EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2015/3711685051

VÝVOJ WEB APLIKÁCIE VO VYBRANOM PROSTREDÍ

Diplomová práca

2015 **Bc. Martin Trubíni**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

VÝVOJ WEB APLIKÁCIE VO VYBRANOM PROSTREDÍ

Diplomová práca

Študijný program: Manažérske rozhodovanie a informačné technológie

Študijný odbor: 6258 Kvantitatívne metódy v ekonómii

Školiace pracovisko: Katedra aplikovanej informatiky FHI

Vedúci záverečnej práce: Ing. Martin Blahušiak, PhD.

Bratislava 2015 **Bc. Martin Trubíni**

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Zadanie záverečnej práce

Meno a priezvisko študenta:	Bc. Martin Trubíni
Študijný program:	Manažérske rozhodovanie a informačné technológie (Jednoodborové štúdium, inžiniersky II. st., externá forma)
Študijný odbor:	3.3.24 Kvantitatívne metódy v ekonómii
Typ záverečnej práce:	Inžinierska záverečná práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Vývoj web aplikácie vo vybranom prostredí

Anotácia: Zameranie práce je na výber prostredia pre vývoj konkrétnej web aplikácie a na analýzu, návrh a implementáciu tejto web aplikácie vo vybranom prostredí.

Vedúci:	Ing. Martin Blahušiak, PhD.
Katedra:	KAI FHI - Katedra aplikovanej informatiky FHI
Vedúci katedry:	doc. Ing. Gabriela Kristová, CSc.
Dátum zadania:	15.11.2013
Dátum schválenia:	26.11.2014

Čestné vyhlásenie

Čestne vyhlasujem že záverečnú prácu som vypracoval samostatné a že som uviedol všetku použitú literatúru.

Dátum: 27.4.2015 .....

(podpis študenta)

Poďakovanie

Týmto by som rád poďakoval Ing. Martinovi Blahušiakovi, PhD. za odbornú pomoc a cenné rady, ktoré mi pomohli pri vypracovaní tejto diplomovej práce.

Abstrakt

TRUBÍNI, Martin: *Vývoj web aplikácie vo vybranom prostredí* - Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. - Vedúci záverečnej práce: Ing. Martin Blahušiak, PhD. - Bratislava: FHI EU, 2015, počet strán 58.

Cieľom záverečnej práce je analýza a návrh webovej aplikácie pre skladový systém firmy Starfit a jej implementácia s využitím objektovo orientovaného jazyka PHP 5. Práca je rozdelená do troch kapitol. Obsahuje 27 obrázkov, 4 tabuľky a 4 prílohy.

Prvá kapitola je venovaná charakteristike jazyka PHP 5, jeho princípom objektovo orientovaného programovania v tomto jazyku a relačnému databázovému systému MySQL. V záverečnej časti prvej kapitoly je popísaná firma, jej súčasný stav riešenia evidencie tovaru a tržieb a popis existujúcich programov, ktoré riešia danú problematiku.

V nasledujúcej kapitole sme vymedzili hlavný cieľ diplomovej práce, ako aj čiastkové ciele, ktorými sú analýza, návrh a vytvorenie webovej aplikácie. Súčasťou kapitoly je aj definovaná metodika práce a použité metódy skúmania.

Záverečná kapitola sa zaoberá popisom výsledkov riešenia, analýze a návrhu aplikácie a na koniec samotnej implementácie aplikácie do webového prostredia firmy.

Výsledkom riešenia danej problematiky je vytvorenie databázy a webovej aplikácie využitím objektovo orientovaného programovania v jazyku PHP 5.

Kľúčové slova: webová aplikácia, objektovo orientované programovanie, databáza, PHP, MySQL

Abstract

TRUBÍNI, Martin: *Web application development in choosen environment* - The University of Economics in Bratislava, The Faculty of Business Informatics; The Department of Applied Informatics - Thesis Supervisor: Ing. Martin Blahušiak, PhD. - Bratislava: FHI EU, 2015, The number of pages: 58.

The aim of this thesis is to analysis and design a web application for a warehousing system and its implementation using a PHP 5 object-oriented language. The work is divided into three chapters. It contains 27 pictures, 4 tables and 4 attachments.

The first chapter is devoted to the characterisation of the PHP5 language, its history and principles of its object-oriented programming and the relational database management system MySQL. The final section of the first chapter describes the company Starfit, its current solution of evidence of goods and sales and description of existing programs which solve this issue.

The following chapter defines the main aim of this thesis, as well as its partial goals, which are: analysis, design and development of the web application. This chapter describes methodology of work and survey methods used.

The final chapter deals with the description of the results, analysis, the design of the application and at the end - the actual implementation of the application into the web environment of the company.

The result of solving the issue is the creation of the database and the web application using object-oriented programming in PHP 5.

Keywords: web application, object oriented programming, database, PHP, MySQL

Obsah

Úvod

V súčasnom rýchlom a konkurenčnom podnikateľskom prostredí je pre firmy nevyhnutné efektívne využívať informačné technológie. Vďaka Internetu sa služby, ktoré informačné systémy poskytujú, stávajú čoraz dostupnejšie. Požiadavky firmy Starfit na návrh a tvorbu webovej aplikácie pre potreby ich skladového hospodárstva, sú vhodnou témou pre túto diplomovú prácu, pretože ukazujú využitie webovej aplikácie v prostredí ekonomického subjektu.

Hlavným cieľom práce je navrhnúť a vytvoriť webovú aplikáciu skladového hospodárstva, na základe špecifických požiadaviek firmy Starfit. Našou úlohou bolo zistiť potreby tejto firmy, ktorá prevádzkuje fitness a aerobic centrum a poskytuje služby v tejto oblasti, spracovanie týchto potrieb a navrhnutie webovej aplikácie, ktorá bude slúžiť na evidenciu tovaru na sklade a každodenné pohyby tohto tovaru v predajni. Ďalej sme sa venovali praktickému riešeniu webovej aplikácie, t.j. vytvorenie databázy a webovej aplikácie s použitím objektovo orientovaných prvkov a implementácii tejto webovej aplikácie do existujúcej webovej stránky firmy.

V prvej kapitole si predstavíme jednotlivé technológie. Najprv jazyk PHP, jeho vývoj a aktuálny stav. Analyzujeme výhody a nevýhody tohto jazyka. Následne sa venujeme objektovo orientovanému programovaniu, zadefinujeme si základne pojmy a popíšeme princípy objektovo orientovaného programovania. Venujeme sa implementácii objektovo orientovaného programovania v jazyku PHP 5 a práci s objektmi v tomto jazyku. Ďalej sa venujeme databázovému systému. Popíšeme si zvolený databázový systém MySQL, jeho výhody a nevýhody. V prvej kapitole si ešte predstavíme skladový systém pre firmu Starfit, ktorý sme sa rozhodli vytvoriť. Zanalyzujeme si súčasný stav, zameranie aplikácie a možnosti jej využitia a rozvoja. Takisto si zanalyzujeme existujúce produkty, ktoré riešia danú problematiku. V druhej kapitole si charakterizujeme predmet riešenia a popíšeme si hlavný cieľ našej práce, ako aj jednotlivé čiastkové ciele, ktoré nám pomôžu pri naplnení hlavného cieľa. Venujeme sa metodike práce a metódam skúmania, ktoré sú pri písaní práce využité. V tretej časti sú výsledky práce a diskusia, ktoré sú najvýznamnejšou časťou záverečnej práce. Najprv sa venujeme analýze potrieb firmy, pre ktorú tvoríme skladový systém ako webovú aplikáciu. Potom navrhneme objektový model, ktorý bude pristupovať k údajom databázy. Objektový model bude slúžiť na pripojenie k

databáze, na zadávanie, výber, úpravu a mazanie údajov v databáze. Ďalej si navrhneme databázový model, ktorý zohľadňuje všetky požiadavky skladového systému firmy a vytvoríme databázu v nami zvolenom databázovom systéme MySQL. V poslednej časti tretej kapitoly sa venujeme samotnej tvorbe webovej aplikácie a implementácii do existujúceho webového prostredia firmy. Táto časť ukazuje čo aplikácia dokáže a popisuje jej jednotlivé časti.

1 Súčasný stav riešenej problematiky

V tejto práci navrhneme konkrétnu webovú aplikáciu a jej implementáciu s využitím objektovo orientovaného programovania. Princípy objektovo orientovaného programovania boli rozpracované už v šesťdesiatych rokoch 20. storočia na americkej univerzite MIT, kde boli prvýkrát uvedené pojmy ako "objekty" a "inštancie". V rokoch 1962 až 1967 dvaja programátori Kristen Nygaard a Ole-Johan Dahl z Nórskeho výpočtového strediska v Osle navrhli prvý objektovo orientovaný programovací jazyk SIMULA[1]. Aj keď sa tento jazyk nikdy nestal široko používaný, významne ovplyvnil moderné programovacie metodológie. Jazyk SIMULA priniesol dôležité koncepty objektovo orientovaného programovania akými sú objekty, triedy a dedičnosť.[2] Na rozdiel od štruktúrovaného programovania, ktoré pozostáva z dát uložených v premenných a procedúr, ktoré s týmito dátami pracujú. V štruktúrovanom programovaní sa používajú vybrané riadiace štruktúry ako sú podmienky, sekvencie a cykly. Výhody objektovo orientovaného programovania sa prejavia hlavne pri vývoji rozsiahlych projektov. Tieto výhody spočívajú najmä v prehľadnosti kódu, lepšej správe a údržbe a tiež v ľahšej rozšíriteľnosti. Objektovo orientované programovanie má jasnú štruktúru, rozhranie a premyslený koncept. Optimálne môže byť aj použiť obidva prístupy, štruktúrované aj objektovo orientované, súčasne.

Webová aplikácia v softvérovom inžinierstve je aplikácia poskytovaná používateľom z webového serveru cez počítačovú sieť Internet, alebo jej vnútropodnikovú obdobu intranet. Webové aplikácie sú používateľom prístupné cez webový prehliadač, ktorý sa nazýva tenký klient. Webové aplikácie dynamicky generujú sériu webových

[1] Wikipedia, the free encyclopedia, 2015, *Object-oriented programming*, [online], [2015-03-25] Dostupné na internete: https://en.wikipedia.org/wiki/Object-oriented_programming#History

[2] HOLMEVIK, J. R. 1994. *Compiling SIMULA: A Historical Study of Technological Genesis. IEEE Annals of the History of Computing, Vol. 16, No. 4, p. 25-37*. Dostupné na Internete: http://www.idi.ntnu.no/grupper/su/publ/simula/holmevik-simula-ieeeannals94.pdf

stránok v štandardnom formáte HTML. Pre pridanie dynamických prvkov do používateľského rozhrania sa používa skriptovací jazyk JavaScript. Webový prehliadač počas behu aplikácie interpretuje a zobrazuje stránky. Podstatnou výhodou vývoja webových aplikácii je ich schopnosť pracovať podľa určenia bez ohľadu na operačný systém či jeho verziu inštalovanú na danom klientskom počítači.

Webové aplikácie sú obvykle štruktúrované ako trojvrstvové. V tej najbežnejšej forme je webový prehliadač prvou vrstvou (prezentačnou), nástroje pre generovanie stránok (napr. PHP, ASP, Java servlety, Perl) sú vrstvou strednou (logickou) a databáza je vrstvou treťou (dátovou). Webový prehliadač posiela požiadavky strednej vrstve, ktorá ich obsluhuje prostredníctvom dotazov na databázu a generovaním používateľského rozhrania.

1.1 Jazyk PHP

PHP (Hypertext Preprocessor) je skriptovací programovací jazyk. Je vydaný pod licenciou PHP License ako slobodný softvér. Tento jazyk je určený predovšetkým na programovanie aplikácií na strane servera (server-side) a pre vývoj dynamických webových stránok. Server-side programovací jazyk PHP používa až 82 % webových stránok.[3] Je to veľký náskok pred druhým najpoužívanejším server-side programovacím jazykom pre webové stránky ASP.NET (17 %). Na tretej pozícii s 2,8 % je jazyk Java. PHP je nezávislý na operačnom systéme a platforme. Obľúbená je kombinácia s operačným systémom Linux, webovým serverom Apache a databázovým systémom MySQL. Pre túto kombináciu sa zaužívala skratka LAMP, teda Linux, Apache, MySQL a PHP. Syntax jazyka je inšpirovaná viacerými programovacími jazykmi podporujúcimi štruktúrované programovanie. Z jazykov Perl a C prebral jazyk PHP najviac vlastností. V neskorších verziách bolo PHP rozšírené o možnosť využívať objekty. V jazyku PHP sú naprogramované populárne internetové aplikácie ako WordPress, PrestaShop, MediaWiki, Drupal, Joomla!, phpMyAdmin, RoundCube a mnohé ďalšie.

Prvú verziu jazyka PHP napísal grónsky programátor Rasmus Lerdorf v roku 1994 v programovacom jazyku Perl, ako binárnu časť Common Gateway Interface (CGI). Túto sadu skriptov pomenoval "Personal Home Page Tools", známejšiu však ako "PHP Tools".[4] Dňa 13. júla 2004 bolo vydané PHP 5, ktoré obsahovalo nové jadro Zend Engine II. Bola

[3] W3Techs, *Usage of server-side programming languages for websites*, [online], [2015-03-25] Dostupné na internete: http://w3techs.com/technologies/overview/programming_language/all

[4] PHP.net, *History of PHP*, [online], [2015-03-25] Dostupné na internete: http://www.php.net/manual/en/history.php.php

vylepšená podpora pre objektovo orientované programovanie, PHP Data Objects extension (rozhranie pre napojenie k databázam), zvýšený výkon a ďalšie. Zároveň bolo opravených množstvo chýb a vylepšená bezpečnosť. Verzia PHP 5.4, ktorá bola vydaná 1.3.2012 priniesla hlavne zmeny v odobratí register_global, magic quotes a safe mode. Dôvodom boli bezpečnostné nedostatky. Súčasná verzia má označenie PHP 5.6 a prináša viaceré menšie vylepšenia. Vo verzii PHP 6 by mala pribudnúť hlavne nativná podpora Unicode a vnútorná reprezentácia reťazcov ako UTF-16.[5] Avšak niekedy v roku 2010 narazil na problém implementácie Unicode. Mnoho podstatných zmien priniesli už verzie 5.3 a 5.4. V marci roku 2010 bol projekt PHP 6, ako unicode experiment, ukončený. V roku 2014 sa skupina The PHP Group, ktorá sa v súčasnosti stará o vývoj jazyka PHP rozhodla, že nasledujúcou verziou bude PHP 7. Táto nová verzia by mala priniesť optimalizáciu výkonu vďaka refaktoringu jadra Zend Engine, ktorý bol vykonaný v rámci vývoja experimentálnej vetvy PHP, pôvodne pomenovanej ako phpng (PHP next generation). Tento prerobený Zend Engine ponesie názov Zend Engine 3. Nové PHP 7 ďalej prinesie zjednotenie syntaxe premenných, deklaráciu návratových hodnôt, deklaráciu skalárnych typov (integer, float, string and boolean). Vydanie PHP 7 sa očakáva v októbri roku 2015.

1.1.1 Charakteristika PHP

Programovací jazyk PHP bol vyvinutý na to, aby obohatil statické HTML stránky o dynamické skriptovanie. Pokým PHP neexistovalo, používali sa na tieto účely programy, ktoré boli volané cez rozhranie Common Gateway Interface (CGI). Na to aby sa mohol akýkoľvek CGI program spustiť, musel webový server zakaždým spustiť nový proces, čo bola značná záťaž pre operačný systém. V prípade PHP je to inak. Funkcie PHP sa načítajú ako modulová súčasť webového servera a sú teda spustené priamo v kontexte procesu webového servera. Zrýchlenie vybavovania používateľských požiadaviek je zreteľné najmä pri veľmi frekventovaných stránkach.[6]

Spracovanie PHP skriptov prebieha na strane servera. Interpreter jazyka PHP spracováva každý súbor s príponou .php a vykoná v ňom zadané inštrukcie. Výsledok je vrátený webovému serveru, ktorý odošle tento obsah klientovi. Klientom je zvyčajne webový prehliadač. Medzi najdôležitejšie inštrukcie patria značky (tagy) pre začiatok

[5] Wikipedia, the free encyclopedia, 2015, *PHP*, [online], [2015-03-25] Dostupné na internete: http://en.wikipedia.org/wiki/PHP

[6] KOFLER, M. - ÖGGL, B. 2007. *PHP 5 a MySQL 5 : Průvodve webového programátora*. Prvé vydanie. Brno : Computer Press, a.s., 2007. 608 s.
ISBN 978-80-251-1813-9

`<?php` a koniec `?>` PHP kódu. Na tvorbu PHP skriptu stačí, podobne ako pri HTML, aj obyčajný textový editor. Vhodnejšie je však použiť textový editor, ktorý zvýrazňuje syntax jazyka a tým zvyšuje prehľadnosť a orientáciu v zdrojovom kóde.

Jednou z výhod jazyka PHP je, že nie je nutné dopredu definovať dátový typ premennej. Názov premennej sa začína symbolom doláru `$`, za ktorým nasledujú znaky ASCII, číslice alebo znak podčiarkovníka (_). Názov premennej nesmie začínať číslicou. Jazyk PHP podľa obsahu premennej sám rozpozná o aký dátový typ sa jedná. Jazyk PHP je case sensitive, čo znamená, že v názvoch premenných sa rozlišujú malé a veľké písmena. V PHP existujú tri druhy komentárov. Znakmi `/* */` sa označuje blokový komentár a znakmi `//` alebo `#` sa označujú jednoriadkové komentáre. Riadiace štruktúry ako podmienený príkaz `if`, `switch` alebo cykly `while`, `for` a `foreach` majú podobnú syntax a návratové hodnoty ako v jazykoch C, C++, Java alebo Perl. Funkcie sa definujú príkazom `function`, po ktorom nasleduje názov a parametre funkcie oddelené čiarkou.

Výhody jazyka PHP

Medzi najdôležitejšie výhody jazyka PHP patrí:

- nezávislosť na operačnom systéme a platforme,
- jednoduchá syntax,
- rozsiahly súbor funkcií priamo v základnej knižnici PHP a ďalšie v repozitároch PEAR a PECL,
- vstavaná podpora rôznych databázových systémov,
- podpora objektovo orientovaného programovania,
- široká podpora na webhostingových systémoch,
- slobodná licencia free software,
- dobrá dokumentácia.

Nevýhody jazyka PHP

Jazyk PHP má aj svoje nevýhody, akými sú napríklad:

- nekonzistentné pomenovanie niektorých funkcií. Napr. `strpos()` a `str_replace()`,
- nejednotné poradie parametrov,
- slabá podpora Unicode,

- rozdiely medzi verziami a problémy so spätnou kompatibilitou,
- po spracovaní požiadavku sa stráca kontext aplikácie, vytvára sa vždy nanovo, čo oslabuje výkon.

1.1.2 Objektovo orientované programovanie v PHP 5

Objektovo orientované programovanie (skratka OOP z anglického Object-oriented programming) je metóda vývoja softvéru založená na používaní dátových štruktúr zvaných objekty. OOP sa niekedy označuje aj ako programátorské paradigma, pretože popisuje nie len spôsob vývoja a zápisu programu, ale aj spôsob akým návrhár programu o probléme premýšľa.

Základným pojmom v OOP je objekt (object). Objekt je súbor dát a funkcií. Abstrakciou objektu, ktorá na všeobecnej úrovni podchycuje v architektúre programu podstatu všetkých objektov podobného typu, je trieda (class). Trieda je predpis, ktorý určuje ako vytvoriť objekt daného typu. Objekt je potom inštancia triedy. Je to konkrétny prípad realizácie predpisu. Objekt má svoj stav v podobe vlastností (properties) a cez operácie zvané metódy (methods) poskytuje rozhranie ako s ním pracovať.[7]

Novú triedu vytvoríme prostredníctvom kľúčového slova `class`, za ktorým nasleduje názov triedy. Zdrojový kód triedy je uzatvorený v zložených zátvorkách. Vo vnútri triedy potom môžeme definovať premenné, vlastnosti, metódy aj konštanty. Triedy môžu existovať buď samostatne, alebo môžu rozširovať ďalšie triedy. Podtrieda ktorú vytvoríme, zdedí všetky metódy nadradenej triedy. Túto vlastnosť nazývame dedičnosť (inheritance) a jedná sa o jeden z hlavných pilierov objektovo orientovaného programovania. Na rozšírenie existujúcej triedy použijeme kľúčové slovo `extends` hneď za menom triedy.

Metóda, ktorá je automaticky volaná pri vytvorení triedy, sa nazýva konštruktor. Od verzie PHP 5 má funkcia konštruktora jednotné meno a to `__construct`. Parametre, ktoré sú triede pri jej vytvorení odovzdané, sú potom k dispozícií vo vnútri konštruktora. V odvodených triedach sa konštruktor základnej triedy nevolá automaticky. V odvodených triedach ho je nutné zavolať pomocou konštrukcie `parent`, tak ako je to znázornené v Príklad 1: **Vytvorenie triedy a odvodenej triedy**.

[7] JACOBSEN, I. - CHRISTERSON, M. - JONSSON, P. - OVERGAARD, G. 1992. *Object Oriented Software Engineering : A Use Case Driven Approach*. Prvé vydanie. Addison-Wesley ACM Press, 1992. ISBN: 0-201-54435-0

Príklad 1: Vytvorenie triedy a odvodenej triedy

```
class BaseClass {
      function __construct() {
            echo "In BaseClass constructor\n";
      }
}
class SubClass extends BaseClass {
      function __construct() {
            parent::__construct();
            echo "In SubClass constructor\n";
      }
}
```

Podobne ako sa pri vytváraní triedy volá konštruktor, tak sa pri odstraňovaní triedy volá deštruktor. Názov tejto funkcie je `__destruct`. PHP robí správu pamäte automaticky a doba behu skriptu je veľmi krátka. Takže nie je nutné odstraňovať objekty ručne a deštruktor sa používa len zriedka.

Nový objekt, čiže novú inštanciu triedy, vytvoríme pomocou operátora `new`. Príkaz v Príklad 2: **Vytvorenie inštancie triedy** vytvorí nový objekt triedy BaseClass resp. SubClass a priradí ho premennej `$object`. Cez túto premennú potom môžeme pristupovať ku všetkým metódam a atribútom triedy.

Príklad 2: Vytvorenie inštancie triedy

```
// In BaseClass constructor
$object = new BaseClass();

// In BaseClass constructor
// In SubClass constructor
$object = new SubClass();
```

Ďalším z dôležitých pilierov OOP je zapúzdrenie (encapsulation). Zapúzdrenie znamená, že pri vytváraní zdrojového kódu novej triedy môžeme rozhodnúť, kto bude mať prístup k dátam a funkciám tejto triedy. Zend Engine II v PHP 5 používa pre prístup k premenným a metódam tri modifikátory prístupu: `public` (ľubovoľný prístup), `protected` (prvky triedy sú dostupné iba z triedy samotnej, alebo z nej odvodených tried) a `private` (pristupovať k prvkom môže iba trieda, ktorá ich vytvorila). Ak chceme vo

vnútri jednej metódy pristupovať k metóde druhej alebo členskej premennej triedy, použijeme odkaz na aktuálny objekt pomocou premennej `$this`.

Označením vlastnosti alebo metódy triedy kľúčovým slovom `static` ich spravíme prístupné aj mimo kontext daného objektu. Vzhľadom na to, že k statickým metódam pristupujeme bez toho, aby bola vytvorená inštancia triedy, pseudo-premenná `$this` ani šípkový operátor `->` používaný pri objektoch, nie sú k dispozícii. Na prístup k aktuálnemu objektu sa používa popisovač `self` a operátor `::` sa používa namiesto šípky, tak ako je to znázornené v Príklad 3: **Statická trieda**.

Príklad 3: Statická trieda

```
class MyStatic {
      public static $version = "PHP 5";
      public static function printVersion() {
            echo self::$version;
      }
}

MyStatic::printVersion(); // PHP 5
```

Kľúčovým slovom `final` sa v PHP 5 označujú triedy, ktoré nesmú byť rozšírené alebo metódy, ktoré nesmú byť prepísané žiadnou z podtried. Pokus o rozšírenie triedy alebo o prepísanie metódy označenej ako `final`, vedie ku kritickej chybe.

Interpretácia preťažovania premenných a metód je v PHP iná ako vo väčšine objektovo orientovaných jazykoch. Preťažovanie tradične znamená schopnosť mať viacero metód s rovnakým názvom, ale rozdielnym počtom a typom parametrov. Toto sa nazýva polymorfizmus. Preťažovanie (overloading) v PHP sa využíva, keď potrebujeme pristúpiť na nejakú ešte nedefinovanú premennú. Na to máme k dispozícii dve metódy: pre priradenie je to `__set` pre dotazovanie je to `__get`. Pre volanie nedefinovaných metód sa používa metóda `__call`.

PHP 5 prináša aj abstraktné triedy a metódy. Pri abstrakcii sa zameriavame na kľúčové vlastnosti objektu, ktoré sa snažíme abstrahovať pre viacero typov objektov. Triedy definované ako abstraktné nemôžu vytvárať inštancie. Vytvárať sa dajú len inštancie zdedených tried. Abstraktné triedy a metódy sa definujú kľúčovým slovom `abstract`.

Rozhranie objektov umožňuje vytvoriť kód, ktorý určí ktoré metódy musia obsahovať všetky triedy používajúce dané rozhranie. Rozhranie sa definuje kľúčovým slovom `interface`, podobným spôsobom ako trieda, ale s tým rozdielom, že sa tu nachádzajú len prázdne metódy označené ako `public`. V Príklad 4: **Rozhranie objektov** je znázornené použitie rozhrania objektov.

Príklad 4: Rozhranie objektov

```
// Deklarácia rozhrania 'iObject'
interface iObject {
      public function setX($x);
}

// Implementácia rozhrania
class Object implements iObject {
      public $length = 100;
      function setX($x) {
            $this->length = $x;
      }
}
```

Na ošetrenie chýb v triedach sa v objektovo orientovanom programovaní používajú výnimky, ktoré sú znázornené v Príklad 5: **Ošetrenie chýb**. Model výnimiek v PHP 5 je podobný iným programovacím jazykom.

Príklad 5: Ošetrenie chýb

```
class Division {
      public function divide($x) {
            if (!$x) {
                  throw new Exception('Division by zero.');
            }
            else return 1/$x;
      }
}

$number = new Division();

try {
```

```
      $number->divide(0);
} catch (Exception $e) {
      echo 'Caught exception: ',  $e->getMessage(), "\n";
}

// Continue execution
echo 'Hello World';
// Caught exception: Division by zero.
// Hello World
```

Jazyk PHP 5 umožňuje vďaka funkcii `__autoload` v prípade potreby načítať potrebné triedy. Táto funkcia sa volá automaticky vždy, keď sa pokúsime použiť nejakú funkciu, ktorej kód sa ešte nenačítal. Je zvykom každú triedu ukladať do samostatného súboru. Nasledujúci Príklad 6: **Funkcia __autoload** sa pokúsi načítať triedu MyClass zo súboru myclass.php.

Príklad 6: Funkcia __autoload

```
function __autoload($class_name) {
      require strtolower($class_name . '.php');
}
$obj = new MyClass();
```

Ak chceme naprogramovať cyklus cez všetky členské premenné daného objektu, môžeme to v PHP 5 urobiť pomocou konštrukcie `foreach`. Cyklus mimo triedy vidí len členy označené ako `public`. V Príklad 7: **Cyklus cez členské premenné objektu** je znázornený cyklus cez všetky členské premenné objektu.

Príklad 7: Cyklus cez členské premenné objektu

```
class MyClass
{
      public $public = 'public var';
      protected $protected = 'protected var';
      private $private = 'private var';

      function iterateVisible() {
            foreach($this as $key => $value) {
                  echo "$key => $value\n";
            }
```

```
        }
}
$object = new MyClass();

// public => public var
// protected => protected var
// private => private var
$object->iterateVisible();

// public => public var
foreach($object as $key => $value) {
        echo "$key => $value\n";
}
```

V klonovaní objektov je jeden z veľkých rozdielov medzi PHP verziami 4 a 5. Keď spravíme priradenie objektov v PHP 4, vytvorí sa kópia objektu. Obsah premenných sa potom nachádza na rôznych miestach operačnej pamäte. Zatiaľ čo v PHP 5 sa priradí iba odkaz (ukazovateľ). Obe objektové premenné odkazujú na jeden a ten istý objekt. Pokiaľ chceme vytvoriť kópiu objektu v PHP 5, využijeme techniku zvanú klonovanie. Ak je v triede, ktorú klonujeme, k dispozícii aj metóda `__clone`, tak sa príkazy tejto metódy po klonovaní použijú aj na nový objekt. Klonovaný objekt potom už nijako neovplyvní pôvodnú triedu.[8]

1.2 Alternatívy k jazyku PHP

Vzhľadom na to, že navrhovaný skladový systém má byť integrovaný do existujúcej webovej stránky firmy, ktorá ja naprogramovaná v jazyku PHP, je práve jazyk PHP optimálnou voľbou. Pre skladový systém bude využitý existujúci server Apache s podporou skriptovacieho jazyka PHP a relačnej databázy MySQL. Využitím existujúceho webového servera aj na účely skladového systému nevzniknú firme Starfit dodatočné náklady na prevádzku systému. Jazyk PHP poskytuje dostatočné možnosti na vytvorenie skladového systému a uspokojenie požiadaviek firmy. Popri jazyku PHP, ktorý je najrozšírenejším server-side programovacím jazykom, existujú aj ďalšie technológie na programovanie webových aplikácií na strane servera. Medzi najrozšírenejšie patria ASP.NET a JSP. Tieto technológie majú takisto svoje výhody i nevýhody.

[8] Živě.cz, 2005, *Objekty v PHP 5 - 5. díl*, [online], [2015-03-25] Dostupné na internete: http://www.zive.cz/clanky/objekty-v-php--5---5-dil/sc-3-a-123376/default.aspx

1.2.1 ASP.NET

ASP (Active Server Pages) je skriptovacia platforma od spoločnosti Microsoft, ktorá je primárne určená na dynamické spracovanie webových stránok na strane servera. ASP je dostupný ako modul k produktu Internet Information Services (IIS). IIS je softvérový webový server s kolekciou rozširujúcich modulov, vytvorený spoločnosťou Microsoft pre operačný systém Windows. Jedná sa o druhý najpoužívanejší webový server po servery Apache. ASP je technológia nezávislá na programovacom jazyku. Programovacie jazyky, ktoré ASP používa sú VBScript a JScript.

Nástupcom ASP je ASP.NET, ktorý je súčasťou .NET Fameworku pre tvorbu webových aplikácií. ASP.NET sa od svojho predchodcu ASP značne líši. ASP.NET je založený na CLR (Common Language Runtime). Programátori tak môžu realizovať svoje projekty v akomkoľvek jazyku podporujúcom CLR, ako sú napr. Visual Basic .NET, JScript .NET, C# a ďalšie. Aplikácie založené na ASP.NET sú rýchlejšie, pretože sú predkompilované do jedného či niekoľko málo DLL súborov, na rozdiel od čisto skriptovacích jazykov, kde sú stránky pri každom prístupe znovu a znovu parsované. ASP.NET podporuje tri rozdielne programovacie modely: Web Pages, MVC (Model View Controller) a Web Forms.[9]

ASP.NET Web Pages je najjednoduchší ASP.NET model, ktorý poskytuje jednoduchú možnosť kombinácie HTML, CSS, JavaScriptu a serverového kódu. Je podobný jazyku PHP a klasickému ASP.

Druhý ASP.NET model využíva návrhový vzor MVC. Ten rozdeľuje webovú aplikáciu do troch rozdielnych komponentov. Model, ktorý reprezentuje jadro aplikácie (napríklad zoznam databázových vstupov). View zobrazuje dáta (záznamy v databáze). A Controller, ktorý spracováva interakciu s používateľom. Typicky Controller číta dáta z View, kontroluje používateľské vstupy a posiela vstupné dáta do Modelu. Model MVC taktiež poskytuje plnú kontrolu nad HTML, CSS a JavaScriptom. Programovací model MVC je ľahšou alternatívou k tradičnému ASP.NET (Web Forms).

ASP.NET Web Forms je tradičný model ASP.NET. Je založený na udalosťami riadených webových stránkach napísaných ako kombinácia HTML, serverových

[9] ASP.net, *Get Started with ASP.NET Web Sites*, [online], [2015-04-06] Dostupné na internete: http://www.asp.net/get-started/websites

ovládacích prvkov a kódu servera. Web Forms sú kompilované (zostavené) a vykonané na serveri, ktorý generuje HTML webové stránky. ASP.NET Web Forms ponúka stovky rôznych webových ovládacích prvkov a webových komponentov na vytvorenie používateľom riadených webových stránok s prístupom k údajom.

Výhody ASP.NET

Medzi hlavné výhody technológie ASP.NET patria:

- rýchlejší beh aplikácie vďaka kompilovanému kódu a viac chýb zachytených už pri vývoji,
- používateľsky definované ovládacie prvky je možné použiť ako šablóny, čím sa významne redukuje duplicitný kód,
- bohatý výber ovládacích prvkov a knižníc zrýchľuje vývoj aplikácií,
- programátori majú na výber veľké množstvo programovacích jazykov,
- schopnosť cachovať celú stránku alebo len jej časť podstatne zvyšuje výkon serveru

Nevýhody ASP.NET

Nevýhodami technológie ASP.NET sú najmä:

- viazanosť na platformu Windows + IIS, ktorá podlieha licenčným poplatkom,
- problémy so spätnou kompatibilitou pri rôznych verziách .NET Framework,
- väčšia náročnosť pri dizajne aplikácie,
- skryté polia pri Viewstate je možné použiť len pre relatívne malé dáta.

1.2.3 Java

JavaServer Pages (JSP) je technológia pre vývoj dynamických webových stránok založená na jazyku Java, vyvinutá spoločnosťou Sun (dnes vlastnená firmou Oracle). JSP súbor je zmesou kódu Javy, XML, JDB, HTML a JavaScriptu. Stránka JSP sa skladá z viacerých častí: skriptovacie elementy, direktívy, inštrukcie a elementy XML. Skriptovacie elementy JPS sú ohraničené znakmi <% a %>. Technológia JSP používa tri druhy skriptovacích elementov a to deklarácia, výrazy a skriplety.[10]

[10] BURD, B. - KISZKA, B. 2003. *JSP : JavaServer Pages : Podrobný průvodve*. Prvé vydanie. Praha : Computer Press, a.s., 2003. 381 s. ISBN 80-7226-804-X

JavaServer Pages sú textové súbory s príponou .jsp, umiestnené na serveri (najčastejšie Apache Tomcat), kde beží servletový kontajner JSP. V okamžiku požiadavky zo strany klienta napr. na dokument index.jsp, kontajner JSP prevedie dokument na servlet v jazyku Java. Výsledný preložený dokument bude mať názov _index.java. Nasleduje preklad _index.java do bytového kódu Java triedy _index.class. Kontajner JSP načíta súbor triedy a na základe zistenej definície vytvorí nový objekt. Dochádza k inicializácii všetkých premenných. Tento krok môžeme pomenovať ako inicializácia samotného dokumentu JSP. Potom kontajner vytvorí zodpovedajúci dokument a odošle ho k používateľovi. Tento dokument neobsahuje žiadny zostatkový kód JSP. Pri ďalších požiadavkách na dokument JSP použije existujúci dokument, čo zlepšuje výkon. Ak kontajner JSP potrebuje uvoľniť časť pamäte a nikto dokument JSP nepožaduje môže kontajner súbor triedy z pamäti vymazať.

Dá sa povedať že JSP je nadstavba nad servletom. Hlavnou výhodou JSP je množstvo napísaného kódu, ktoré niekoľkonásobne menšie, než keby bol kód napísaný v triede Servletu. Servlety poskytujú na komponentoch založený a platformovo nezávislý spôsob pre vytváranie webových aplikácií. Majú prístup k celej rodine Java API, vrátane JDBC pre komunikáciu s databázami. Servlet je Java trieda, ktorá implementuje rozhranie `javax.servlet.Servlet` a `javax.servlet.http.HttpServlet`.[11]

Výhody JSP a Java Servletov

JSP v porovnaná s konkurenčnými technológiami ponúka niekoľko výhod:

- dynamická časť je napísaná v Jave, čo je komplexný, robustný a bezpečný jazyk,
- Java je platformovo nezávislá,
- prísna typová kontrola,
- rýchlejší beh aplikácie,
- rozsiahle triedy na prácu so sieťou, databázami, elektronickou poštou a pod.

Nevýhody JSP a Java Servletov

Medzi mínusy Javy pre tvorbu webovej aplikácie patria:

- na prevádzku JSP je nutné mať Tomcat alebo iný aplikačný server,

[11] Oracle.com, *The Java EE 6 Turorial : Chapter 15 : Java Servlet Technology*, [online], [2015-04-06] Dostupné na internete: http://docs.oracle.com/javaee/6/tutorial/doc/bnafd.html

- náročnejšia inštalácia prostredia,
- pomalá krivka učenia.

1.3 Databázový systém SQL

Databázový systém je množina vzájomne súvisiacich dát organizovaných v báze dát, spoločne s programovým vybavením - systémom riadenia bázy dát, ktorý umožňuje prístup k dátam.[12] Databázový systém slúži na efektívne ukladanie, modifikáciu a výber veľkého množstva perzistentných údajov. Systém riadenia bázy dát (SRBD), po anglicky database management system (DBMS), musí byť schopný efektívne pracovať s veľkým množstvom dát, riadiť ich (vkladať, modifikovať, mazať) a definovať štruktúru týchto dát. Na definovanie dátových typov a vzťahov medzi nimi slúži jazyk DDL (Data Definition Language). Na vkladanie, úpravu a mazanie údajov sa využíva jazyk DML (Data Manipulation Language). Vyhľadávanie údajov a výpočty v získaných dátach zabezpečuje dopytovací jazyk (Query Language). Súčasné databázové systémy majú okrem týchto základných funkcií aj mnoho ďalších charakteristických vlastností, ako:

- správa primárnych a cudzích kľúčov,
- podpora pre dátový model (napr. relačný, logický, objektový), ktorý oddeľuje konceptuálnu (logickú) úroveň od fyzickej,
- využitie jazyka vyššej úrovne pre manipuláciu a definíciu dát (napr. SQL),
- zabezpečenie integrity databázy,
- robustnosť a zotaviteľnosť po chybách bez straty dát,
- riadenie prístupu pre rôznych používateľov a paralelný prístup,
- správa transakcií,
- riadenie katalógov (schéma databázy).

1.3.1 Relačný databázový systém MySQL

Relačné databázy a jazyk SQL v súčasnosti medzi databázovými systémami dominujú.[13] Relačný databázový model organizuje dáta do tzv. usporiadaných relácií -

[12] KULTÁN, J. 2012. *Databázové systémy : Vybrané kapitoly (Učebnica pre denných a externých študentov netechnických vysokých škôl).* Prvé vydanie. Bratislava : Vydavateľstvo EKONÓM, 2012. 128 s. ISBN: 978-80-225-3350-8

[13] Wikipedia, the free encyclopedia, 2015, *Database*, [online], [2015-03-25] Dostupné na internete: http://en.wikipedia.org/wiki/Database

tabuliek.[14] Každá tabuľka sa skladá z jednotlivých riadkov (rows), označovaných ako záznam (record) a stĺpcov (columns) označovaných ako atribúty. Tabuľky sú navzájom prepojené vzťahmi (relationships). Medzi dvoma tabuľkami môžu existovať tri typy vzťahov - 1:1, 1:N a N:M. Pri rozsiahlych tabuľkách závisí rýchlosť spracovania dotazov na tom, či sa pre jednotlivé atribúty (stĺpce) používa vhodný index - kľúč (key). Špeciálnym druhom indexu je primárny kľúč (primary key), ktorý sa od ostatných indexov líši tým, že predstavuje jednoznačný identifikátor daného záznamu. Na druhej strane cudzí kľúč (foreign key) ukazuje na záznam v podriadenej tabuľke. Primárne a cudzie kľúče slúžia na zabezpečenie referenčnej integrity.

Dopytovací jazyk SQL (Standard Query Language) bol vyvinutý v roku 1974 firmou IBM. Príkazy jazyka SQL sa delia nasledovne:

- definícia dát (DDL) - `CREATE`, `ALTER`, `DROP`
- manipulácia dát (DML) - `INSERT`, `UPDATE`, `DELETE`[15]
- riadenie dát (DCL - Data control language) - `GRAND`, `REVOKE`

MySQL je voľne šíriteľný SQL relačný databázový systém. Bol vytvorený švédskou formou MySQL AB a momentálne je vlastnený firmou Sun Microsystems, dcérskou spoločnosťou Oracle Corporation.[16] Je k dispozícii v rámci bezplatnej licencie GPL a aj pod komerčnou platenou licenciou. Databázový systém MySQL podporuje širokú škálu platforiem. Veľmi dobre spolupracuje s jazykom PHP vďaka čomu je populárny pri webových aplikáciách. Administráciu databázy je možné vykonávať pomocou príkazového riadka, alebo pomocou administrátorských nástrojov s GUI (Graphical user interface) ako je napríklad phpMyAdmin.

MySQL podporuje rôzne typy tabuliek. Medzi najdôležitejšie typy tabuliek patria MyISAM a InnoDB.[17] MyISAM predstavuje štandardný a stabilný typ tabuľky. Pri

[14] KULTÁN, J. 2012. *Databázové systémy : Vybrané kapitoly (Učebnica pre denných a externých študentov netechnických vysokých škôl).* Prvé vydanie. Bratislava : Vydavateľstvo EKONÓM, 2012. 128 s. ISBN: 978-80-225-3350-8

[15] Wikipedia, the free encyclopedia, 2015, *Data manipulation language*, [online], [2015-03-25] Dostupné na internete: http://en.wikipedia.org/wiki/Data_Manipulation_Language

[16] Wikipedia, the free encyclopedia, 2015, *MySQL*, [online], [2015-03-25] Dostupné na internete: https://cs.wikipedia.org/wiki/MySQL

[17] KOFLER, M. - ÖGGL, B. 2007. *PHP 5 a MySQL 5 : Průvodve webového programátora*. Prvé vydanie. Brno : Computer Press, a.s., 2007. 608 s.
ISBN 978-80-251-1813-9

statických tabuľkách, kde nad zapisovaním do tabuľky prevláda čítanie z nej je typ MyISAM rýchlejší. Typ InnoDB je novší a okrem vlastnosti ako má MyISAM podporuje aj transakcie a používanie pravidiel pre integritu. Ďalším z rozdielov je, že zatiaľ čo MyISAM podporuje len uzamykanie celej tabuľky, InnoDB podporuje uzamykanie riadku.

1.3.2 Alternatívy k databázovému systému MySQL

Podobne ako pri programovacích jazykoch, tak aj pri relačných databázových systémoch existuje viacero alternatív. Tie je možné rozdeliť na proprietárne, t.j. licencované alebo platené a voľne šíriteľné. Medzi najznámejšie proprietárne relačné databázy patria DB2 od firmy IBM, Microsoft SQL Server a Oracle Database. S pomedzi voľne šíriteľných relačných databázových systémov sú najpopulárnejšie PostgreSQL, MongoDB a MariaDB.

DB2

IBM DB2 je moderný relačný databázový systém. Je to multiplatformová databáza s pokročilými možnosťami kontroly dát, ich využívania a ochrany. V rodine DB2 existujú tri produkty, ktoré sú si veľmi podobné, ale rozdielne práve v hardvérovom riešení, na ktorom operujú. Konkrétne to sú DB2 pre Linux, UNIX a Windows, DB2 pre z/OS a DB2 pre i.

Produkt DB2 pre Linux, UNIX a Windows je dostupný vo viacerých licenčných dohodách ako napríklad Express Edition, Workgroup Server Edition alebo Enterprise Server Edition. Najviac sofistikovaná edícia pre Linux/UNIX/Windows je Advanced Enterprise Server Edtiodion. Táto edícia je určená pre zmiešané pracovné zaťaženie, ako je OLTP (Online Transaction Processing), dátové skladovanie, alebo business intelligence (BI). Edícia Advanced zahŕňa rôzne BI črty ako je ETL, data mining, OLAP (Online Analytical Processing).

DB2 pre z/OS je dostupná pod jej vlastnými licenčnými podmienkami. Má niektoré exkluzivity, ako sú Multi-Level Security, extrémne veľké rozmery tabuliek a hardvérovo asistovaná kompresia. DB2 pre z/OS je spoľahlivý systém, má vysoký OLTP výkon a dostupnú podporu najdôležitejších operácii pre podnikanie.

Microsoft SQL Server

Microsoft SQL Server je výkonný relačný databázový a analytický systém pre biznis a riešenie dátových skladov. Bol navrhnutý pre zvládnutie veľkého objemu OLTP

transakcií, rovnako ako aj pre skladovanie dát a beh aplikácií. Primárnymi jazykmi sú T-SQL a ANSI SQL.

Existuje mnoho rôznych edícií programu Microsoft SQL Server, ktoré sú zamerané na rôzne cieľové skupiny a pracovné zaťaženie v rozsahu od malých aplikácií až po veľké Internetové aplikácie s mnoho konkurujúcimi si používateľmi. Hlavnými edíciami vo verzii SQL Server 2014 sú Express, Web, Developer, Standard, Business Intelligence a Enterprise. Existujú aj špecializované edície ako napríklad Azure SQL Database pre cloud.

Oracle Database

Oracle je multiplatformový objektovo-relačný systém riadenia bázy dát, s pokročilými možnosťami spracovania dát, vysokým výkonom a jednoduchou škálovateľnosťou. K dispozícií sú rôzne edície Oracle Database, od Express Edition, ktorá je obmedzená počtom procesorov, veľkosťou operačnej pamäte a veľkosťou databázy, cez Standard Edition One a Standard Edition, ktoré sú limitovaná počtom procesorov až po edíciu Enterprise, ktorá nemá výkonnostné obmedzenia.

Oracle Database podporuje nielen štandardný relačný dopytovací jazyk SQL, ale aj proprietárne firemné rozšírenie Oracle, napríklad na hierarchické dopyty, programovací jazyk PL/SQL rozširujúci možnosti vlastného SQL. V tomto jazyku je možné tvoriť uložené procedúry, používateľské funkcie, programové balíky a spúšťať triggery. Podporuje i objektové databázy a databázy uložené v hierarchickom modeli dát, ako napríklad XML databázy a jazyk XSQL.

PostgreSQL

PostgreSQL je voľne šíriteľný objektovo-relačný databázový systém. PostgreSQL je primárne vyvíjaný pre Linux, resp. unixové systémy všeobecne, ale existuje aj pre platformu Windows.

Konverzia údajov zo sveta relačných databáz do objektovo-orientovaného programovania prináša ťažkosti, pretože tieto svety majú veľmi rozdielnu organizáciu dát. PostgreSQL sa vie vysporiadať s mnohými problémami na úrovni databázy. Dovoľuje používateľovi definovať si nové typy takmer všetkých objektov v databáze, medzi ktoré patria dátové typy, funkcie, indexy, alebo operátory. V PostgreSQL je možné napísať funkcie a triggery v zabudovanom jazyku PL/pgSQL, ktorý pripomína procedurálny jazyk PL/SQL, alebo iných jazykoch ako PL/Perl, C, C++, Java.

MongoDB

MongoDB je voľne dostupná multiplatformová dokumentovo orientovaná NoSQL databáza. Podporované operačné systémy sú Apple Mac OS X, Linux, Microsoft Windows a Solaris. MongoDB sa odlišuje od "klasických" relačných databáz tým, že ukladá štruktúrované dáta ako JSON dokumenty s dynamickými schémami (MongoDB nazýva tento formát BSON). Vďaka tomu je práca s určitými typmi dát rýchlejšia a jednoduchšia.

Dopyty v MongoDB môžu vrátiť špecifické oblasti dokumentu a môžu obsahovať server-side funkcie v jazyku JavaScript, ktoré sú vykonané priamo v databáze. Ako komunikačný prostredník medzi samotnou databázou a programovacím jazykom v ktorom je aplikácia napísaná slúži ovládač (driver). Dostupné sú knižnice pre najznámejšie jazyky ako Java, Ruby, Python, PHP, Perl či C++.

MariaDB

Databáza MariaDB vznikla ako komunitne vyvíjaná odnož MySQL v časoch, keď MySQL kúpila spoločnosť Sun Microsystems. Väčšina funkcií je pri oboch databázach rovnaká. Vďaka mnohým zlepšeniam jadra má MariaDB lepší výkon ako MySQL. MariaDB ponúka aj nové funkcie, ktoré v MySQL chýbajú.

Na rozdiel od MySQL, môže MariaDB získať dáta navrátené vo formáte JSON za použitia dynamických stĺpcov. MariaDB ponúka klastrovaciu technológiu a vďaka úložným enginom aj integráciu dát s NoSQL.

1.4 Charakteristika a popis navrhovaného skladového systému

Rozhodli sme sa navrhnúť a vytvoriť nový skladový systém pre firmu Starfit a integrovať tento systém do existujúceho webového prostredia firmy. Nový systém by mal oproti tomu starému zjednodušiť používanie a tým zvýšiť efektívnosť a predovšetkým umožniť vzdialený prístup do systému integrovaním do webového prostredia firmy.

1.4.1 Popis firmy a súčasného riešenia

Firma Starfit sa zaoberá prevádzkou fitness a aerobic centra v Bratislave. Vo svojich priestoroch taktiež poskytuje občerstvenie, nápoje, ovocie a výživové doplnky pre návštevníkov. Prevádzka je rozdelená do viacerých priestorov, ako telocvičňa, miestnosť pre skupinové cvičenia, šatňa, sprchy a wc, pokladňa, predajňa, sklad a kancelária vedúceho. Pre potreby skladového systému sú zaujímavé len predajňa a sklad, ako miesta

prevádzky navrhovaného skladového systému a potom pokladňa a kancelária ako prístupové miesta pre obsluhu skladového systému.

V súčasnosti firma používa na dennú evidenciu pohybu tovaru v sklade a v predajni tabuľky v Excely, ktoré potrebné výpočty robia pomocou makier. Terajší systém tvoria dva súbory - Sklad a Predajňa. Sklad predstavuje tabuľka, ktorá obsahuje zoznam tovaru. Ďalej obsahuje päť stĺpcov: Počiatočný stav, Príjem, Výdaj, Odpis/Oprava, Koncový stav. Počiatočný stav je počet kusov daného tovaru na začiatku dňa. Pri dodaní nového tovaru na sklad sa počet kusov zapíše do stĺpca Príjem. Stĺpec Výdaj predstavuje presun tovaru zo skladu do predajne. Stĺpec Odpis/Oprava slúži ako opravná položka v prípade že sa zistí nesúlad medzi skutočným počtom kusov daného tovaru a počtom kusov v zázname o sklade. Posledný stĺpec, Koncový stav, je rozdiel medzi počiatočným stavom, príjmom na sklad, výdajom zo skladu a odpisu/opravy. Koncový stav je na konci dňa automaticky vypočítaný pomocou makra a prenesený ako počiatočný stav do dňa nasledujúceho. V Tabuľka 1: **Ukážka tabuľky Tovar v súbore Sklad** je znázornená ukážka súboru Sklad, ktorý firma používa.

Tabuľka 1: Ukážka tabuľky Tovar v súbore Sklad

Tovar	PS	Príjem	Výdaj	Odpis/Oprava	KS
Aloe	0				0
Isofruit	6		1		5
Burger	0	10	9	1	0
50% Protein Bar	79		3		76
Banány	9		9		0

Zdroj: Firma Starfit

Súbor Predajňa obsahuje viacero tabuliek. Hlavnou tabuľkou je zoznam tovaru, ktorého ukážka je znázornená v Tabuľka 2: **Ukážka tabuľky Tovar v súbore Pokladňa.** Pre každú položku sú tu opäť Počiatočný stav, Príjem, Koncový stav a Predaj. Počiatočný stav predstavuje počet kusov na začiatku dňa. Príjem je automatický doplnený zo stĺpca Výdaj v tabuľke Sklad. To znamená že výdaj zo skladu je príjem do predajne. Koncový stav je počet kusov tovaru na konci dňa. Stĺpec Predaj je rozdiel medzi počiatočným stavom, príjmom a koncovým stavom. V súčasnosti tento systém funguje tak, že zamestnanec firmy na konci každého dňa spočíta počet kusov všetkých tovarov v predajni a zapíše tento údaj do stĺpca Koncový stav. Ostatné stĺpce (Počiatočný stav, Príjem, Predaj) sú automatický doplnené alebo dopočítané pomocou makra na konci dňa. Ďalšími

tabuľkami v súbore Pokladňa sú Kasa Start a Kasa End, ktoré sú znázornené v Tabuľka 3: **Ukážka tabuľky Kasa Start** a Tabuľka 4: **Ukážka tabuľky Kasa End**. Tieto dve tabuľky obsahujú počet bankoviek a mincí jednotlivých nominálnych hodnôt a k ním prislúchajúce hodnoty v eurách. Tieto údaje zamestnanec taktiež vyplní na konci dňa. V súbore sa nachádza ešte niekoľko ďalších tabuliek ako suma hotovosti a suma platieb kartou podľa bločkov v pokladni, výpočet tržby za daný deň a meno zamestnanca, ktorý robil dennú uzávierku.

Tabuľka 2: Ukážka tabuľky Tovar v súbore Pokladňa

Tovar	PS	Príjem	KS	Predaj
Aloe	8	0	8	0
Isofruit	24	1	24	1
Burger	5	9	1	13
50% Protein Bar	13	3	14	2
Banány	5	9	4	10

Zdroj: Firma Starfit

Nevýhodou súčasného riešenia je, že tabuľky sú dostupné iba lokálne na počítači vo firme. Ďalej, že toto riešenie nie je používateľsky prívetivé. Neposkytuje dostatočné možnosti na vyhľadávanie a editáciu záznamov. Nie je možnosť prehľadávania v historických údajoch, t.j. možnosť prezretia si stavu k určitému dňu v minulosti. A v neposlednom rade nevýhodou je aj to, že program Excel od firmy Microsoft je platený. Z tohto dôvodu prešla firma na náhradné riešenie a používa open source softvér OpenOffice.

Firma Starfit by chcela nedostatky ich súčasného spôsobu evidencie tovaru a tržieb vyriešiť novým skladovým systémom. Nový systém by mal poskytovať aspoň tie funkcie, ako terajší systém. Navyše by mal byť integrovaný do webovej stránky firmy, tak aby si majiteľ firmy po prihlásení sa do administrátorského rozhrania webu, mohol cez Internet pozrieť aktuálne údaje o skladne, predajni a tržbách.

Tabuľka 3: Ukážka tabuľky Kasa Start

Hodnota	Počet	Suma
0,01 €	31	0,31 €
0,02 €	41	0,82 €
0,05 €	77	3,85 €
0,10 €	70	7,00 €
0,20 €	38	7,60 €
0,50 €	18	9,00 €
1,00 €	4	4,00 €
2,00 €	52	104,00 €
5,00 €	3	15,00 €
10,00 €	12	120,00 €
20,00 €	2	40,00 €
50,00 €		0,00 €
100,00 €		0,00 €
200,00 €		0,00 €
500,00 €		0,00 €
	Spolu:	311,58 €

Zdroj: Firma Starfit

Tabuľka 4: Ukážka tabuľky Kasa End

Hodnota	Počet	Suma
0,01 €	33	0,33 €
0,02 €	48	0,96 €
0,05 €	77	3,85 €
0,10 €	73	7,30 €
0,20 €	44	8,80 €
0,50 €	14	7,00 €
1,00 €	5	5,00 €
2,00 €	37	74,00 €
5,00 €	1	5,00 €
10,00 €	5	50,00 €
20,00 €	4	80,00 €
50,00 €		0,00 €
100,00 €		0,00 €
200,00 €		0,00 €
500,00 €		0,00 €
	Spolu:	242,24 €

Zdroj: Firma Starfit

1.4.2 Existujúce produkty pre skladové hospodárstvo

Na slovenskom trhu existuje viacero produktov, ktoré slúžia na účely skladového hospodárstva. Najznámejšie a najrozšírenejšie sú program POHODA od firmy STORMWARE, programy ALFA plus a OMEGA od firmy KROS, program OASIS od firmy SunSoft Plus, alebo program MRP. Výhodou týchto produktov je komplexná sada funkcií, ktoré poskytujú a možnosť prepojenia na príslušný účtovný softvér.

POHODA od firmy STORMWARE

Skladové agendy ekonomického a informačného systému POHODA slúžia dvom základným účelom:

- na reálne vedenie skladov,
- na jednoduché vystavovanie položkových dokladov.

POHODA umožňuje vedenie neobmedzeného počtu skladov a ich členenie podľa vlastných potrieb. Podporuje:

- evidenciu tovaru, materiálu, služieb, ale i tzv. súprav, kompletov a výrobkov,
- záruky, šarže a výrobné čísla,
- používanie čiarových kódov,
- zľavy, cenové hladiny, individuálne cenníky,
- účtovanie zásob metódou A alebo B,
- sledovanie reklamácií a opráv,
- priamy predaj tovaru pomocou zabudovanej predajne Kasa Online,
- predaj tovaru cez Internet,
- automatické objedávanie zásob,
- výkazy pre Intrastat,
- používanie náhľadu tovaru.

ALFA plus a OMEGA od firmy KROS

Program ALFA plus resp. OMEGA je moderný účtovný softvér. Sklad v programe ALFA plus alebo OMEGA umožňuje viesť kompletné skladové hospodárstvo firmy metódou oceňovania FIFO alebo priemerom. Malý sklad je obmedzený 50 skladovými kartami. Veľký sklad obsahuje neobmedzený počet skladov a skladových kariet. Skladové karty sa môžu priradiť do skupín (podľa kategórie tovaru). V sklade je možné nastaviť cenové úrovne s troma možnosťami tvorby cien: fixná predajná cena, priemerná cena s maržou a FIFO cena s maržou. Sklad je prepojený s elektronickými registračnými pokladnicami Elcom, fiškálnymi modulmi a tlačiarňami VAROS TRADE, umožňuje vzájomný prenos údajov a vytlačenie bločku priamo z programu. Každý sklad má širokú ponuku prehľadov a tlačových zostáv.

OASIS od firmy SunSoft Plus

Programový modul OASIS obsahuje plne prepojený skladový, nákupný, odbytový, fakturačný a objednávkový systém. Umožňuje sieťové aj pobočkové nasadenie od maloobchodných predajní, cez supermarkety až po náročné veľkoobchody. Po prepojení s modulom podvojného účtovníctva umožňuje automatickú prípravu a spracovanie 95% mesačných účtovných zápisov. OASIS poskytuje komplexný systém skladových pohybov viazaný na evidenciu skladových a materiálových kariet. Skladové pohyby sú tesne napojené na objednávkový a fakturačný modul.

MRP

MRP je ucelený účtovno-informačný systém. Sklady v programe MRP ponúkajú výstupné zostavy ako napr. príjemka a výdajka, zoznam pohybov, obraty podľa druhov pohybov, zoznamy kariet, inventúry, cenníky, a pod. V programe je implementovaná podpora pre hlásenia INTRASTAT s možnosťou exportu do XML súboru. V skladoch je možnosť vytvárania zložených kariet v niekoľkých úrovniach a kariet s náväznosťou na iné karty. Cenník je možné definovať pre rôzne meny. Samozrejmosťou je vytváranie skladových dokladov z objednávok, či možnosť prenosu skladových dokladov, načítanie skladových dokladov z pomocných dokladov alebo textových súborov.

Možnosti, ktoré tieto programy poskytujú, sú nad rámec požiadaviek firmy Starfit. Nevýhodou je aj pomerne zložité ovládanie do ktorého treba používateľov zaškoliť. Jedným z hlavných nedostatkov je, že prístup do programu je možný len priamo z počítača, na ktorom je program nainštalovaný, prípadne z lokálnej siete. Priamy prístup cez Internet je možný len v prípade niektorých programov a len pri drahších licenciách. A v neposlednom rade nevýhodou existujúcich riešení pre skladové hospodárstvo je aj ich cena, ktorá sa pohybuje rádovo v stovkách eur.

2 Cieľ a metodika práce

Táto kapitola charakterizuje predmet riešenia, ako aj čiastkové ciele, ktoré podmieňujú dosiahnutie cieľa hlavného.

Cieľ práce

Hlavným cieľom práce je navrhnúť a vytvoriť webovú aplikáciu skladového hospodárstva, na základe špecifických požiadaviek firmy Starfit.

Počas osobných stretnutí so zodpovednými osobami z firmy Starfit sme získali informácie o súčasnom systéme skladového hospodárstva, ktorý firma používa ako aj o jeho nedostatkoch. K dosiahnutiu hlavného cieľa diplomovej práce sme si stanovili nasledujúce čiastkové ciele. Na základe informácii o existujúcom riešení skladového systému sme identifikovali potreby firmy. Ďalším čiastkovým cieľom je navrhnutie nového skladového systému. Následne na základe teoretických poznatkov o PHP prezentovaných v prvej kapitole a na základe vlastných skúseností, naprogramovať navrhnutý skladový systém a na základe poznatkov o MySQL navrhnúť a implementovať databázu pre tento systém. Posledným čiastkovým cieľom je integrácia navrhnutého a vytvoreného skladového systému do existujúceho webového prostredia firmy.

Metodika práce a metódy skúmania

Pri písaní tejto práce sme využili rôzne metódy výskumu na dosiahnutie nami stanovených cieľov.

Na začiatku bolo nutné zabezpečiť si zdroje informácií vo forme slovenskej i zahraničnej literatúry. Následne sme na získané údaje aplikovali systémovú analýzu. syntézu a dedukciu. Tieto metódy sú neoddeliteľnou súčasťou výskumu. Analýza rozčleňuje celok na základné časti, vzťahy a súvislosti. Túto metódu sme využili napríklad pri oboznamovaní sa s jazykom PHP, jeho charakteristikami a objektovom programovaní v tomto jazyku. Následne pomocou syntézy, ktorá analýzou vyčlenené časti, vzťahy a súvislosti spája a zjednocuje do určitého celku, sme zhrnuli výhody a nevýhody tohto jazyka do niekoľkých bodov. Obdobne sme postupovali aj pri ďalších programovacích jazykoch opísaných v teoretickej časti tejto práce. Analýza bola použitá aj pri skúmaní súčasného riešenia skladového systému firmy Starfit a následne, pomocou dedukcie, boli praktickej časti špecifikované jednotlivé požiadavky na skladový systém.

Na návrh štruktúry databázy a webovej aplikácie sme v praktickej časti použili nasledujúce postupy:

- dátové modelovanie a analýza pomocou UML diagramov,
- modelovanie používateľských prípadov,
- modelovanie používateľského rozhrania,
- návrh doménového a databázového modelu.

Okrem spomenutých metód skúmania je potrebné mať nainštalované programové vybavenie. Na vizualizáciu informácií, tvorbu diagramov a UML zápis návrhu systému je použitý program Enterprise Architect. Webová aplikácia a databáza bude umiestnená na webovom serveri Apache, s podporou jazyka PHP a databázovým systémom MySQL. Na správu databázy je použitý administrátorský nástroj phpMyAdmin. Webová aplikácia je vyvíjaná vo voľne šíriteľnom textovom editore Notepad++.

3 Výsledky práce a diskusia

Táto kapitola predstavuje praktickú časť diplomovej práce a ukážeme si v nej vývoj webovej aplikácie od analýzy, návrhu, až po jej implementáciu. Následne si uvedieme využite objektov a ich implementáciu vo vytvorenej webovej aplikácii, ktorej funkčnosť predvedieme v poslednej časti tejto kapitoly.

Postup pri výstavbe webovej aplikácie je nasledovný:

1. Analýza požiadaviek používateľov (Requirements Model)
2. Definovanie služieb a ich priebehu (Use Case Model)
3. Návrh doménového modelu (Domain Model)
4. Návrh databázy (Database Model)
5. Návrh používateľského prostredia (User Interface Model)
6. Implementácia databázy v MySQL
7. Naprogramovanie webovej aplikácie v PHP
8. Overenie funkcionality celého systému
9. Integrácia systému do existujúceho webového prostredia firmy
10. Prevádzkovanie skladového systému a jeho ďalší rozvoj

3.1 Analýza požiadaviek (Requirements)

Prvým krokom pri vývoji webovej aplikácie je analýza požiadaviek používateľov, ktorými sú v našom prípade zamestnanci firmy. Navrhovaný skladový systém bude pracovať s údajmi, ktoré je možné rozdeliť do niekoľkých základných skupín. Prvou skupinou je zoznam tovaru, druhou sú aktuálne zásoby tohto tovaru na sklade a v predajni, a treťou skupinou je pohyb tovaru ako napríklad príjem, výdaj alebo predaj.

V prvom rade je nutné zadefinovať si, kto sú používatelia navrhovaného skladového systému. Pracovať so skladovým systémom budú môcť len zamestnanci firmy. Zamestnanci sú rozdelení do dvoch skupín - zamestnanci a šéf. Šéf bude mať okrem možností aké majú zamestnanci aj určité práva navyše. Požiadavkou firmy je, aby nový skladový systém bol integrovaný do webu firmy a využil tak existujúci spôsob autentifikácie zamestnancov a zároveň umožnil prezerať si údaje o sklade a predajni cez Internet.

Na začiatku je potrebné vytvoriť zoznam tovarov. Systém má umožniť používateľom pridávať tovar do zoznamu, editovať ho alebo vymazať.

Nasledujúcou požiadavkou je príjem tovaru na sklad. Systém má ponúknuť používateľovi možnosť vybrať si konkrétny tovar a zadať množstvo, t.j. počet kusov naskladňovaného tovaru.

Skôr ako bude môcť byť tovar predaný zákazníkovi, je nutné presunúť ho do predajne. Používateľ má mať v systéme možnosť urobiť výdaj tovaru zo skladu a príjem tohto tovaru do predajne. Zároveň má mať možnosť zvoliť si množstvo presúvaného tovaru. Systém má zobrazovať aktuálny stav tovaru na sklade a v predajni.

Systém má byť schopný vytvoriť aj Odpis, ktorý slúži ako opravná položka, pre prípad že sa inventúrou zistí nesúlad medzi skutočným počtom tovaru na sklade a záznamom v systéme.

Evidencia tovaru v predajni funguje trochu inak ako evidencia v sklade. Zodpovedný pracovník firmy na konci každého dňa robí inventúru tovaru v predajni. To znamená, že fyzicky spočíta počet kusov z každého tovaru, ktorý sa nachádza v predajni. Systém má poskytovať možnosť zaznamenať tento počet ako koncový stav daného dňa. Systém má zaznamenať meno zamestnanca, ktorý dennú inventúru tovaru v predajni robil.

Pracovníci firmy každý deň zapisujú aj stav hotovosti v kase a stav bločkov v elektronickej pokladni. Existujú dva typy bločkov a to pokladničný blok hotovosť a pokladničný blok platobná karta. Navrhovaný systém má umožniť zaznamenať koncový stav hotovosti a bločkov. Systém má zaznamenať meno zamestnanca, ktorý koncový stav hotovosti a bločkov v predajni zapisoval.

Druhým typom funkcionality, ktorú má systém ponúkať je vytváranie reportov. Prvým reportom je stav skladu k určitému dňu. Pri požiadavke na zobrazenie stavu skladu, má systém zobraziť zoznam tovaru a pre každý tovar zobraziť stav na začiatku dňa, príjem, výdaj, odpis a koncový stav v danom dni.

Druhým dôležitým reportom je stav tovaru v predajni k určitému dňu. Podobne ako pri reporte o stave skladu, aj pri reporte o stave predajne má systém zobraziť zoznam tovaru a počiatočný stav. V ďalších stĺpcoch má systém zobraziť príjem tovaru na predajňu, množstvo predaného tovaru a stav ku koncu daného dňa.

Pre report o stave hotovosti v kase, má systém vytvoriť zoznam mincí a bankoviek jednotlivých nominálnych hodnôt a podobne ako pri tovare, zobrazí počiatočný a koncový

stav. V tabuľke o hotovosti má byť navyše ešte stĺpec suma, ktorý je súčinom nominálnej hodnoty a počtu kusov jednotlivých mincí a bankoviek. Na konci tabuľky má byť zobrazená celková suma hotovosti v kase na začiatku a na konci dňa.

Pre report o bločkoch má systém zobraziť jednoduchú tabuľku, kde jeden údaj predstavuje platbu v hotovosti a druhý platbu platobnou kartou. Tieto dva údaje predstavujú stav na konci dňa. Počiatočný stav bločkov je vždy nulový.

Najdôležitejšou informáciou pre šéfa firmy je denná tržba. Tento report má obsahovať údaje o hotovosti, tržbe a sprepitnom. Sprepitné je rozdiel medzi hotovosťou v kase a údajom na bločkoch z registračnej pokladnice.

Zhrnutie požiadaviek

Podľa požiadaviek firmy Starfit má systém umožňovať nasledovné funkcie:

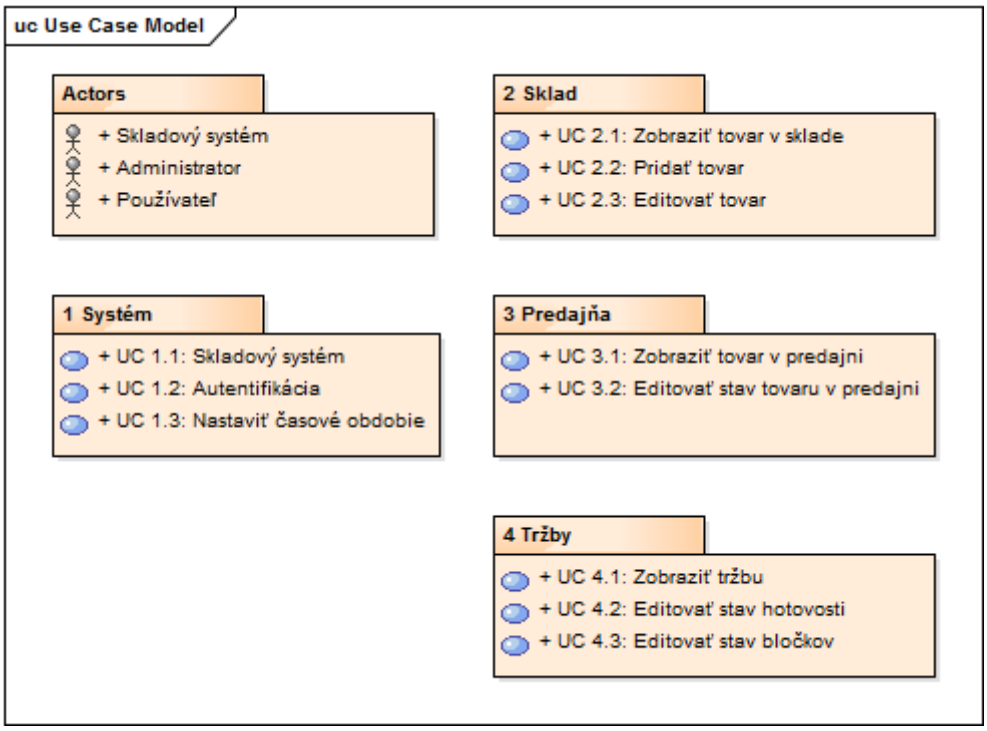
1. pridať tovar to systému,
2. editovať tovar v systéme,
3. zmazať (deaktivovať) tovar zo systému,
4. prijať tovar na sklad,
5. vydať tovar zo skladu do predajne,
6. zapísať opravnú položku (odpis) tovaru na sklade,
7. zaznamenať stav tovaru v predajni na konci dňa,
8. zaznamenať stav hotovosti v predajni na konci dňa,
9. zaznamenať stav bločkov z pokladne na konci dňa,
10. zaznamenať meno zamestnanca, ktorý zapisoval koncové stavy v predajni,
11. zobraziť počiatočný stav, príjem, výdaj, odpis a koncový stav tovarov v sklade k určitému dňu,
12. zobraziť počiatočný stav, príjem, koncový stav a predaj tovaru v predajni k určitému dňu,
13. zobraziť počiatočný a koncový stav hotovosti a sumu hotovosti k určitému dňu,
14. zobraziť koncový stav bločkov z elektronickej pokladne k určitému dňu,
15. zobraziť informácie o dennej tržbe k určitému dňu.

3.2 Návrh systému

Pri návrhu systému sme jednotlivé požiadavky najprv spracovali do formy prípadov použitia. Prípady použitia sú na základe funkcionality rozdelené do štyroch modulov a to Systém, Sklad, Predajňa a Tržby. Po detailnom rozpracovaní všetkých funkcionalít skladového systému sme navrhli základné entity doménového modelu, ktorými sú sú Tovar a Pohyb. V diagrame tried, si definujeme všetky triedy, ich atribúty a metódy. Posledným krokom pri návrhu systému je návrh používateľského prostredia.

3.2.1 Definovanie služieb a ich priebehu (Use Case Model)

Model prípadov použitia (Use Case Model) zobrazuje správanie systému, tak ako ho vidí používateľ. Účelom diagramu je popísať funkcionalitu systému. Use case model sa skladá z prípadov použitia (use cases), aktérov (actors) a vzťahov medzi nimi. Model prípadov použitia pre navrhovaný skladový systém je znázornený na Obrázok 1: **Use Case Model**.



Obrázok 1: Use Case Model

Zdroj: Vlastné spracovanie

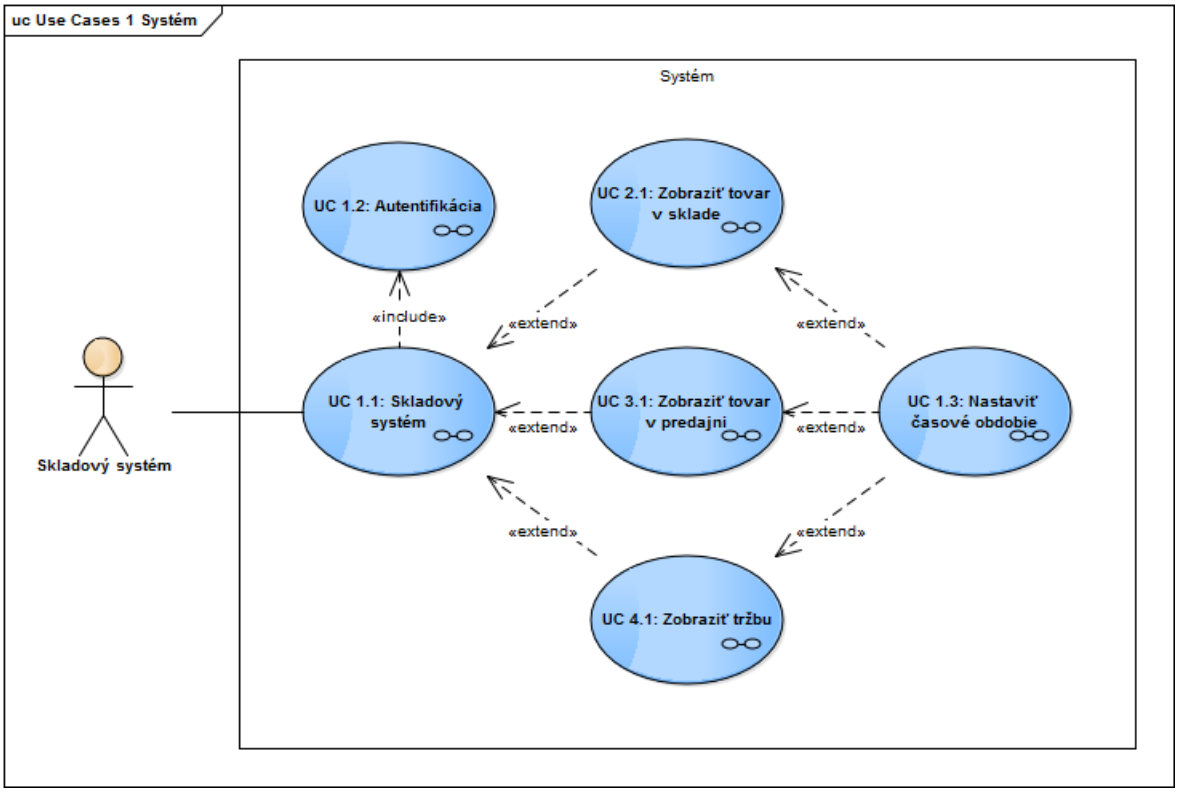
Aktéri (Actors)

Aktér je rola, ktorá komunikuje s jednotlivými prípadmi použitia. Aktér je teda prepojený s prípadmi použitia ktoré sa ho týkajú. Táto väzba je v diagrame prípadu

použitia znázornená jednoduchou čiarou a nazýva sa asociácia. Základnou rolou v navrhovanom skladovom systéme je Používateľ, ktorý predstavuje prístup pre zamestnanca firmy do systému. K dispozícii má funkcionality ako sú zobrazenie tovaru v sklade a predajni alebo pridanie nového záznamu do databázy. Šéf firmy vystupuje ako Administrátor. Rola Administrátor dedí všetky väzby z role zamestnanec. Tento vzťah je v diagrame znázornený prázdnou uzavretou šípkou smerom k predkovi a nazývame ho generalizácia. Administrátor na rozdiel od zamestnanca má niekoľko prípadov použitia navyše. Napríklad úpravu pohybu tovaru za predchádzajúce dni. Funkcionality sprístupnené jednotlivým roliam use case modelu je možné meniť na základe aktuálnych potrieb firmy. Tretím aktérom je skladový systém, ktorý predstavuje obsluhu základných služieb systému a prepojenie s existujúcu databázu firmy.

Modul Systém

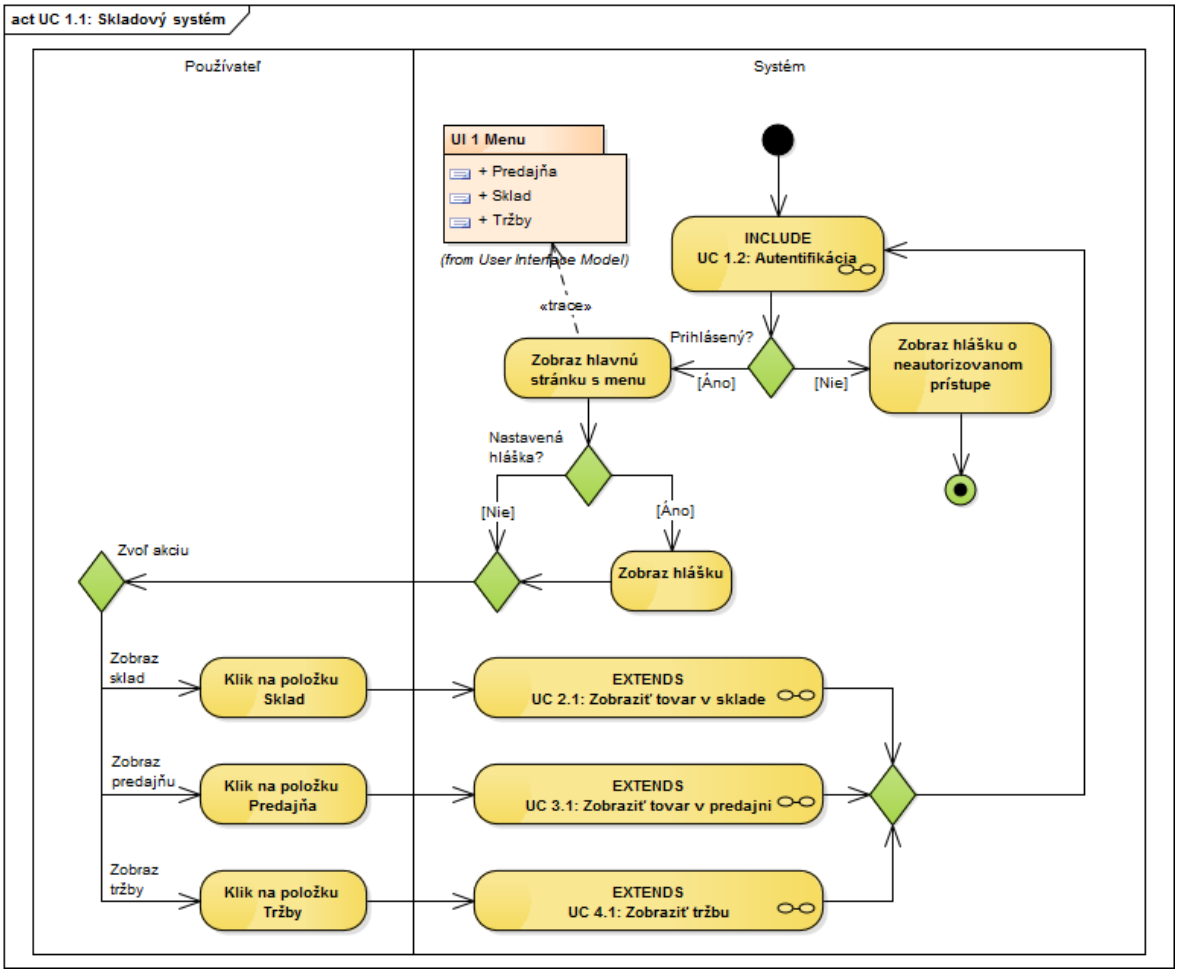
Modul Systém, znázornený na Obrázok 2: **Use Cases 1 Systém**, je hlavnou časťou aplikácie. V prvom rade preberá autentifikáciu používateľa z externého systému, ktorým je webová stránka firmy a na základe role prihláseného používateľa sprístupňuje jednotlivé služby celého skladového systému.



Obrázok 2: Use Cases 1 Systém

Zdroj: Vlastné spracovanie

Na navigáciu na stránke používateľ využíva menu, z ktorého sa dá dostať do všetkých modulov skladového systému. Spôsob navigácie na stránke a sprístupňovanie jednotlivých služieb skladového systému znázorňuje diagram aktivít na Obrázok 3: **Activity Diagram pre Use Case 1.1 Skladový systém**.

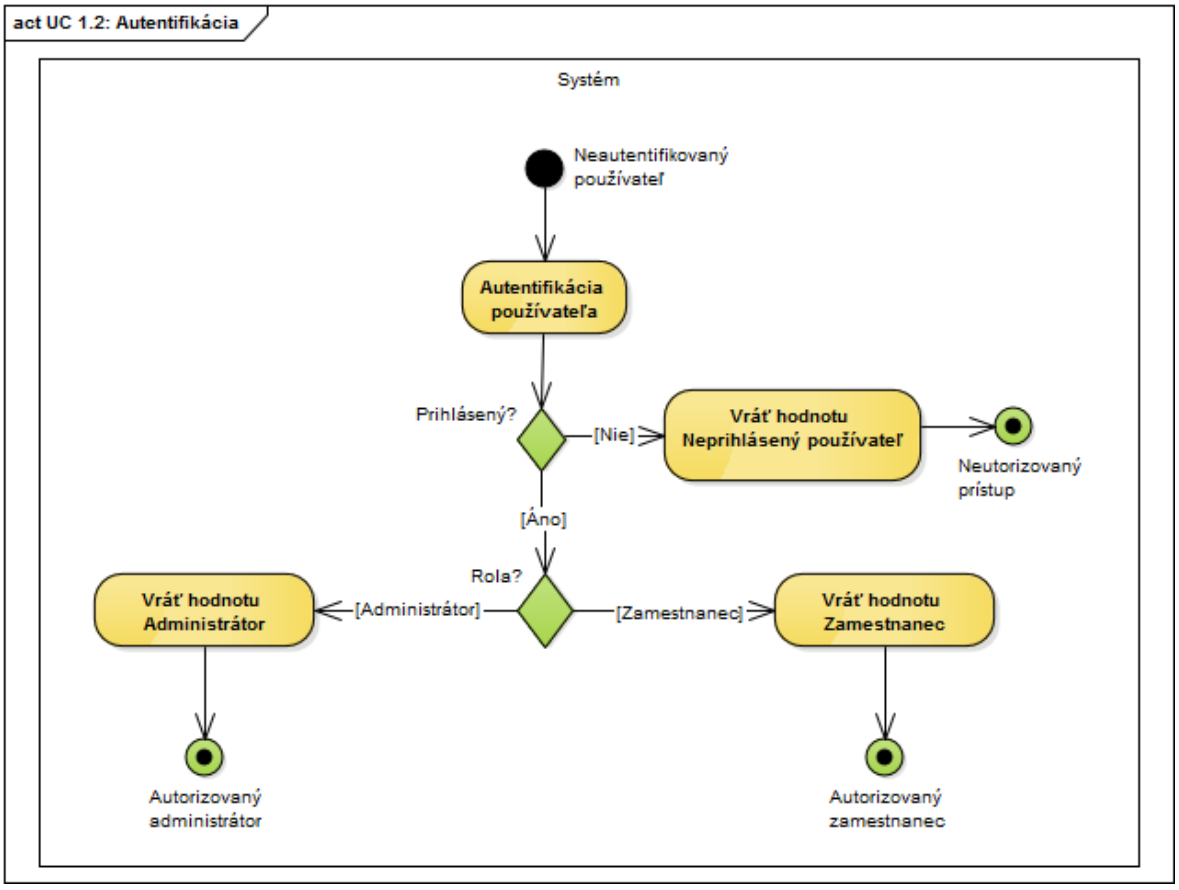


Obrázok 3: Activity Diagram pre Use Case 1.1 Skladový systém

Zdroj: Vlastné spracovanie

Proces autentifikácie je zobrazený v diagrame aktivít na Obrázok 4: **Activity Diagram pre Use Case 1.2 Autentifikácia**4. Na začiatku vstupuje do systému neautentifikovaný používateľ webovej stránky firmy. Systém si najprv overí či je používateľ prihlásený. Prihlasovanie a odhlasovanie prebieha v externom systéme, ktorým je webové prostredie firmy. To slúži klientom a zamestnancom firmy na využívanie a manažovanie rôznych obchodných služieb ktoré firma poskytuje. V prípade že sa prihlásený používateľ autorizuje ako administrátor alebo zamestnanec firmy, systém mu

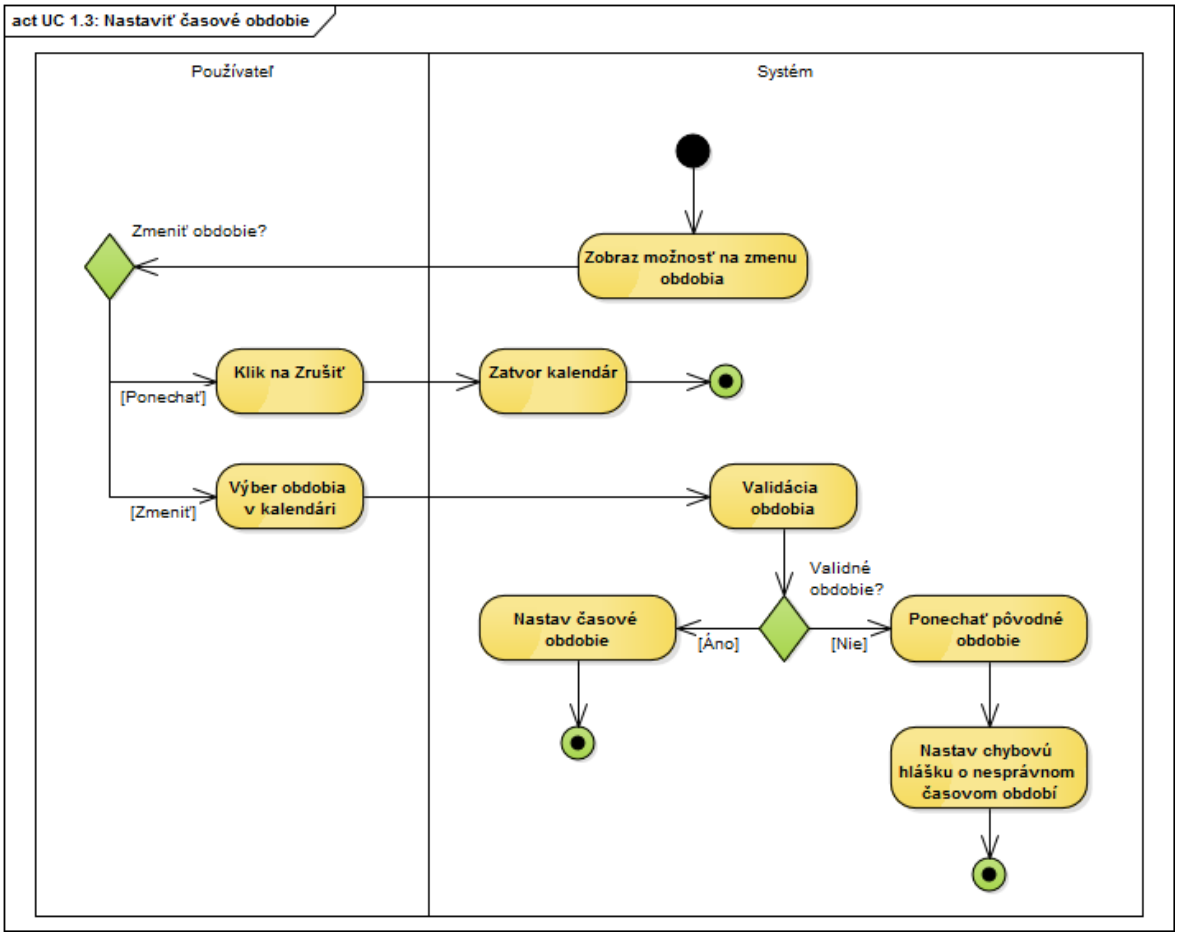
sprístupní svoje služby prostredníctvom menu, ktoré je integrované do existujúceho webu firmy. Neautorizovaná osoba nemá žiadny prístup do skladového systému.



Obrázok 4: Activity Diagram pre Use Case 1.2 Autentifikácia

Zdroj: Vlastné spracovanie

Hlavný systémový modul ďalej obsluhuje časové obdobie nastavené v systéme. Údaje v sklade a v predajni sa zobrazujú v časovom rozsahu, ktorý si môže používateľ zmeniť. Základným obdobím systému je aktuálny deň. Proces nastavenia časového obdobia je znázornený v diagram aktivít na Obrázok 5: **Activity diagram pre Use Case 1.3 Nastaviť časové obdobie**.



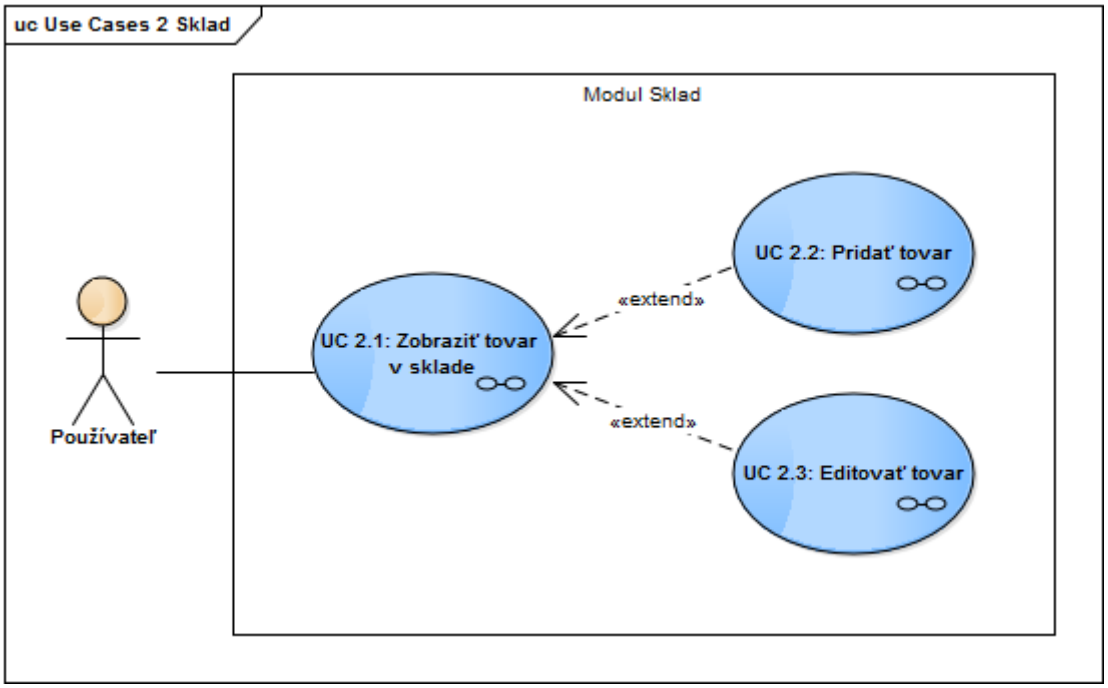
Obrázok 5: Activity diagram pre Use Case 1.3 Nastaviť časové obdobie

Zdroj: Vlastné spracovanie

Modul Sklad

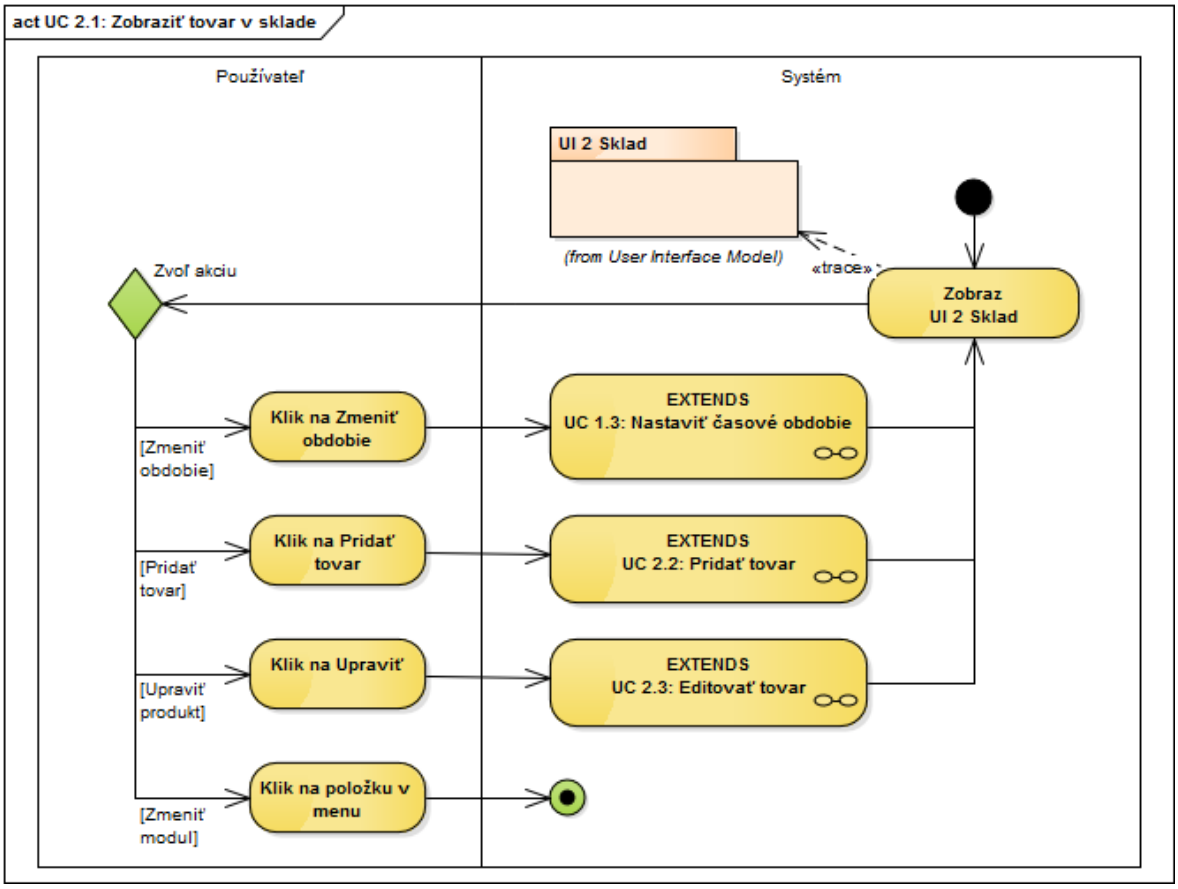
Modul Sklad pokrýva všetku funkcionalitu skladu, ktorou je zobrazenie, pridanie a editovanie tovaru v skladovom systéme. V diagrame na Obrázok 2: Use Cases 1 S sú znázornené prípady použitia v module sklad, ako aj aktéri a vzťahy medzi prvkami tohto diagramu.

Hlavnou úlohou modulu Sklad je zobrazenie tovaru v sklade a poskytnutie možnosti pre používateľa na editáciu tovaru a pridanie nového tovaru do skladu. Priamo v module Sklad si používateľ môže zmeniť časové obdobie. Zobrazenie stránky a sprístupnenie jednotlivých služieb skladu je znázornené v diagrame na Obrázok 7: **Activity Diagram pre Use Case 2.1 Zobraziť tovar v sklade**.



Obrázok 6: Use Cases 2 Sklad

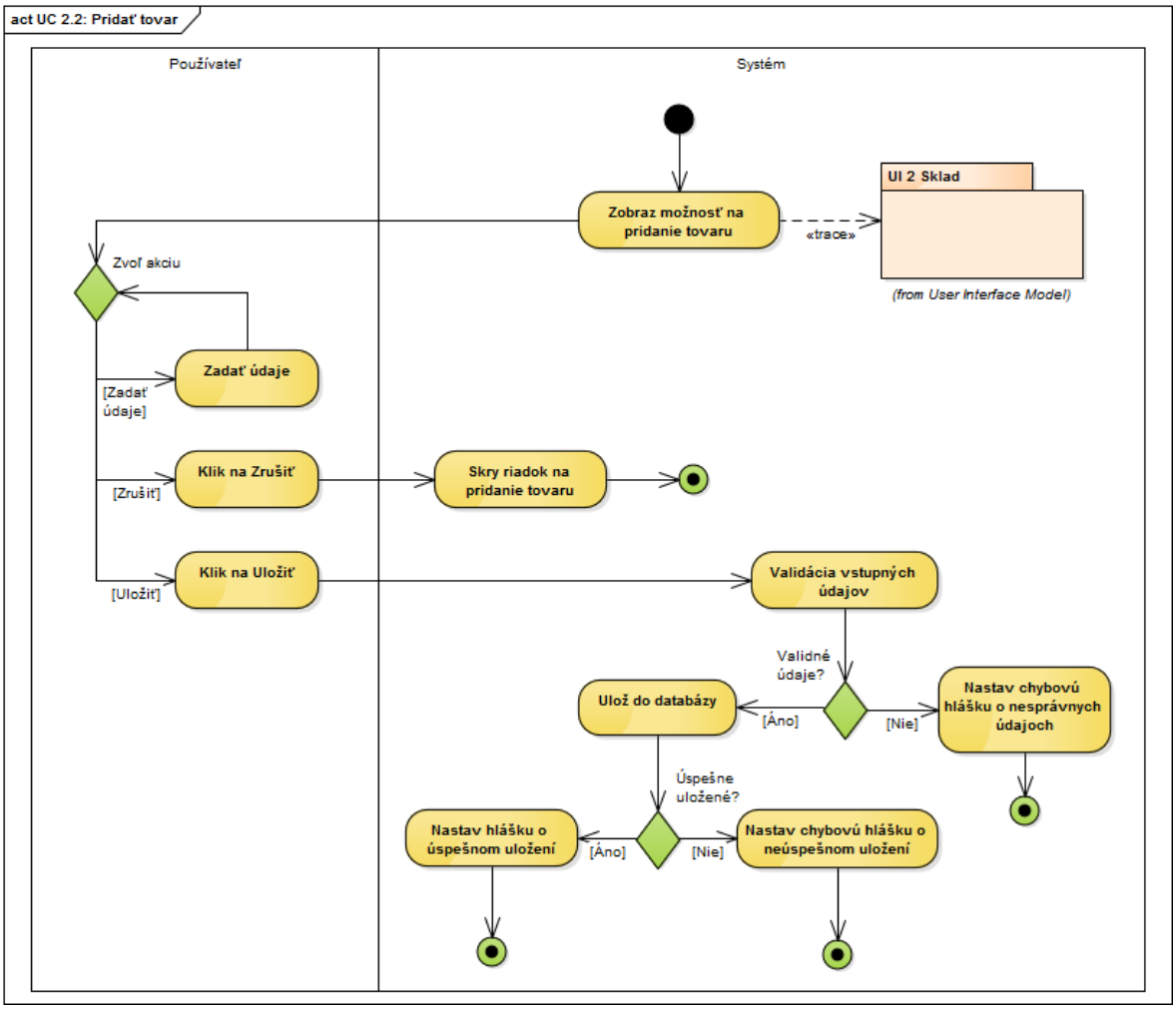
Zdroj: Vlastné spracovanie



Obrázok 7: Activity Diagram pre Use Case 2.1 Zobraziť tovar v sklade

Zdroj: Vlastné spracovanie

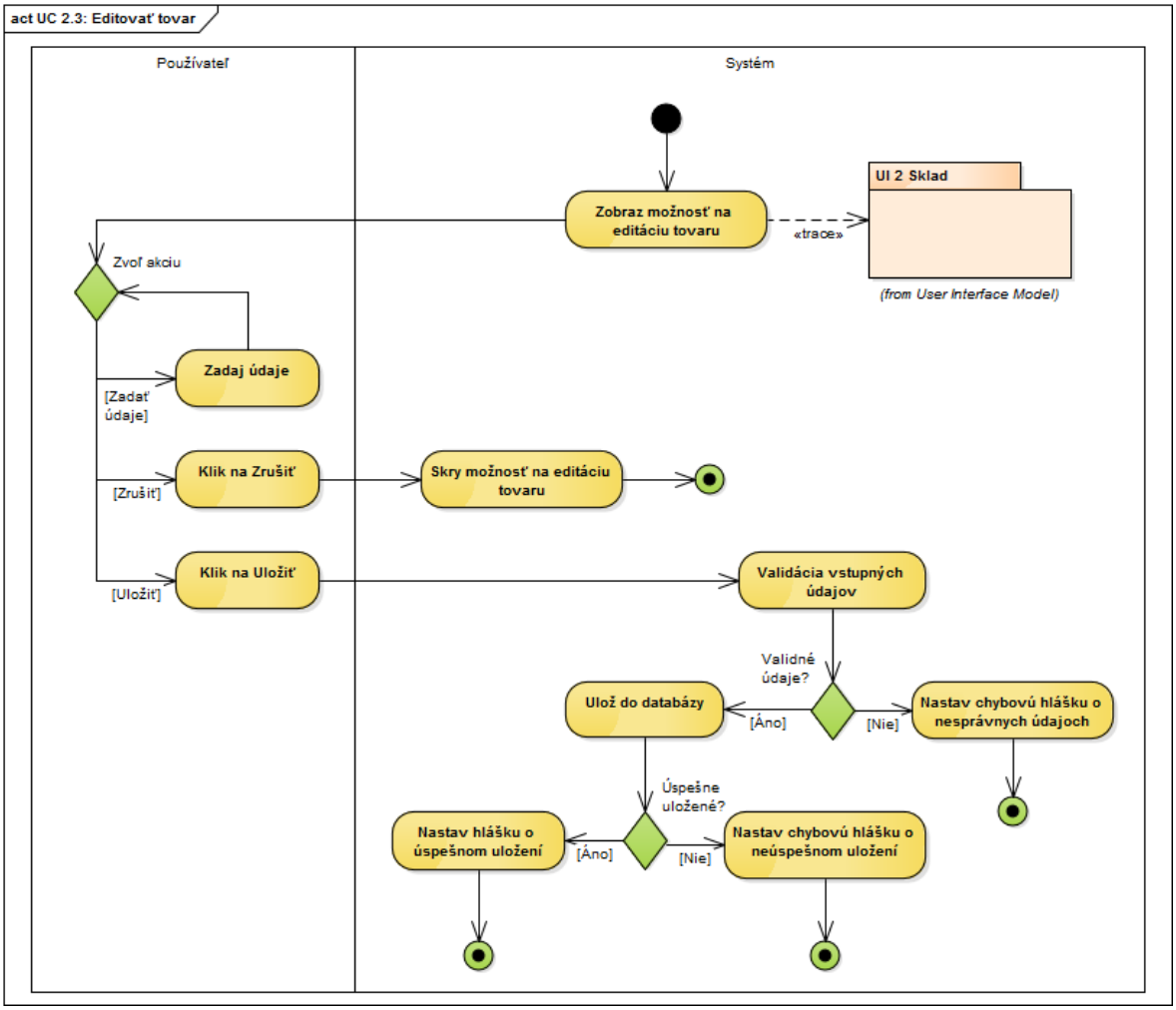
Pridávanie nového tovaru do systému opisuje prípad použitia UC 2.2, ktorého diagram aktivít je znázornený na Obrázok 8: **Activity Diagram pre Use Case 2.2 Pridať tovar**. Po zvolení možnosti na pridanie tovaru sa používateľovi zobrazí riadok v tabuľke, kde má možnosť zadať názov tovaru a na množstvo naskladňovaného tovaru. Názov tovaru musí byť v systéme unikátny. O úspešnom alebo neúspešnom uložení tovaru do systému je používateľ upozornený prostredníctvom správ, ktoré systém zobrazuje.



Obrázok 8: Activity Diagram pre Use Case 2.2 Pridať tovar

Zdroj: Vlastné spracovanie

Prípad použitia Editovať tovar popisuje spôsob na zmenu názvu tovaru, aktiváciu alebo deaktiváciu tovaru a zaznamenanie pohybu tovaru v sklade. Tento proces je znázornený na Obrázok 9: **Activity Diagram pre Use Case 2.3 Editovať tovar**. Pri zmene názvu sa overuje či je tento názov unikátny. Pri zápise pohybu tovaru používateľ môže zadať príjem, výdaj alebo odpis tovaru.



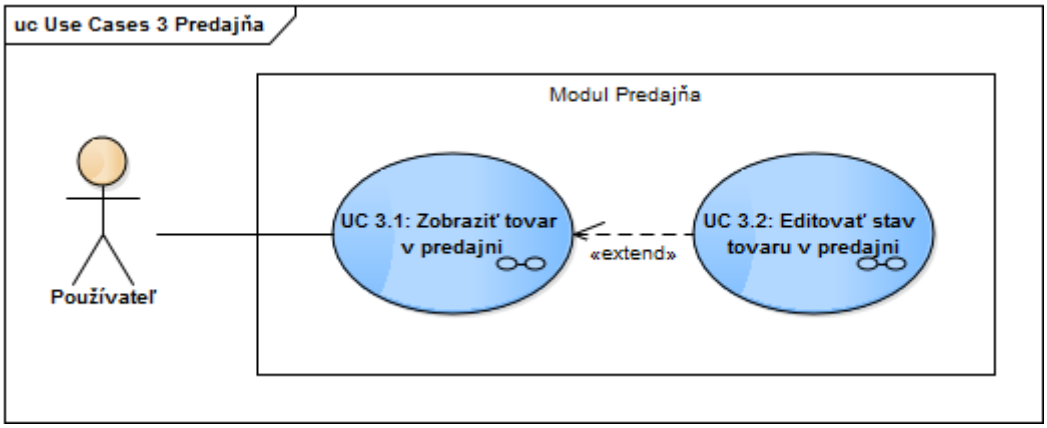
Obrázok 9: Activity Diagram pre Use Case 2.3 Editovať tovar

Zdroj: Vlastné spracovanie

Modul Predajňa

Prípady použitia v module predajňa popisujú zobrazenie stavu tovaru v predajni za zvolené časové obdobie. A tiež zaznamenanie pohybu tovaru v predajni. Všetky tieto aktivity sú zahrnuté v prípadoch použitia znázornených na Obrázok 10: **Use Cases 3 Predajňa** a v prílohe B na obrázkoch 17 a 18. V predajni sa zobrazujú len aktívne tovary.

Požiadavkou firmy je, aby bola možnosť zadávať konečný stav tovaru v predajni za daný deň. Systém následne automaticky podľa rozdielu medzi počiatočným a konečným stavom dopočíta predaj. V prípade potreby je však program možné upraviť tak, aby sa zadávala hodnota predaj a konečný stav sa dopočítal.

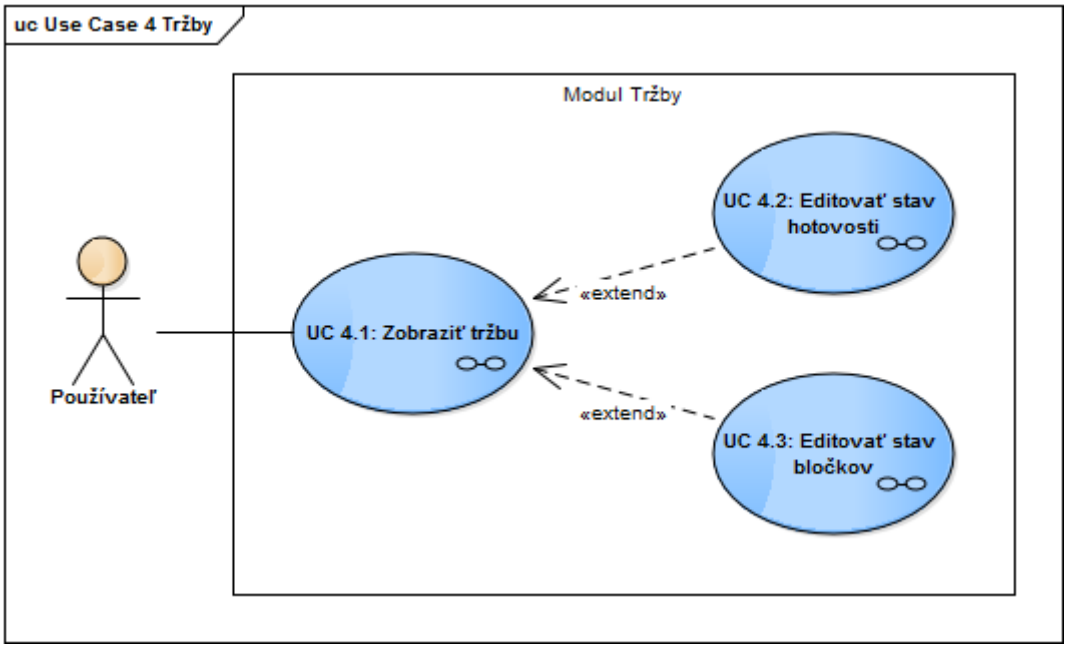


Obrázok 10: Use Cases 3 Predajňa

Zdroj: Vlastné spracovanie

Modul Tržby

Modul tržby popisuje zobrazenie a editáciu hotovosti a pokladničných bločkov a zobrazenie tržieb za zvolené časové obdobie. Prípady použitia pre modul Tržby sú znázornené na Obrázok 11: **Use Cases 5 Tržby** a diagramy aktivít pre tieto prípady použitia sú uvedené v prílohe B na obrázkoch 19 až 21.



Obrázok 11: Use Cases 5 Tržby

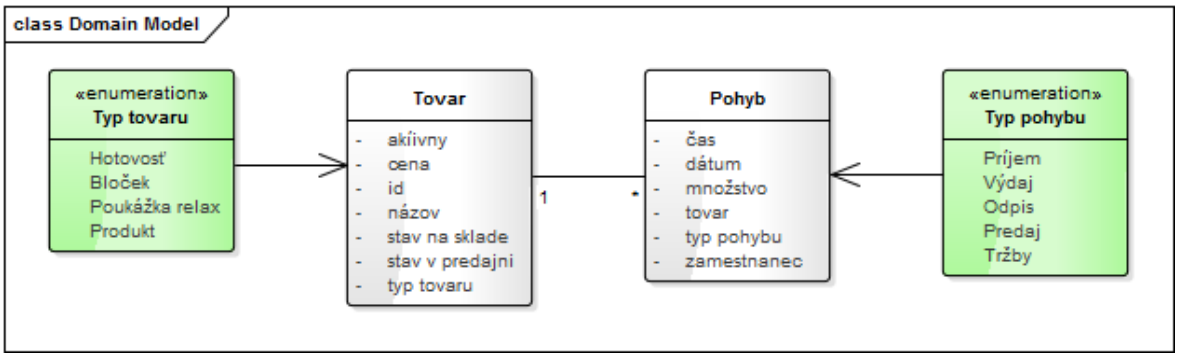
Zdroj: Vlastné spracovanie

3.2.2 Doménový model (Domain Model)

Doménový model, je formou diagramu tried (class diagram). Základnou entitou je trieda. Triedy v doménovom modely sú však značne zjednodušené, neobsahujú metódy a

majú iba dôležité atribúty. Názvy tried, atribútov a ďalších identifikátorov môžeme písať s diakritikou. Model je teda akýsi náčrt základných entít systému a vzťahov medzi nimi. Je platformovo nezávislý a atribúty nemajú dátové typy.

Navrhovaný skladový systém obsahuje dve entity Tovar a Pohyb, tak ako je to znázornené na Obrázok 12: **Doménový model**. Entita Tovar uchováva údaje o jednotlivých tovaroch ako je id, názov, cena a typ tovaru. Ďalej to, či je daný tovar aktívny a teda či sa má zobrazovať v module predajňa. A tiež aktuálny stav tovaru na sklade a v predajni. V entite Pohyb sú zaznamenané všetky pohyby tovaru, spolu s dátumom a časom kedy bol tento pohyb tovaru vykonaný a identifikáciou zamestnanca, ktorý daný záznam zapísal do systému.



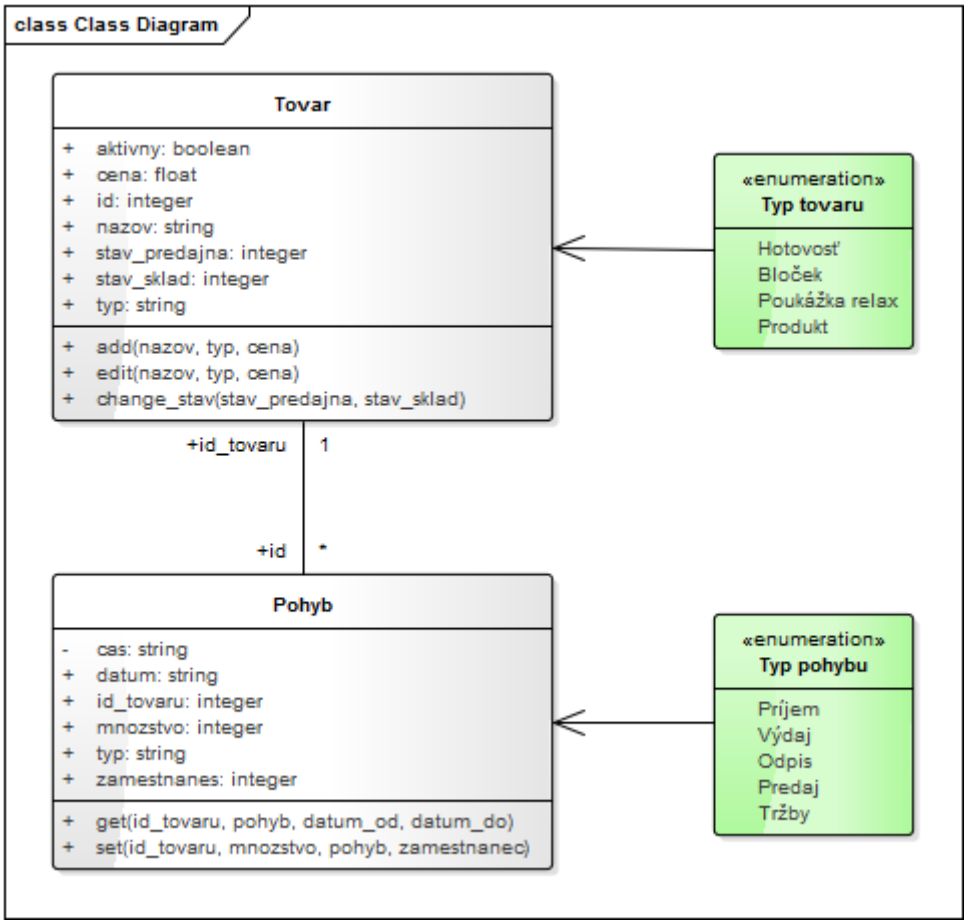
Obrázok 12: Doménový model

Zdroj: Vlastné spracovanie

3.2.3 Diagram tried (Class Diagram)

Diagram tried je diagram zobrazujúci ako bude návrh implementovaný, čo je rozdiel oproti doménovému modelu, ktorý bol skôr náčrt systému. V diagrame tried sú všetky triedy, ktoré program bude obsahovať. Triedy majú všetky atribúty a tiež metódy. Diagram je platformovo závislý, teda špecifický pre určitý programovací jazyk, v našom prípade jazyk PHP. Okrem iného to znamená, že sa v identifikátoroch už nevyskytuje diakritika, atribúty majú špecifikované dátové typy a podobne.

Rovnako ako doménový model, aj diagram tried obsahuje dve entity Tovar a Pohyb. V porovnaní s doménovým modelom tu pribudli metódy na pridanie a upravovanie tovaru v systéme. Ďalej na zaznamenanie pohybu tovaru v sklade aj v predajni. Obrázok 13: **Class Diagram** zachytáva obe entity navrhovaného skladového systému ako aj ich atribúty a metódy.



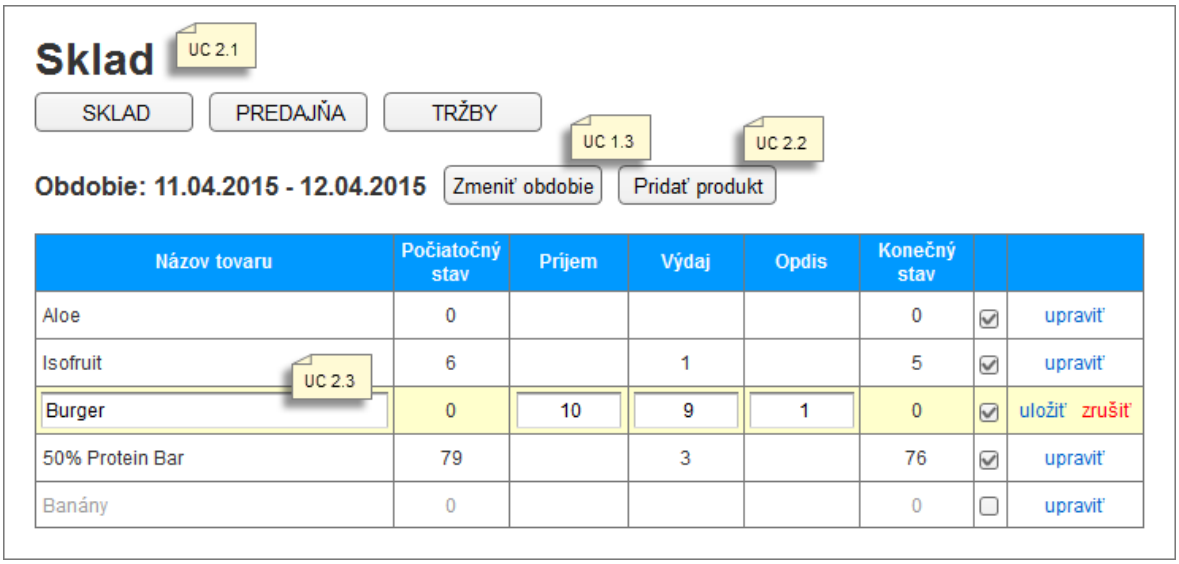
Obrázok 13: Class Diagram

Zdroj: Vlastné spracovanie

3.2.4 Návrh používateľského prostredia (User Interface Model)

Používateľské rozhranie je tá časť systému, ktorá je používateľovi systému k dispozícii na spracovanie dát. Používateľské rozhranie je časť počítačového programu, s ktorou používateľ aktívne komunikuje.

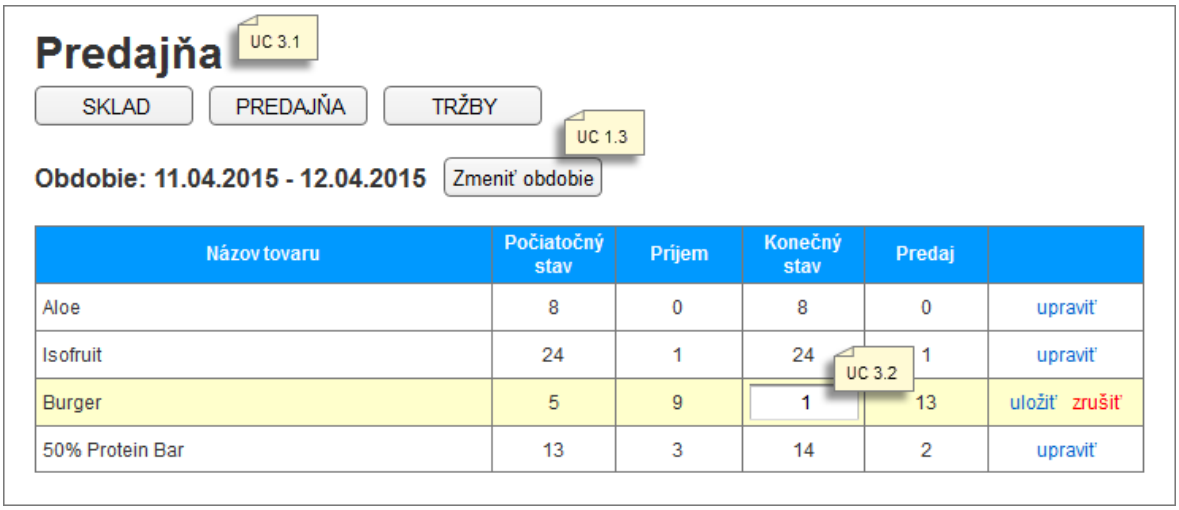
Používateľské rozhranie navarovaného skladového systému je rozdelené na základe modulov na niekoľko zobrazení ktorými sú Sklad, Predajňa a Tržby. Na Obrázok 14: **User Interface Model - modul Sklad** je znázornené používateľské rozhranie modulu Sklad s vyznačenými jednotlivými prípadmi použitia.



Obrázok 14: User Interface Model - modul Sklad

Zdroj: Vlastné spracovanie

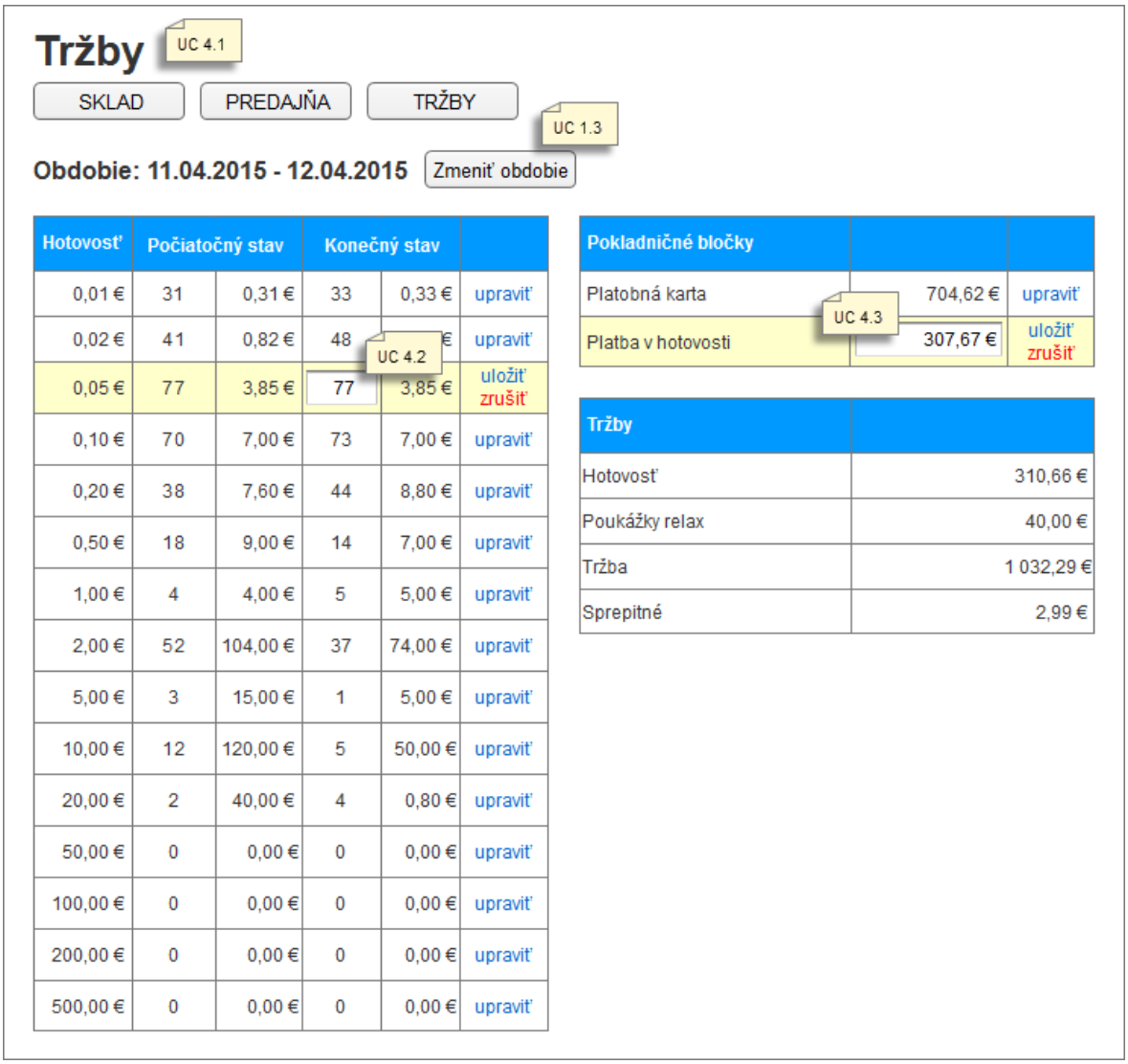
Návrh používateľského rozhrania pre predajňu je znázornený na Obrázok 15: **User Interface pre modul Predajňa.** Toto používateľské rozhranie prezeranie a editáciu tovaru v predajni. A rovnako ako vo všetkých moduloch aj zmenu časového obdobia a prechod do iné modulu.



Obrázok 15: User Interface pre modul Predajňa

Zdroj: Vlastné spracovanie

Používateľské rozhranie pre tržby obsahuje prípady použitia na zobrazenie tržieb, editáciu hotovosti a bločkov z pokladne. Na Obrázok 16: **User Interface pre modul Tržby** je zobrazený návrh používateľského rozhrania pre modul Tržby.



Obrázok 16: User Interface pre modul Tržby

Zdroj: Vlastné spracovanie

3.3 Implementácia systému

Navrhnutý databázový model je vytvorený v databázovom systéme MySQL, ktorý tvorí dátovú vrstvu webovej aplikácie. Používateľské prostredie, tvoriace vrstvu prezentačnú a dostupné používateľovi cez webový prehliadač, je naprogramované pomocou HTML a CSS a vďaka skriptovaciemu jazyku JavaScript je obohatené o dynamické prvky. Strednú vrstvu, logickú, tvorí program napísaný v jazyku PHP. Ten obsluhuje databázu a generuje používateľské prostredie.

3.3.1 Implementácia databázy v MySQL

Tvorba databázy pozostáva zo samotnej realizácie návrhu databázy v zvolenom databázovom systéme, ktorým je relačný databázový systém MySQL 5.5. Využijeme

databázu inštalovanú na webovom serveri firmy. K databáze je možné pristupovať cez administračné rozhranie phpMyAdmin.

Spustením Zdrojový kód 1: **Vytvorenie tabuľky tovar** v databázovom systéme sa vytvorí tabuľka tovar, spolu s jej atribútmi a primárnym kľúčom.

Zdrojový kód 1: Vytvorenie tabuľky tovar

```
CREATE TABLE `tovar` (
      `id` int NOT NULL AUTO_INCREMENT,
      `aktivny` boolean NOT NULL DEFAULT '1',
      `nazov` varchar(64) NOT NULL,
      `typ` enum('Produkt','Hotovosť','Bloček','Poukážka relax')
NOT NULL,
      `cena` float NULL,
      `stav_sklad` int NOT NULL DEFAULT '0',
      `stav_predajna` int  NOT NULL DEFAULT '0',
      PRIMARY KEY (`id`)
);
```

Zdroj: Vlastné spracovanie

Obdobne pomocou Zdrojový kód 2: **Vytvorenie tabuľky pohyb** vytvoríme tabuľku pohyb.

Zdrojový kód 2: Vytvorenie tabuľky pohyb

```
CREATE TABLE `pohyb` (
      `id_tovaru` int NOT NULL,
      `mnozstvo` int NOT NULL,
      `datum` datetime NOT NULL,
      `pohyb` enum('Príjem','Výdaj','Odpis','Predaj') NOT NULL,
      `zamestnanec` int NOT NULL
);
```

Zdroj: Vlastné spracovanie

Nasledujúci SQL príkaz v tabuľke pohyb cudzí kľúč pre atribút id_tovaru. Tento cudzí kľúč odkazuje na primárny kľúč v tabuľke tovar, ktorým je atribút id.

Zdrojový kód 3: Nastavenie cudzieho kľúča

```
ALTER TABLE `pohyb`
```

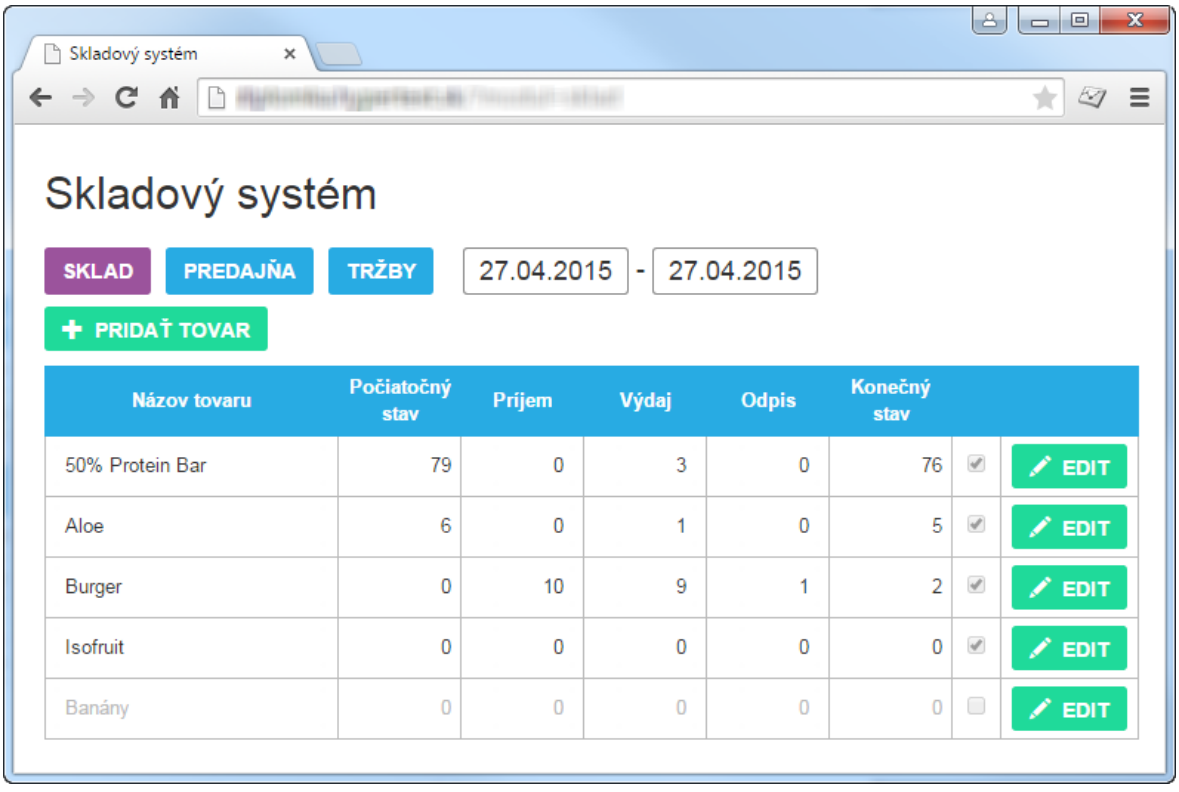
```
ADD FOREIGN KEY (id_tovaru) REFERENCES tovar(id);
```

Zdroj: Vlastné spracovanie

3.3.2 Implementácia webovej aplikácie v PHP

Webová aplikácia umožňuje napĺňanie databázy údajmi pomocou formulárov. Dôležitou funkciou aplikácie je aj prehľadné zobrazovanie údajov z databázy. Na pohodlnú navigáciu po stránke slúži menu. Všetky funkcie aplikácie sú prístupné len registrovaným a prihláseným používateľom.

Webová aplikácia je realizovaná v jazyku PHP 5 s využitím objektovo orientovaného programovania a s napojením na databázový systém MySQL 5.5. Stránky sú naprogramované v jazyku HTML 5, s použitím kaskádových štýlov CSS 3. Na niektoré dynamické prvky je využitý skriptovací jazyk JavaScript bežiaci na strane klienta (client-side) a javascriptová knižnica jQuery. Všetky formulárové vstupné polia sú ošetrené proti útokom na databázu, takzvanému MySQL injection.



Obrázok 17: Modul Sklad

Zdroj: Vlastné spracovanie

Na Obrázok 17: **Modul Sklad** je ukážka spustenej webovej aplikácie. V prílohe C na obrázkoch 24 až 27, sú zobrazené rôzne obrazovky aplikácie spustenej vo webovom prehliadači.

3.3.3 Overenie funkcionality systému

Správne fungovanie výslednej webovej aplikácie je nutné overiť pomocou testov. Testovanie softvéru je rozdelené na dve časti white-box testing a black-box testing.

Pri white-box testoch sa kontroluje vnútorná štruktúra softvéru a preto je nutné mať prístup k zdrojovým kódom aplikácie. Pomocou testovania komponentov, tzv. unit testingu, boli otestované všetky stavy a vetvy softvérového kódu vytvorenej webovej aplikácie.

Druhou metódou testovania softvéru je black-box testing, pri ktorom sa overuje funkcionalita systému z pohľadu používateľa, bez ohľadu na vnútornú štruktúru softvéru. Na overenie funkcionality celého skladové systému sú využité dve sady testovacích scenárov (test cases). Prvou sadou sú testy zamerané na správne zobrazenie používateľského rozhrania a na správne fungovanie ovládacích prvkov aplikácie v rôznych webových prehliadačoch (corss-browser testing). Druhú sadu testov tvoria testovacie scenáre zamerané na funkcionalitu a logiku skladového systému. Tieto testovacie scenáre overujú vkladanie a editáciu údajov, ich zobrazovanie a správne prepočítanie výsledných čísel.

Všetky chyby nájdené pri testovaní webovej aplikácie pre skladový systém boli odstránené a následným re-testingom boli tieto opravy, ako aj celá funkcionalita webovej aplikácie, dôkladne overená. Výsledná aplikácia je schopná uspokojovať potreby firmy, pre ktorú bola navrhnutá.

3.4 Prevádzkovanie skladového systému a jeho ďalší rozvoj

V prvotnej fáze prevádzky navrhnutej webovej aplikácie je potrebné sledovať jej chod a odhaľovať prípadne chyby a nedostatky ktoré sa vyskytnú. Pri dlhodobom používaní sa databáza postupne naplní údajmi, čo umožní ľahšie nájsť chyby, ktoré by sa mohli v určitých špecifických prípadoch vyskytnúť. Väčšie množstvo reálnych prevádzkových údajov taktiež poskytne lepší pohľad na zobrazované čísla, najmä na údaje za rôzne časové obdobia.

S rastom informačných potrieb firmy Starfit, je pravdepodobné, že aplikácia sa v budúcnosti bude rozširovať. Tieto zmeny si síce budú vyžadovať zásah do zdrojového kódu aplikácie, ale vďaka modulovému riešeniu a objektovo orientovanému programovaniu, je úprava a rozšírenie vytvorenej webovej aplikácie pomerne jednoduché.

Jedným z možných rozšírení aplikácie je zobrazovanie číselných údajov vo forme grafov. Takéto vizuálne zobrazenie poskytuje lepší prehľad a tým aj interpretáciu dát, ktoré sa počas používania skladového systému nazbierajú. Ďalej je možné pridať funkcionalitu na porovnanie údajov za rôzne časové obdobia a tak vidieť napríklad medziročný rast tržieb, alebo ich kolísanie počas jednotlivých mesiacov roka.

Inou skupinou rozšírení webovej aplikácie je napojenie na existujúce informačné systémy vo firme. Napojením na kasu by bolo možné získavať údaje o platbách v hotovosti a kartou automaticky a tým by odpadla nutnosť tieto údaje zadávať každý deň ručne. Zoznam tovaru a stav tovaru v sklade a v predajni by bolo možné prepojiť s účtovným alebo skladovým softvérom.

Taktiež je možné dorobiť automatické notifikácie keď nastane určitý stav. Napríklad keď klesne množstvo tovaru v sklade pod určitú hranicu, zodpovednému zamestnancovi firmy príde email aby tento stav doplnil.

Možností na ďalší rozvoj vytvorenej webovej aplikácie pre skladové hospodárstvo je veľa a po nejakom čase reálneho používania sa ukáže, ktoré z nich sú užitočné a vhodné na doprogramovanie.

Záver

Hlavným cieľom práce je navrhnúť a vytvoriť webovú aplikáciu skladového hospodárstva, na základe špecifických požiadaviek firmy Starfit. Výsledkom práce je databáza a webová aplikácia pre skladový systém, ktorej zámerom je zefektívniť prácu pri dennej evidencii tovaru na sklade a v predajni a sprístupniť reporty šéfovi prípadne určitým zamestnancom cez Internet.

Cieľom teoretickej časti diplomovej práce je vymedzenie teoretických poznatkov z oblasti objektovo orientovaného programovania a jeho využitia pri vývoji webovej aplikácie. Sú v nej popísané základné pojmy a princípy OOP. A predstavený programovací jazyk PHP, jeho základnú charakteristiku, históriu a vývoj, ako aj výhody a nevýhody tohto jazyka. Ďalej sa v teoretickej časti venujeme relačnému databázovému modelu a databázovému systému MySQL. Uvádzame jeho výhody aj nevýhody.

V teoretickej časti sú ešte charakterizovaný a popísaný navrhovaný skladový systém a popis súčasného stavu riešenia skladovej evidencie vo firme Starfit. Uvedený je stručný opis existujúcich programov, ktoré riešia danú problematiku, ako aj ich nedostatky a nevýhody z hľadiska potrieb firmy Starfit. Taktiež je načrtnutá užitočnosť nového skladového systému pre firmu.

Cieľom praktickej časti je analýza požiadaviek navrhovaného skladového systému a tvorba modelov. Na konci je pomocou objektovo orientovaného jazyka PHP 5 vytvorená webovú aplikácia, ktorá obsluhuje vytvorenú databázu. Vytvorená webová aplikácia je schopná uspokojovať požadované potreby používateľov firmy Starfit, avšak pri implementácii sa ukázali mnohé možnosti vylepšenia a ďalšieho rozvoje tejto webovej aplikácie.

Pri spracovaná tejto témy sme si rozšírili znalosti o webových aplikáciách a databázových systémoch a hlavne o ich návrhu a tvorbe. Taktiež sme si zlepšili naše programátorské schopnosti a osvojili sme si princípy objektovo orientovaného programovania.

Zoznam použitej literatúry

[1] HOLMEVIK, J. R. 1994. *Compiling SIMULA: A Historical Study of Technological Genesis. IEEE Annals of the History of Computing, Vol. 16, No. 4, p. 25-37*. Dostupné na Internete: http://www.idi.ntnu.no/grupper/su/publ/simula/holmevik-simula-ieeeannals94.pdf

[2] KOFLER, M. - ÖGGL, B. 2007. *PHP 5 a MySQL 5 : Průvodve webového programátora*. Prvé vydanie. Brno : Computer Press, a.s., 2007. 608 s. ISBN 978-80-251-1813-9

[3] Wikipedia, the free encyclopedia, 2015, *Object-oriented programming*, [online], [2015-03-25] Dostupné na internete: https://en.wikipedia.org/wiki/Object-oriented_programming#History

[4] PHP.net, *History of PHP*, [online], [2015-03-25] Dostupné na internete: http://www.php.net/manual/en/history.php.php

[5] W3Techs, *Usage of server-side programming languages for websites*, [online], [2015-03-25] Dostupné na internete: http://w3techs.com/technologies/overview/programming_language/all

[6] Wikipedia, the free encyclopedia, 2015, *PHP*, [online], [2015-03-25] Dostupné na internete: http://cs.wikipedia.org/wiki/PHP

[7] Wikipedia, the free encyclopedia, 2015, *Objektově orientované programování*, [online], [2015-03-25] Dostupné na internete: http://cs.wikipedia.org/wiki/Objektově_orientované_programování

[8] Živě.cz, 2005, *Objekty v PHP 5 - 5. díl*, [online], [2015-03-25] Dostupné na internete: http://www.zive.cz/clanky/objekty-v-php--5---5-dil/sc-3-a-123376/default.aspx

[9] Wikipedia, the free encyclopedia, 2015, *Database*, [online], [2015-03-25] Dostupné na internete: http://en.wikipedia.org/wiki/Database

[10] Wikipedia, the free encyclopedia, 2015, *Relační databáze*, [online], [2015-03-25] Dostupné na internete: http://cs.wikipedia.org/wiki/Relační_databáze

[11] Wikipedia, the free encyclopedia, 2015, *MySQL*, [online], [2015-03-25] Dostupné na internete: https://sk.wikipedia.org/wiki/MySQL

[12] Wikipedia, the free encyclopedia, 2015, *MySQL*, [online], [2015-03-25] Dostupné na internete: https://cs.wikipedia.org/wiki/MySQL

[13] Wikipedia, the free encyclopedia, 2015, *Data manipulation language*, [online], [2015-03-25] Dostupné na internete: http://en.wikipedia.org/wiki/Data_Manipulation_Language

[14] KULTÁN, J. 2012. *Databázové systémy : Vybrané kapitoly (Učebnica pre denných a externých študentov netechnických vysokých škôl).* Prvé vydanie. Bratislava : Vydavateľstvo EKONÓM, 2012. 128 s. ISBN: 978-80-225-3350-8

[15] PHP.net, *MySQL Improved Extension*, [online], [2015-03-25] Dostupné na internete: http://www.php.net/manual/en/book.mysqli.php

[16] JACOBSEN, I. - CHRISTERSON, M. - JONSSON, P. - OVERGAARD, G. 1992. *Object Oriented Software Engineering : A Use Case Driven Approach*. Prvé vydanie. Addison-Wesley ACM Press, 1992. ISBN: 0-201-54435-0

[17] ASP.net, *Get Started with ASP.NET Web Sites*, [online], [2015-04-06] Dostupné na internete: http://www.asp.net/get-started/websites

[18] BURD, B. - KISZKA, B. 2003. *JSP : JavaServer Pages : Podrobný průvodve*. Prvé vydanie. Praha : Computer Press, a.s., 2003. 381 s. ISBN 80-7226-804-X

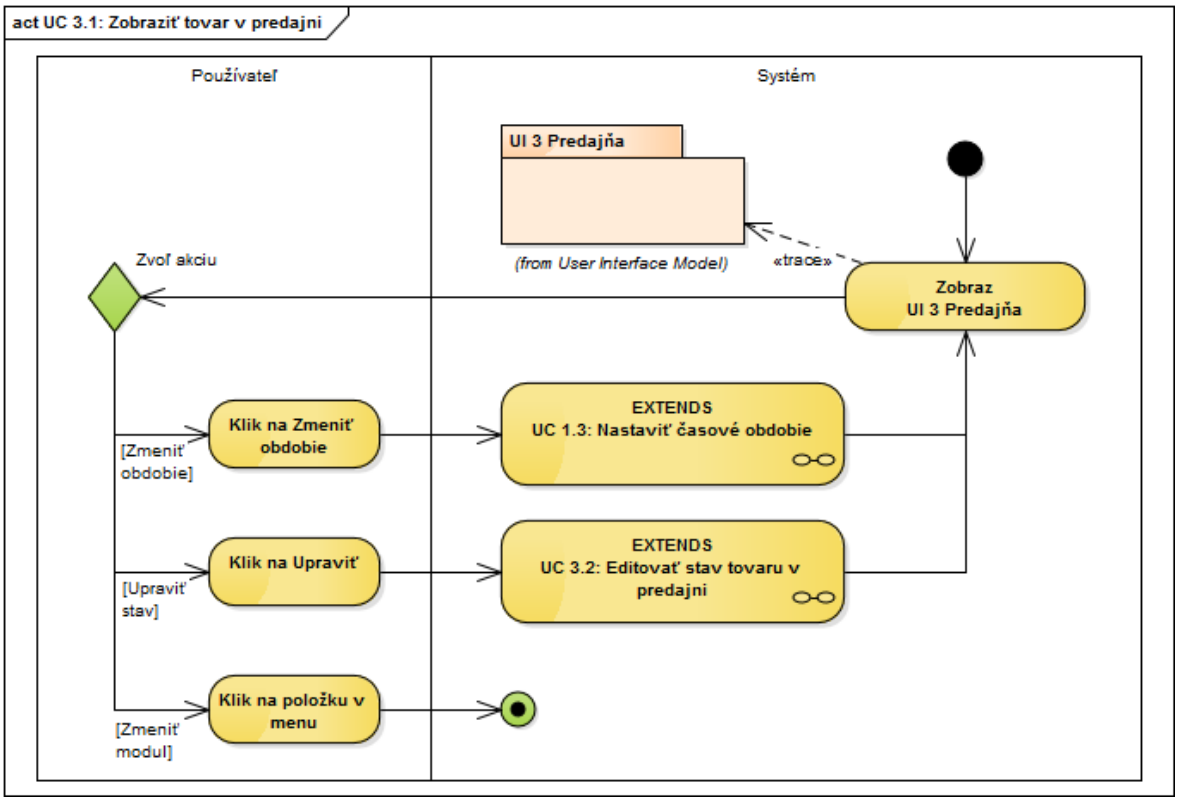
[19] Oracle.com, *The Java EE 6 Turorial : Chapter 15 : Java Servlet Technology*, [online], [2015-04-06] Dostupné na internete: http://docs.oracle.com/javaee/6/tutorial/doc/bnafd.html

Zoznam príloh

Príloha A: SQL kód na tvorbu entít

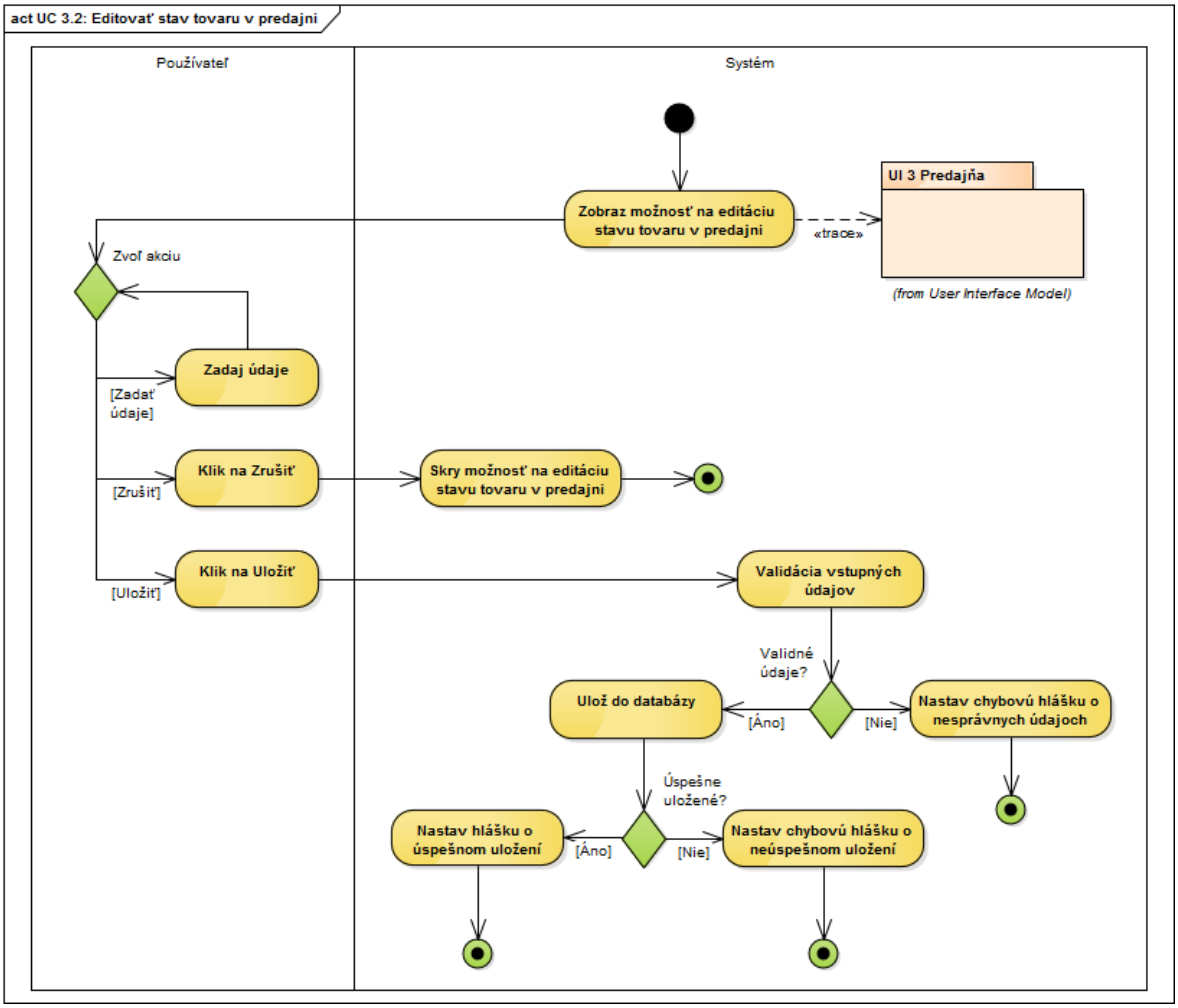
```
CREATE TABLE `tovar` (
      `id` int NOT NULL AUTO_INCREMENT,
      `aktivny` boolean NOT NULL DEFAULT '1',
      `nazov` varchar(64) NOT NULL,
      `typ` enum('Produkt','Hotovosť','Bloček','Poukážka relax')
NOT NULL,
      `cena` float NULL,
      `stav_sklad` int NOT NULL DEFAULT '0',
      `stav_predajna` int  NOT NULL DEFAULT '0',
      PRIMARY KEY (`id`)
);

CREATE TABLE `pohyb` (
      `id_tovaru` int NOT NULL,
      `mnozstvo` int NOT NULL,
      `datum` datetime NOT NULL,
      `pohyb` enum('Príjem','Výdaj','Odpis','Predaj') NOT NULL,
      `zamestnanec` int NOT NULL
);

ALTER TABLE `pohyb`
ADD FOREIGN KEY (id_tovaru) REFERENCES tovar(id);
```

Príloha B: Aktivity diagramy



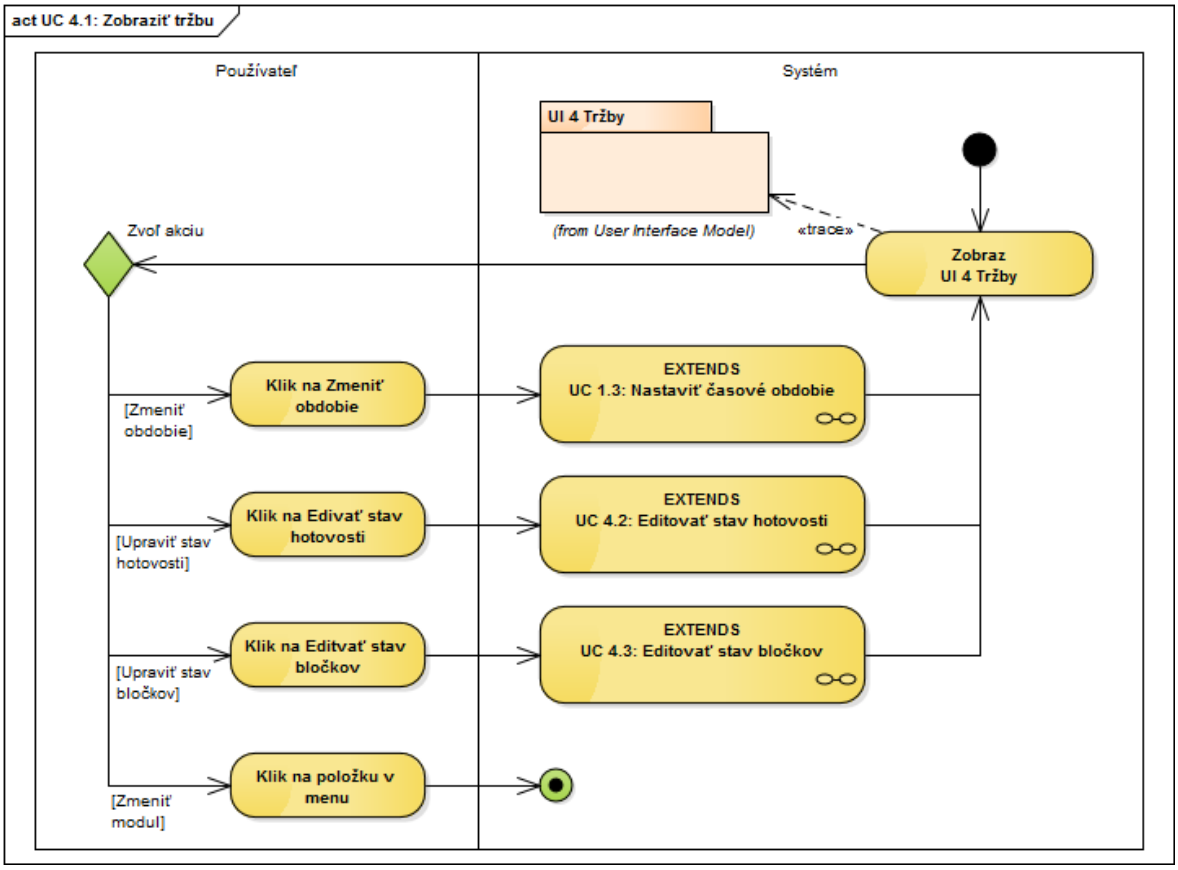
Obrázok 18: Activity Diagram pre Use Case 3.1 Zobraziť tovar v predajni

Zdroj: Vlastné spracovanie



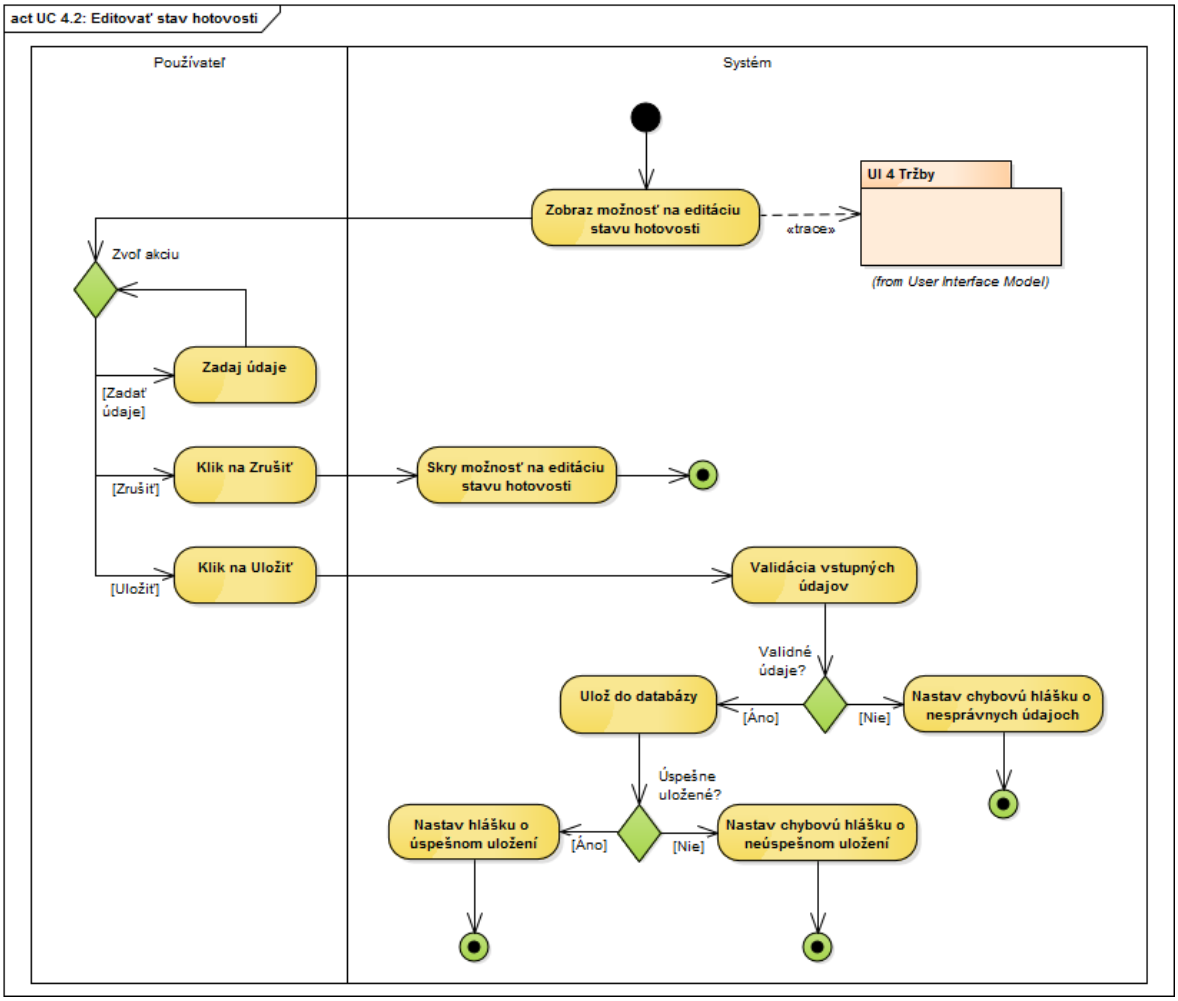
Obrázok 19: Activity Diagram pre Use Case 3.2 Editovať stav tovaru v predajni

Zdroj: Vlastné spracovanie



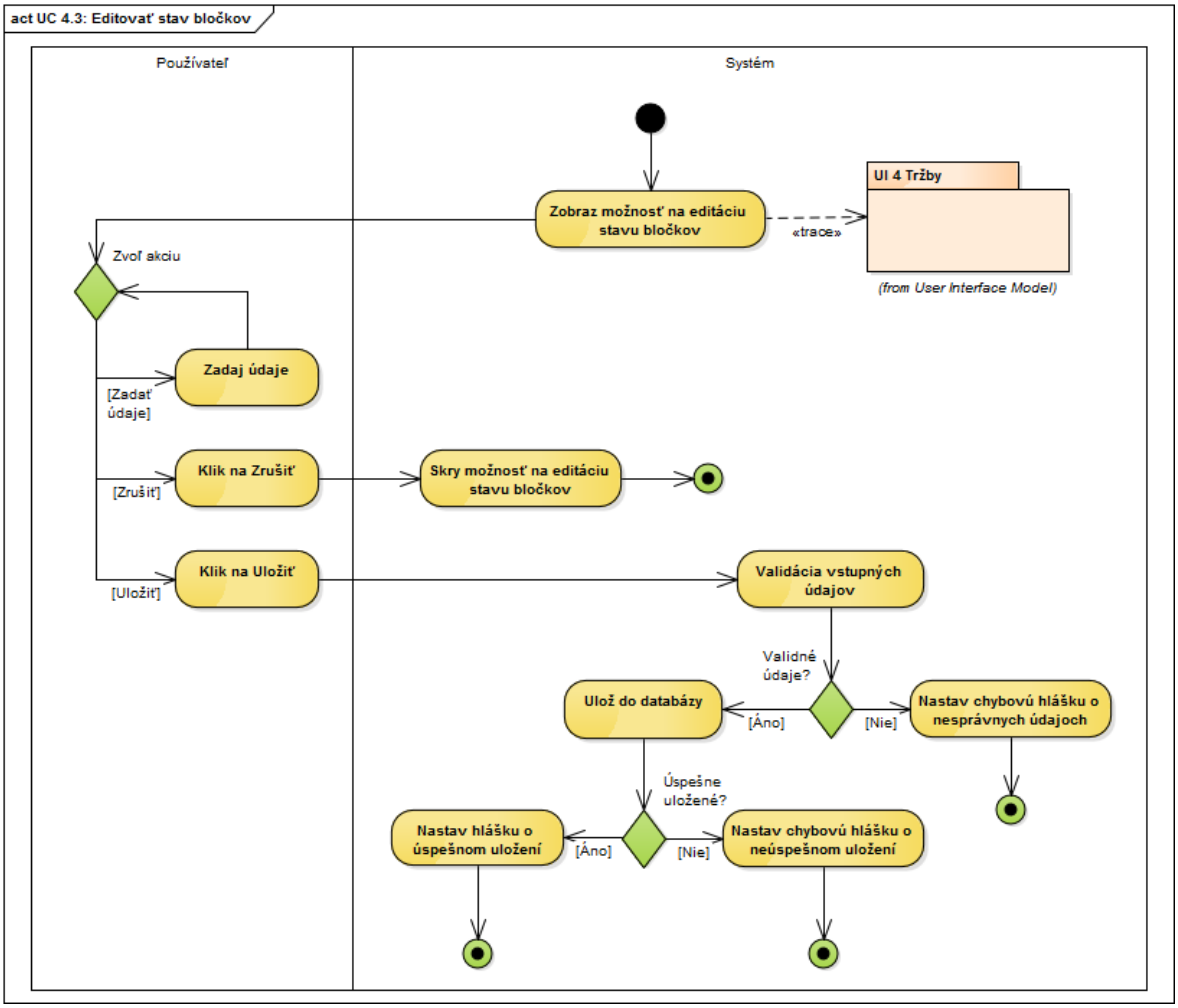
Obrázok 20: Activity Diagram pre Use Case 4.1 Zobraziť tržbu

Zdroj: Vlastné spracovanie



Obrázok 21: Activity Diagram pre Use Case 4.2 Editovať stav hotovosti

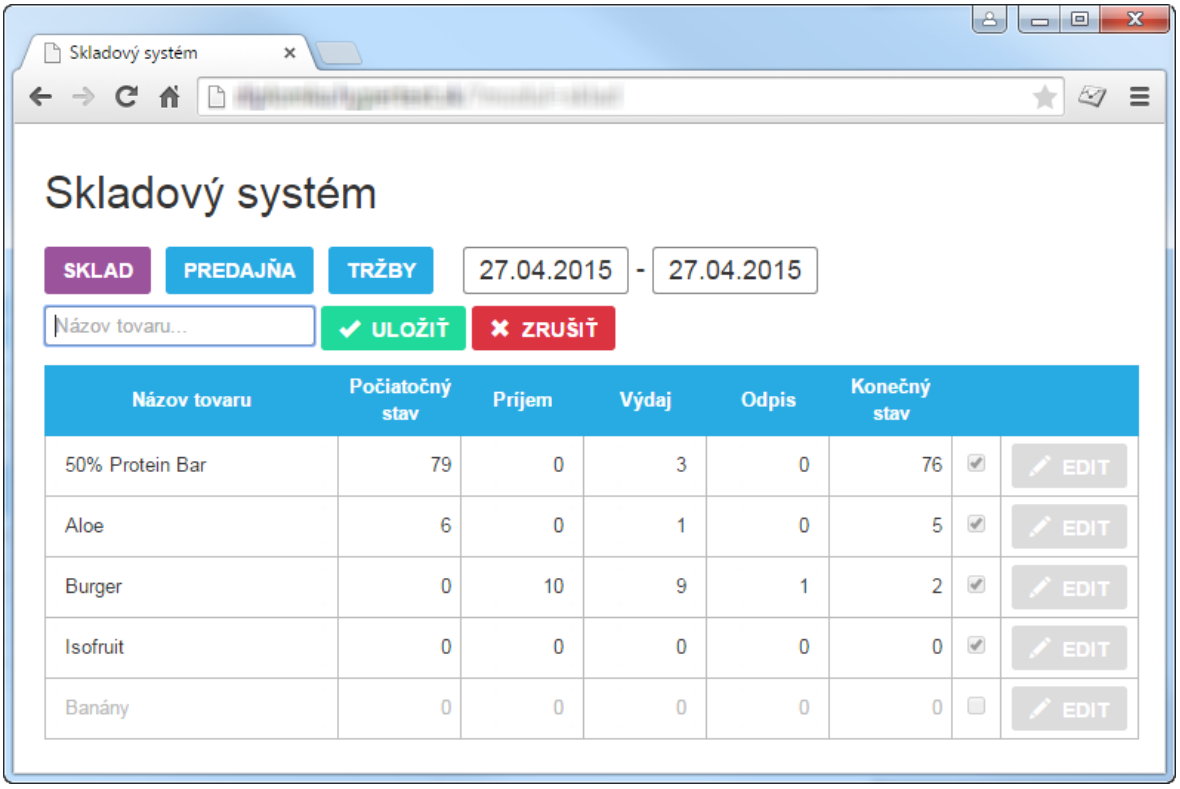
Zdroj: Vlastné spracovanie



Obrázok 22: Activity Diagram pre Use Case 4.3 Editovať stav bločkov

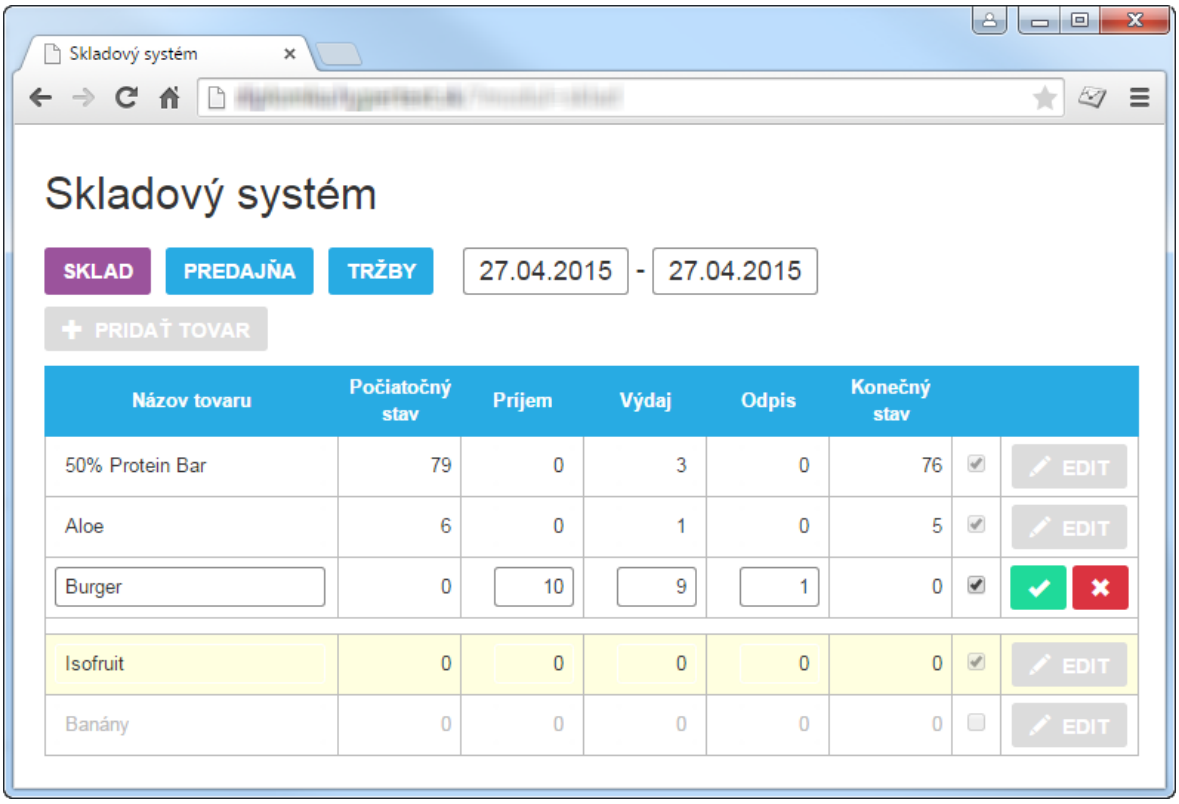
Zdroj: Vlastné spracovanie

Príloha C: Ukážky webovej aplikácie



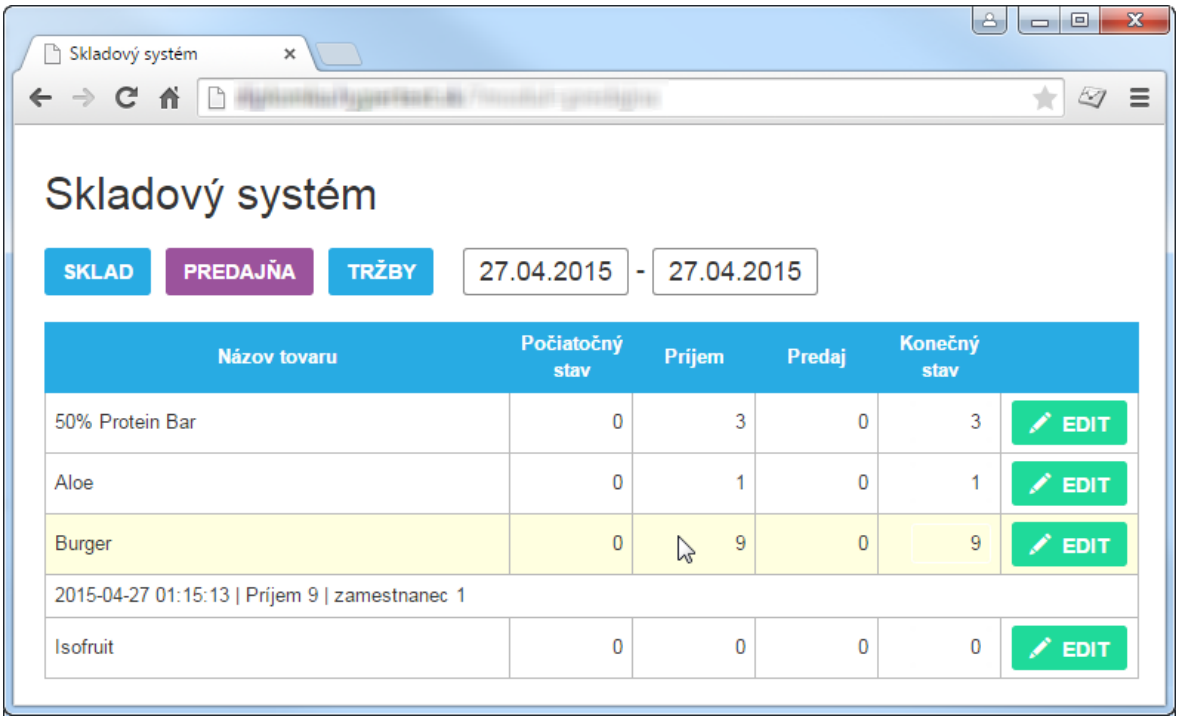
Obrázok 23: Modul Sklad v móde na pridanie tovaru

Zdroj: Vlastné spracovanie



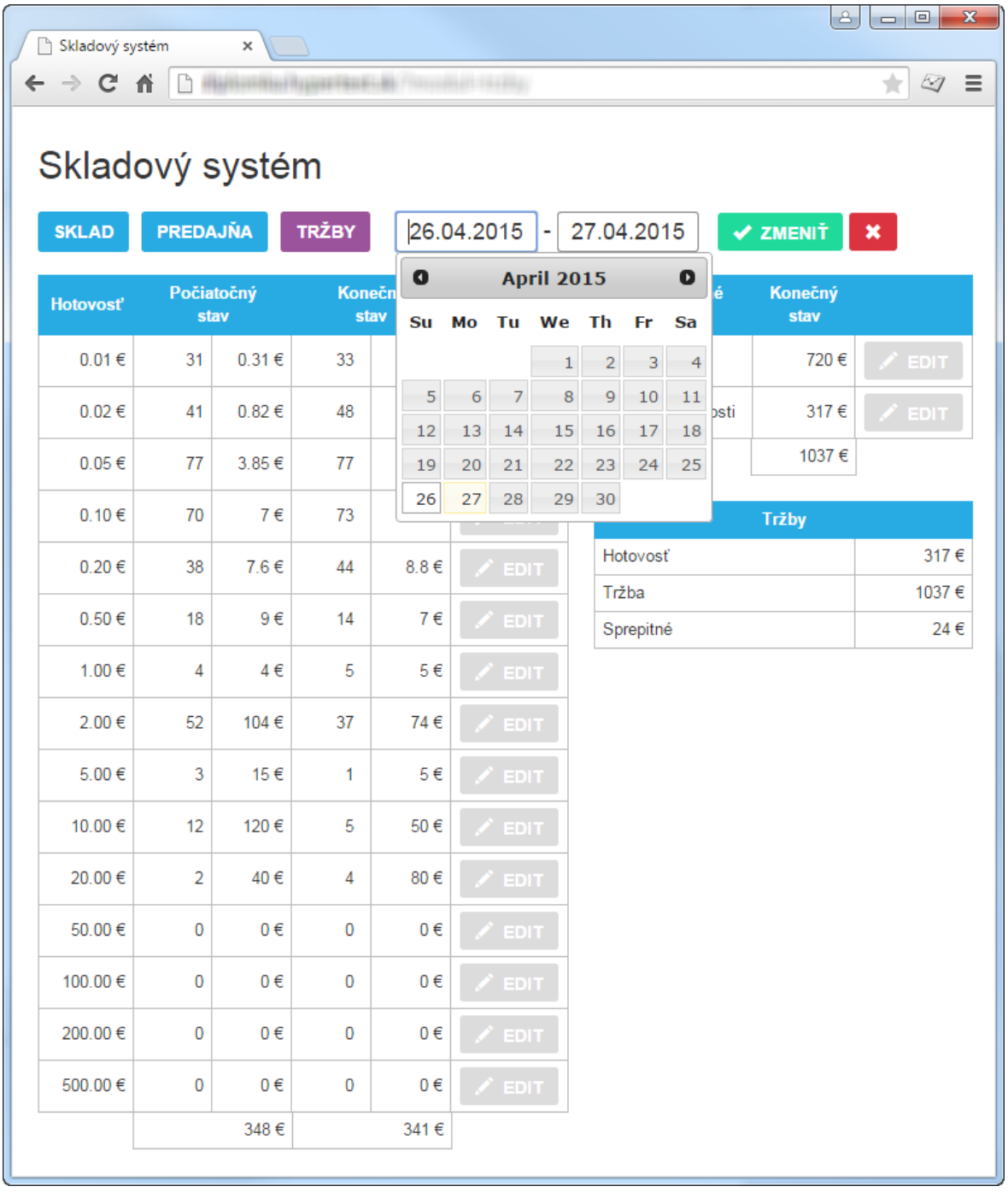
Obrázok 24: Modul Sklad v editačnom móde

Zdroj: Vlastné spracovanie



Obrázok 25: Modul Predajňa

Zdroj: Vlastné spracovanie



Obrázok 26: Modul Tržby, zmena dátumu

Zdroj: Vlastné spracovanie

Príloha D: Priložené CD

1. Model v programe Enterprise Architect
2. Zdrojový kód webovej aplikácie