hodnotu, by byl produkt nastaven do režimu „není skladem“. Současná databáze je na tohle rozšíření připravena, ale jelikož internetový obchod není primární zdroj prodeje, bylo by tohle skladové hospodářství špatně kontrolovatelné.

V sekci objednávky je možné generovat faktury ke každé z objednávek zvlášť. Zde se nabízí další rozšíření, a to potisk s adresou na zásilku. Toto generování by bylo možné u objednávek, které budou předávány přepravní společnosti.

Systém by mohl být také rozšířen o zasílání slevových kupónů na základě definice administrátora. Například u častých zákazníků nebo u objednávky s velkým množství. Nemuselo by jít výhradně o slevové kupóny, ale i o jiné předměty například jeden z produktů zdarma a podobně. Toto rozšíření už spíše závisí na prodejci.

7.3 Ostatní rozšíření

Kromě předchozích rozšíření je možné přidat spoustu dalších. Jelikož jde o internetový obchod a je potřeba získat co nejvíce zákazníků, nabízí se vylepšit marketingovou stránku informačního systému. Ideálním nástrojem, který je v dnešní době u internetových obchodů velmi populární je Facebook Pixel[13]. Tento nástroj umožňuje sledovat pohyb zákazníka na stránkách internetového obchodu a za tímto účelem je pak možné cílit reklamu na sociální síti Facebook. Jedná se o JavaScriptový kód, který obsahuje různé sledovací akce, jako například: produkty které si zákazník prohlíží, přidání do košíku nebo dokončení registrace.

[13] Více o Facebook Pixel na https://www.facebook.com/business/help/952192354843755 .

8 Závěr

Cílem této bakalářské práce byl návrh a implementace webového informačního systému a internetového obchodu pro vinařskou společnost. Součástí práce bylo také propojení aplikace s ekonomickým systémem Pohoda.

Výsledkem je webová aplikace, jejíž logická část je napsána v jazyce Java s využitím frameworku Spring. Webové uživatelské rozhraní je dále implementováno pomocí JSP, CSS a Java Script. Databáze je pak tvořena v PostgreSQL.

Výsledná aplikace umožňuje prodej vinných produktů a automatizaci vyřizování objednávek. Běžný uživatel se může dozvědět základní informace o vinařství a nabízených produktech, případně nahlédnout do fotogalerie. Uživatel s právy administrátora má možnost spravovat celý systém. Může spravovat uživatele a své produkty, které nabízí. Dále má umožněno vytvářet aktuality nebo přidávat fotografie do galerie. Kromě toho má administrátor možnost správy objednávek s generováním faktur a zasílání informačních e-mailů. Všechny tyto objednávky může importovat do ekonomického systému Pohoda.

Součástí práce byla také analýza požadavků zákazníka. Předmětem této analýzy bylo setkání se zákazníkem a studium již existujících webových informačních systémů s podobným motivem. Z této analýzy vychází také návrh řešení, kterým se práce podrobněji zabývá. Návrh vznikl na základě nastudovaných znalostí a zjištění informací z analýzy. V návrhu se mi podařilo vytvořit datový model, diagram případů užití a také návrh webového uživatelského rozhraní.

U aplikace probíhalo také testování ve více etapách. První etapou bylo průběžné testování programátorem. Dále pak testování zákazníkem a nakonec finální testování. Hlavním cílem bylo zajistit, aby výsledkem práce byl funkční produkt a tímto testováním se tak zamezilo případným chybám.

Aplikaci by bylo možné dále rozšířit o skladové hospodářství. Dále je zvažováno zákazníkem aplikaci propojit se sociální sítí, které jsou v poslední době velmi oblíbené. Možností je také implementace slevových systémů. Tato rozšíření pak již závisí na zákazníkovi a není problém je v budoucnu doplnit.

Díky této práci jsem se naučil novým technologiím a podařilo se mi více proniknout do tvorby moderních webových aplikací. Přínosem byla také práce s frameworkem Java Spring a se serverem Apache Tomcat. Vyzkoušel jsem si, jak probíhá celý vývoj webového informačního systému včetně samotného testování.

Literatura

[1] Daryl Kulak, Eamonn Guiney: *Use cases: requirements in context*. Boston Addison-Wesley, 2000. ISBN 0201657678.

[2] Keng Siau: *Journal of database management: an official publication of the International Data*. Management Institute of the Information Resources Management Association. Harrisburg, 1992. ISSN 1063-8016.

[3] Zendulka J, Rudolfová I: *Studijní opora databázové systémy*. FIT VUT v Brně, 2006.

[4] Pohoda: *POHODA - účetní program, ekonomický program POHODA*. [online] https://www.ucetni-systemy.cz/?gclid=CJLwv4_n39ECFeUV0wodnE0F7A

[5] Stormware: *Import a export dat.* [online] https://www.stormware.cz/xml/

[6] Schildt Herbert: *Java 7: výukový kurz*. Brno: Computer Press, 2012. ISBN 978-80-251-3748-2.

[7] Oracle Technology Network: *Java EE – Documentation.* [online] http://www.oracle.com/technetwork/java/javaee/documentation/index.html, 2017.

[8] Spring Framework: *Introduction Spring Framework.* [online] https://projects.spring.io/spring-framework/#quick-start, 2017.

[9] Apache Tomcat: *Documentation index.* [online] http://tomcat.apache.org/tomcat-7.0-doc/index.html, Publikováno 28.3.2017.

[10] Tutorialspoint: *JSP Overview.* [online] https://www.tutorialspoint.com/jsp/jsp_overview.htm, 2017.

[11] PostgreSQL: *About.* [online] https://www.postgresql.org/about/, 2017.

[12] Free CSS templates: *License* [online]
http://www.freecsstemplates.org/license/, 2016.

[13] Bauer Christian, King Gavin: *Hibernate in action*. Greenwich, CT: Manning, 2005, ISBN 1-932394-15-x.

[14] Dušan Janovský: *Úvod do JavaScriptu*. [online]
https://www.jakpsatweb.cz/javascript/javascript-uvod.html

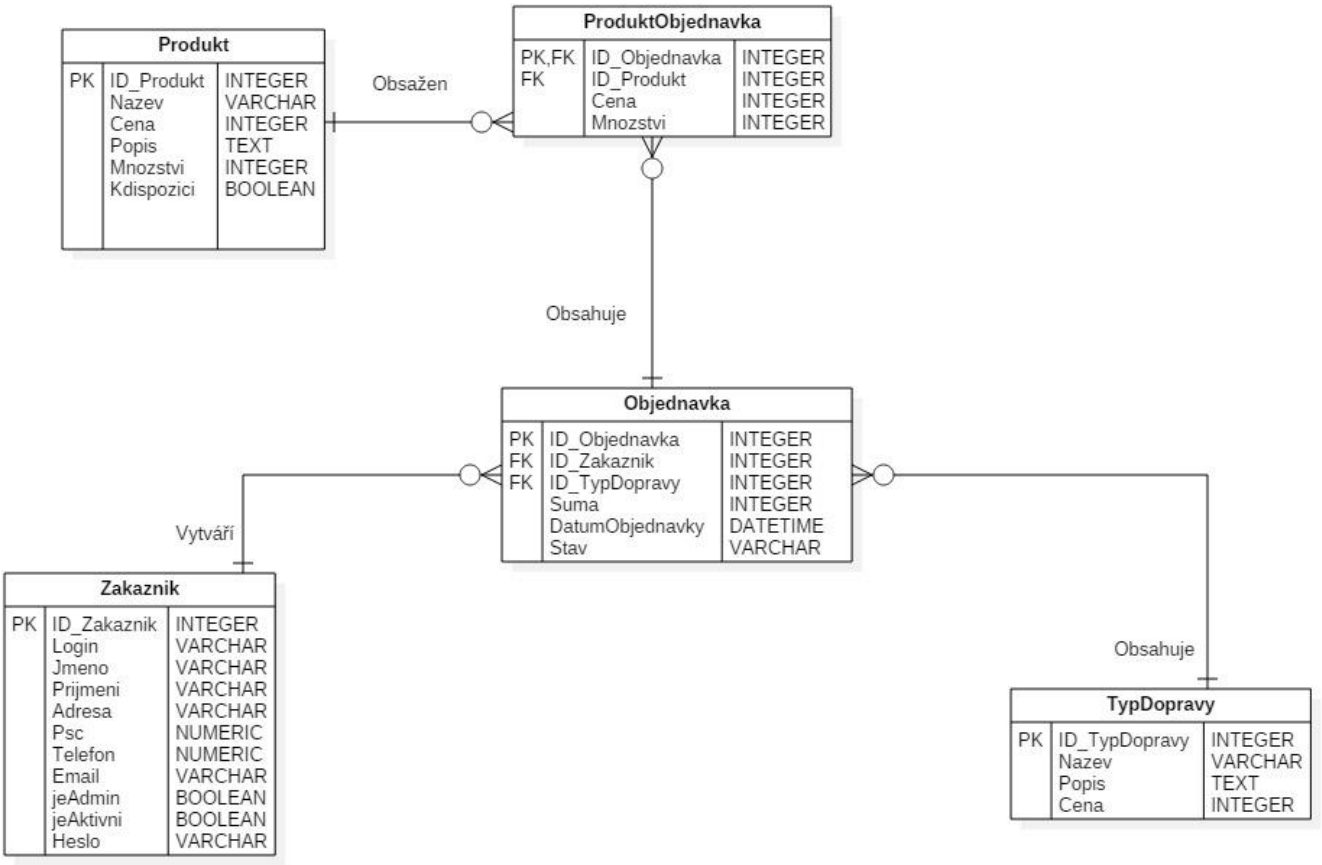
[15] David Čápka: *MVC architektura*. [online].
http://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor, 2014.

[16] Adaptic: *Co je Frontend.* [online]
http://www.adaptic.cz/znalosti/slovnicek/frontend/, 2016.

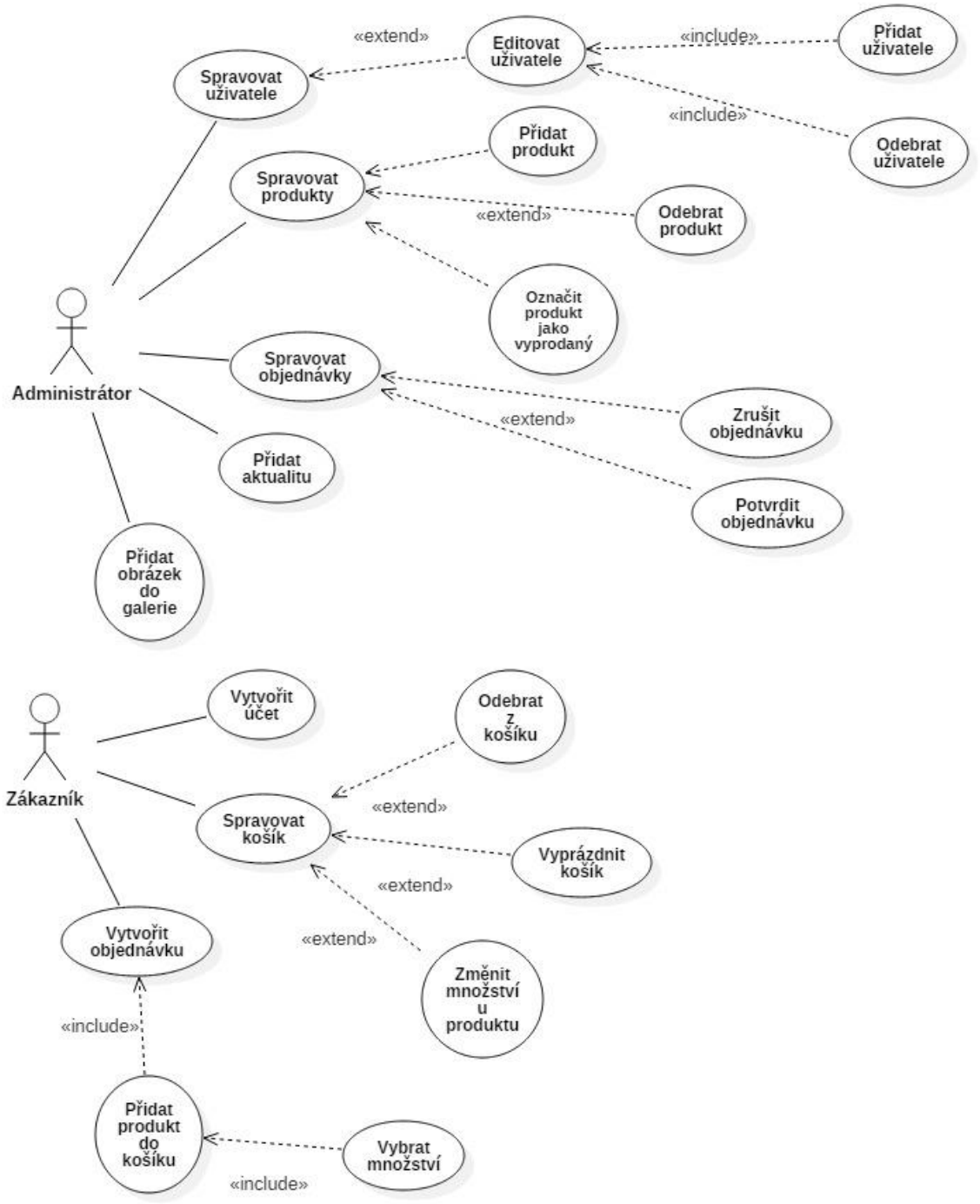
[17] Adaptic: *Co je Backend.* [online]
http://www.adaptic.cz/znalosti/slovnicek/backend/, 2016.

[18] Stormware: *Obecný internetový obchod.* [online]
https://www.stormware.cz/Podpora/FAQ/obecny_obchod.aspx, 2016.

Příloha A: ER diagram



Příloha B: Use case diagram



Příloha C: Obsah CD

/src/ – Složka obsahující zdrojové soubory

/doc/ – Složka obsahující tuto zprávu ve formátu .docx a .pdf

/conf/ - Konfigurace serveru a překlad aplikace

/sql/ - Skripty pro inicializaci databáze

/build/ - Výsledný .war archív

readme.txt – Soubor s manuálem pro instalaci a spuštění