Mendelova univerzita v Brně
Provozně ekonomická fakulta

Softwarová podpora pro řízený sklad

Bakalářská práce

Vedoucí práce:
RNDr. Zuzana Prišťáková, Ph.D.

Dušan Neckař

Brno 2017

Provozně ekonomická fakulta

Ústav informatiky
Akademický rok: 2016/2017

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Zpracovatel : **Dušan Neckař**

Studijní program: Inženýrská informatika

Obor: Automatizace řízení a informatika

Název tématu: **Softwarová podpora pro řízený sklad**

Rozsah práce: 1,5 – 3 AA

Zásady pro vypracování:

1. Nastudujte problematiku zavedení informačních technologií a systémů pro oblast řízeného skladu.
2. Analyzujte současný stav řešení ve firmě se zavedeným řízeným skladem. Definujte vnitřní a vnější procesy s dopadem řízený sklad.
3. Určete funkční a nefunkční požadavky pro daný systém.
4. Navrhněte řešení požadavků prostřednictvím využití jazyka UML.
5. Implementujte navržený model a proveďte otestování.
6. Analyzujte implementované řešení z pohledu rozšíření.

Mendelova univerzita v Brně

Seznam odborné literatury:

1. BASL, J. *Inovace podnikových informačních systémů: podpora konkurenceschopnosti podniků.* Praha: Professional Publishing, 2011. 150 s. ISBN 978-80-7431-045-4.
2. NAGEL, C. – GLYNN, J. – EVJEN, B. *C# 2008 Programujeme profesionálně.* Praha: Computer Press, 2009. 1904 s. ISBN 978-80-251-2401-7.
3. ŘEPA, V. *Podnikové procesy: procesní řízení a modelování.* 2. vyd. Praha: Grada, 2007. 281 s. ISBN 978-80-247-2252-8.

Datum zadání bakalářské práce: říjen 2016

Termín odevzdání bakalářské práce: květen 2017

L. S.

Dušan Neckař
Autor práce

RNDr. Zuzana Priščáková, Ph.D.
Vedoucí práce

Ing. Petr Jedlička, Ph.D.
Vedoucí ústavu

doc. Ing. Arnošt Motyčka, CSc.
Děkan PEF MENDELU

Na tomto místě bych rád poděkoval mé vedoucí RNDr. Zuzaně Priščákové, Ph.D., za její ochotný přístup a poskytnutí cenných rad. Dále bych rád poděkoval společnosti DOBEŠ-STAVBY s. r. o. za vstřícnost a věcné připomínky. V poslední řadě děkuji rodině, která mě při studiu podporovala a motivovala.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Softwarová podpora pro řízený sklad** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 12. května 2017 .....

Abstract

Neckař, D. Software support for warehouse management. Bachelor thesis. Brno, 2017.

The bachelor thesis deals with design and implementation of modules for warehouse management through web applications. The theoretical part introduces the information systems and the principles of their development, and it presents the technologies used during their implementation. The practical part describes the client requirements which are used for the analysis and design of the module. This is then implemented and tested.

Keywords: information system, warehouse management, web application, ASP .NET, C#, MVC

Abstrakt

Neckař, D. Softwarová podpora pro řízený sklad. Bakalářská práce. Brno, 2017.

Bakalářská práce se zabývá návrhem a realizací modulu pro řízený sklad formou webové aplikace. V teoretické části je představena problematika informačních systémů, principů vývoje a jsou prezentovány technologie využité při implementaci. V praktické části jsou popsány klientské požadavky, na základě kterých je provedena analýza a vyhotoven návrh. Dále je tento modul implementován a otestován.

Klíčová slova: informační systém, řízený sklad, webová aplikace, ASP .NET, C#, MVC

Obsah

Seznam obrázků

1 Úvod

V dnešní době je k dispozici nepřeberné množství dat, které je potřeba třídit na data užitečná a neužitečná, jež jsou ukládána v datových úložištích. Z užitečných dat jsou pomocí různých třídících metod a filtrů získávány informace, které jsou podstatné pro chod celého podniku. Tyto procesy se provádí v celku zvaném informační systém.

V dřívějších dobách byl informační systém aplikován podnikem za účelem konkurenční výhody. Naopak dnes je to nezanedbatelná součást podniku, která je využívána pro podporu, ulehčení práce a šetření času zaměstnanců.

Informační systémy určené pro skladové hospodářství jsou využívané pro snadnější přehled skladů, jejich aktuálně naskladněných zásob s případnými budoucími dodávkami dalších produktů či informace o vývozech.

Na trhu je již velké množství softwarových řešení, která jsou nabízena specializovanými vývojářskými společnostmi. Tyto aplikace jsou ve většině případů nabízeny ke koupi, ovšem jsou k dispozici i volně dostupné, případně demo verze poskytované na určité časové období.

Tento systém je využíván i stavební firmou DOBEŠ-STAVBY s. r. o., ale v dnešní době je již bohužel nedostatečný, a proto je cílem vytvořit systém nový.

2 Cíl práce

Cílem této bakalářské práce je tvorba modulu pro řízený sklad. Práce se soustředí na návrh a modelování na základě získaných požadavků a analýzy současného informačního systému. Následně bude provedena implementace, jejíž součástí bude i otestování výsledné aplikace. V závěru práce budou navrhnuta možná budoucí rozšíření pro vytvořený modul řízeného skladu.

3 Informační systémy

Informační systém je definován jako skupina komponent určených pro sběr, ukládání a zpracování dat, která v podniku vedou k poskytnutí požadovaných informací a znalostí. Podniky a organizace spoléhají na informační systémy, které se starají o zkvalitnění prováděných operací a řízení celého podniku (Gála, 2015, s. 13–15).

3.1 Oblast řízeného skladu

Většina firem využívá sklad pro firemní procesy, jimiž jsou naskladnění materiálů sloužících pro výrobu či zhotovených výrobků určených k prodeji. Pokud firma nemá přehled o aktuálním stavu skladu a jeho obsahu, není možné efektivně řídit firemní procesy. Tímto se může stát, že sklad bude příliš naplněn, případně nebude k dispozici potřebné množství zboží pro výrobu či jiné manipulace.

Kvůli eliminaci těchto negativních následků firmy zavádí informační systémy určené speciálně pro oblast skladu, jež jsou nazývány řízené sklady. Tyto systémy firmám pomáhají při organizaci skladu, zdokonalení jeho provozu a optimalizování skladových zásob (Košut, 2016).

3.2 Architektura IS/ICT

Architektura zahrnuje grafické a písemné vyjádření celého řešení informačního systému, provázanost s dalšími komponentami spadajícími pod vyvíjený informační systém a taktéž se stará o udání směru vývoje. Je nepostradatelným komunikačním prostředkem mezi vedením podniku a vývojovým týmem, do kterého se řadí analytici, projektanti, programátoři a designéři. Při dobře navržené architektuře pro konkrétní podnik se lépe organizují nastalé změny, celkové řízení a kontrola vývoje (Sommerville, 2011, s. 147–150).

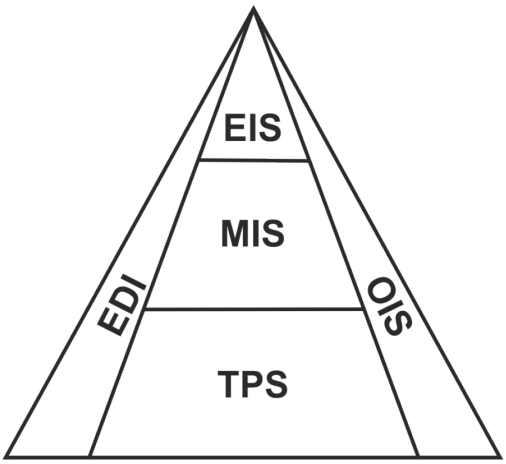
Existují dva pohledy na architekturu, kterými jsou globální pohled zahrnující globální architekturu spolu s detailními dílčími architekturami a moderní pohled, do kterého se řadí architektura 4+1 pohledů, SOA (Service Oriented Architecture), MDA (Model Driven Architecture) a MVC (Model-View-Controller) (Rábová, 2008, s. 26–29).

3.2.1 Globální architektura

Globální architektura je přibližný návrh informačního systému, který znázorňuje jednotlivé bloky, což jsou funkce, služby a procesy se vzájemnými vztahy mezi nimi. Skládá se z několika částí, které mohou být rozdělené podle dimenzí na vertikální a horizontální.

Vertikální dimenze zobrazuje logické uspořádání podniku podle rozdělení managementu na operativní, taktické a strategické řízení. Druhým pohledem je horizontální dimenze, která zobrazuje členění podniku podle podnikových oblastí na management, výrobu, sklad, účetnictví a další (Bruckner, 2012, s. 250).

Jak je zobrazno na obrázku 1, globální architektura se skládá z několika úrovní a bloků. Základní vrstvou globální architektury je TPS, celým názvem Transaction Processing System, který se stará o získávání a shromažďování všech informací a obchodních transakčních dat. Úlohou této vrstvy je tedy shromáždit informace a poskytnout je dalším vrstvám pro budoucí analýzy (Sodomka, 2008, s. 74–75).



Obrázek 1: Globální architektura IS (Bruckner, 2012, s. 259)

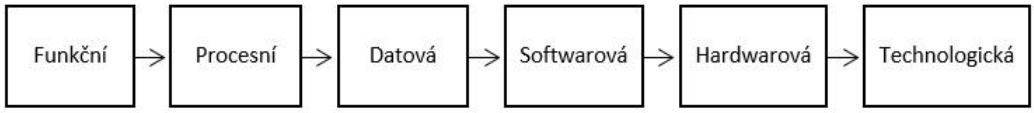
Další vrstvou je Management Information System, označena jako MIS, která je určena pro taktické řízení podniku. Tato vrstva se dělí na tři úrovně, do kterých spadají podobné podnikové procesy. Jsou jimi obchodně–logistická úroveň, která se stará o prodej, nákup a skladování, finančně–účetní úroveň, která zahrnuje účetnictví, mzdy a majetek, a jako poslední je úroveň průřezová, kam patří personalistika, legislativa a celková organizace. Z MIS často vychází blok Decision Support System (DSS), který je určen pro operativní a taktické plánování v grafické podobě.

Nejvyšším blokem informační pyramidy je blok EIS, celým názvem Executive Information System, který je určen pro strategické řízení. Tento blok pracuje nad daty získanými z nižších vrstev, ze kterých jsou vytvářeny grafické prognózy v závislosti na časovém úseku. EIS využívá specializované programy, které využívají agregovaná data pro ukládání do datových skladů.

Posledními bloky globální architektury jsou systémy EDI a OIS, které pracují na všech vrstvách vertikální dimenze. Electronic Data Interchange (EDI) je určena pro komunikaci mezi informačními systémy patřícími obchodním partnerům za účelem elektronické výměny strukturovaných dat, čímž vytváří hlavní nástroj umožňující elektronické obchodování.

Office Information System (OIS) je blok určený pro podporu administrace v systému za účelem zvýšení produktivity a ulehčení práce zaměstnanců. Cílem OIS je tvořit a upravovat písemná a grafická administrativní data.

Součástí globální architektury jsou architektury dílčí, které navazují na již vyhotovený hrubý nástin a detailněji popisují informační systém. Existuje šest typů dílčích architektur, které na sebe logicky navazují viz obrázek 2 (Bruckner, 2012, s. 258–263).



Obrázek 2: Dílčí architektury (Rábová, 2008, s. 21–23)

První dílčí architekturou je architektura funkční, která vychází z globální architektury a namodelované funkce rozděluje na menší části pomocí hierarchického rozkladu. Tuto rozšiřuje architektura procesní, která znázorňuje základní pohled na okolí systému a související výměnu dat s externími entitami.

Po vytvoření Data Flow Diagramu (DFD), který přísluší právě procesní architektuře, přichází na řadu Entity Relationship Diagram (ERD), což je návrh datové základny uvnitř databáze, jenž přísluší datové architektuře. Dalšími architekturami jsou softwarová a hardwarová architektura, které řeší programové a hardwarové komponenty a vzájemné vztahy mezi nimi.

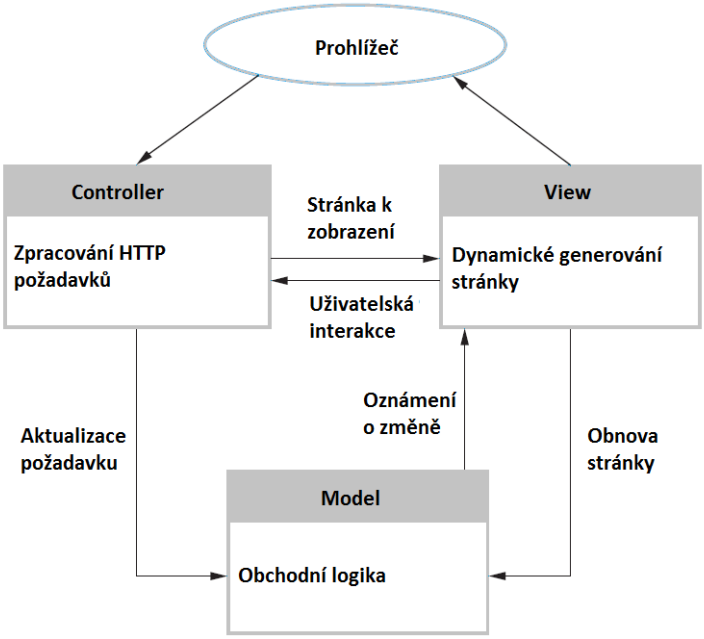
Konečnou dílčí architekturou je architektura technologická, která dohromady propojuje výše zmíněné části, jako je software, hardware, databáze, a vytváří tak celkový pohled a způsob tvorby informačního systému (Rábová, 2008, s. 21–23).

3.2.2 Architektura MVC

Architektura MVC, celým názvem Model-View-Controller, je populární architektura zaměřená na vývoj informačních systémů a webových aplikací. Jak název napovídá, architektura MVC se skládá ze tří částí, čímž je Model, View (Pohled) a Controller (Kontrolér) (Sommerville, 2011, s. 432).

Jednotlivé části jsou od sebe navzájem odděleny, a proto je možné je editovat samostatně s minimálním vlivem na ostatní součásti, čímž dochází ke značnému zjednodušení z hlediska implementace. Základním smyslem této architektury je oddělit grafické uživatelské rozhraní od datového modelu obsahujícího logické operace, jako jsou SQL dotazy na databázi či samotný kód aplikace. O správu komunikace mezi těmito dvěma částmi se stará řídící rozhraní (Munro, 2015, s. 432).

Využití má u složitějších návrhů, kde existuje více způsobů, jak prohlížet a zpracovávat data, nebo při zatím nejasných požadavcích na interakci a prezentaci dat. Umožňuje měnit data nezávisle na jejich vystupování, změna v jednom místě je zobrazena u všech (Freeman, 2014, s. 9–14).



Obrázek 3: Webová aplikace využívající MVC vzor (Sommerville, 2011, s. 157)

Model

O zmíněné logické operace datového modelu se stará první část architektury, což je Model. Model zahrnuje celkovou logiku systému, čímž vytváří datový základ pro další spolupracující části. Nejenže zahrnuje logickou stavbu aplikace, ale stará se i o přístup k datům uložených v databázi a jejich správu. I přes to, že se jedná o nejsložitější komponentu, model sám o sobě neví, jak budou zpracovávaná a ukládaná data reprezentována uživateli a jakou strukturu má zbytek celého systému.

Jeho úkolem je pouze přijatá data uložit, bez bližšího povědomí o zdroji, nebo je na základě podaných parametrů v databázi vyhledat a předat je části odpovídající za řízení komunikace.

View

View neboli pohled se stará o prezentaci uložených dat uživateli a o celkové grafické zobrazení aplikace. Nejčastěji je pohled vytvořen pomocí značkovacího jazyka HTML, který umožňuje tvorbu webových stránek a aplikací, s použitím kaskádových stylů CSS, které se starají o grafickou úpravu HTML tagů.

Pomocí formulářů jsou od uživatelů získávána data, která jsou následně předána kontroléru pro uložení do databáze.

Controller

Kontrolér je umístěn mezi modelem a pohledem, mezi kterými obstarává komunikaci, čímž dochází k propojení všech zmíněných částí a vytvoření provozuschopné architektury. Kontrolér pracuje na základě procesů pohledu, ve kterém dochází ke generování uživatelských akcí.

Pokud uživatel klikne na odkaz vztahující se k výpisu dat, kontrolér na to reaguje zasláním požadavku do modelu. Model má za úkol požadovaná data nalézt v databázi a pomocí kontroléru je předat pohledu, který se postará o jejich reprezentaci (Freeman, 2014, s. 9–14).

3.3 Přístupy k analýze

3.3.1 Strukturovaný přístup

Základním smyslem strukturovaného přístupu je hierarchický postup činností a rozdělení modelace na menší, vzájemně propojené celky. Znamená to tedy, že jednotlivé části mají výrazný přínos pro celkový vývoj informačního systému a hierarchicky vykonávané procesy usnadňují řízení vývoje a celkovou úsporu času.

Hlavními složkami strukturované analýzy jsou procesy probíhající uvnitř systému a data, respektive datová struktura, která je tvořena na základě vykonaných procesů. Tyto dvě součásti jsou analyzovány a modelovány odděleně pomocí speciálních CASE (Computer Aided Software Engineering) nástrojů s grafickým rozhraním (Vymětal, 2009, s. 30).

Jak bylo řečeno, strukturovaná analýza se skládá ze dvou oddělených částí, kterými jsou procesy a data, a proto jsou tyto části modelovány pomocí dvou rozdílných nástrojů.

Nástroj sloužící pro modelaci procesů se nazývá Data Flow Diagram (DFD) neboli diagram datových toků. Využívá se ke grafickému vyjádření vnitřních procesů informačního systému, které slouží k přeměně vstupních dat na výstupní se vztahem k externím entitám, nacházejícím se mimo budovaný systém. Jelikož je diagram datových toků vytvářen hierarchicky, hlavní proces v sobě zahrnuje další rozšiřující procesy pro detailnější analýzu systému. Součástí DFD je komponenta zvaná datový sklad, která znázorňuje úložiště dat a je primární komponentou pro následující diagram ERD (Dickerson, 2010, s. 109–113).

Entitně relační diagram (ERD) je druhým nástrojem strukturovaného přístupu, který má za úkol definovat datovou strukturu systému. V tomto diagramu jsou modelovány datové sklady z DFD, které se zde nazývají entity, a vztahy mezi nimi. Komponenty vytvořené v DFD musí korespondovat s entitami v ERD, jelikož tyto dva diagramy vytvářejí plnohodnotný model založený na strukturované analýze. Vytvořené datové sklady v DFD, které jsou obsluhovány procesy, proto musí mít přesně definovanou strukturu v ERD.

Výhodou strukturovaného přístupu je jednoduchost modelace díky grafickému rozhraní, které usnadňuje práci i méně zkušeným pracovníkům. Ovšem v dnešní době je již tento přístup mírně zaostalý a kvůli nepraktickému rozdělení funkční a datové struktury se dává přednost spíše přístupu objektovému (Kossiakoff, 2011, s. 380).

3.3.2 Objektově orientovaný přístup

Objektově orientovaný přístup je tvořen z objektů, které jsou definovány svými vlastnostmi a metodami. Mezi základní rysy objektů patří schopnost využití dědičnosti, při které mohou nižší objekty využívat vlastnosti či metody objektů vyšší úrovně.

Hlavním rozdílem od strukturovaného přístupu je zapouzdření, kdy jsou operace uvnitř třídy prováděny skrytě, čímž je umožněno využívat nezávislé objekty při provádění analýzy. Dále je možné aplikovat polymorfismus, který umožňuje využít stejně pojmenované metody, avšak s rozdílnou implementací.

Jednotlivé třídy jsou při návrhu spojovány pomocí vazeb, které představují vztahy mezi třídami, a takto propojené třídy dohromady představují zhotovený model systému. K tvorbě těchto modelů se využívá jazyk UML, který se v průběhu let stal oblíbeným nástrojem objektově orientovaného přístupu (Vymětal, 2009, s. 31).

UML

Jazyk UML (Unified Modeling Language) je univerzální jazyk pro modelování a vizualizaci návrhů informačních systémů, které jsou založeny na objektovém přístupu. UML je navrženo vhodným způsobem, aby mohlo být implementováno uvnitř CASE nástrojů.

Tento jazyk není svázán s žádnou metodikou, která by definovala, jak má návrh probíhat a za jakých podmínek. Nabízí pouze vizuální syntaxi a prvky pro sestavení návrhu, čímž vytváří statickou strukturu, do které spadají stavební bloky, společné mechanismy a architektura.

Do kategorie stavebních bloků náleží jednotlivé diagramy, znázorňující pohled na model, základní prvky a vztahy propojující jednotlivé prvky. Společné mechanismy a architektura definují obecné principy a pohled na modelovaný systém. Modely také obsahují dynamické chování, které definuje funkce navzájem spolupracujících prvků.

UML definuje několik typů diagramů, které umožňují modelovaný systém zaznamenat z různých hledisek. Vytvoření vhodného typu diagramu má značný vliv na podobu výsledného řešení.

Diagramy se dělí na diagramy chování, které definují chování výsledného systému, a na diagramy struktury, které komponenty znázorňují nezávisle na čase. Dále jsou představeny a popsány jen ty nejdůležitější a nejvyužívanější diagramy (Arlow, 2007, s. 28, 33–36).

Diagram případu užití

Diagram případu užití, originálním názvem Use case diagram, popisuje chování informačního systému z pohledu uživatele. Skládá se z prvků, kterými jsou případ užití, aktér a vztahy mezi nimi. Případ užití definuje konkrétní funkcionalitu systému, kterou může využívat aktér. Aktér představuje uživatele nebo externí systém, který má jakýsi vztah k vytvářenému systému.

Celková funkcionalita jednotlivých případů užití je definována postupnou interakcí aktéra a modelovaného systému, jež je slovně popsána a pojmenována jako scénář případu užití. Scénář představuje systematický postup kroků mezi aktérem a systémem od počátku do konce.

Tok informací mezi aktérem a případem užití je dán vztahem, kdy existují tři typy specifických vztahů, čímž jsou include, extend a generalizace. V případě použití include je možné do jednoho případu užití zahrnout i jiný případ užití, který se provede vždy při zavolání toho hlavního.

Při vztahu extend dojde k rozšíření původního chování, pokud je úspěšně splněna rozšiřující podmínka, případně může základní případ užití fungovat i samostatně. Vztah generalizace se nejčastěji využívá u aktérů, v jejichž případě dojde ke zjednodušení diagramu založením obecného společného předka, pomocí kterého dochází k propojení případů užití, jež vykonává většina aktérů (Arlow, 2007, s. 92–106).

Diagram tříd

Originálním názvem Class diagram, zachycuje strukturu všech tříd informačního systému, čímž vytváří statický pohled na model systému. Primárním prvkem tohoto diagramu je třída obsahující vlastnosti a metody, která je abstrakcí reálného objektu se stejnou charakteristikou. Jednotlivé třídy jsou vzájemně provázány pomocí vztahů, jejichž typy jsou asociace, kompozice a generalizace.

Asociace je základní vztah, který pouze zobrazuje vzájemně spolupracující třídy. Při kompozičním vztahu je do hlavní třídy přidána reference na odkazující třídu, která určuje povinnost vytvoření vedlejší třídy ještě před tou hlavní. Posledním vztahem je generalizace, což je abstraktní vztah nabízející sdílení stejných vlastností a metod skrze větší počet tříd, ovšem při zachování rozdílností tříd (Bruckner, 2012, s. 309–314).

Sekvenční diagram

Sekvenční diagram má za cíl graficky zobrazit průběh zpracování dat uvnitř systému za pomoci posílání zpráv. Zprávy jsou zasílány mezi objekty, případně mezi třídami a aktéry, kdy dochází ke spouštění metod, v rámci kterých vznikají sekvence zpráv, jež jsou přijímány jinými objekty. Právě tato sekvence je znázorňována sekvenčním diagramem.

Po získání zprávy objektem může dojít ke spuštění některé z jeho metod, kdy na základě různých zpráv mohou být spouštěny různé metody. V tomto diagramu jsou definovány dva typy zpráv, jimiž jsou synchronní a asynchronní zprávy.

Objekt při vyslání synchronní zprávy čeká na příjem zprávy o ukončení od protějšího objektu. Naproti tomu asynchronní zpráva je pouze zaslána a objekt již nečeká na doporučení o ukončení.

Tvorba tohoto diagramu je závislá na předchozím diagramu případu užití a vytvořených scénářích, které detailněji popisují celkový proces sekvence zpráv mezi aktéry a objekty informačního systému (Bruckner, 2012, s. 314–316).

Diagram aktivit

Diagram aktivit zobrazuje posloupný průběh aktivit zachycujících logiku scénáře vytvořeného v diagramu případu užití. Tento diagram je jakousi variantou zobrazení vývojového diagramu za pomoci objektově orientovaného přístupu.

Diagram se skládá z jednotlivých nedělitelných aktivit, které na sebe vzájemně navazují a jsou propojené pomocí toků. Součástí diagramu jsou rozhodovací bloky, které umožňují rozdělení toků na více částí, kdy vstupem je pouze jediný tok, ovšem výstupem může být toků několik.

Pro určení toho, kdo jakou aktivitu vykonává, se využívají plavecké dráhy, kdy je typ aktéra definován jednotlivou plaveckou dráhou, do níž jsou aktivity umisťovány (Arlow, 2007, s. 286–289).

3.4 Životní cyklus vývoje

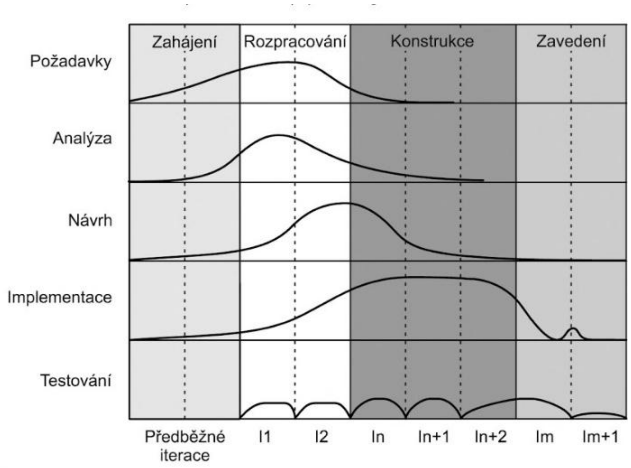
Vývoj informačního systému je složitý proces, při kterém dochází ke komplexnímu řešení návrhu a implementace na základě zákazníkových požadavků. Z hlediska vysoké náročnosti na řízení vývoje bylo v průběhu let zhotoveno několik typů metodik, které definují jednotlivé fáze a jejich obsah, s cílem usnadnění a urychlení celkové realizace informačního systému.

Obecná metodika, zabývající se tvorbou podnikových informačních systémů, se skládá z několika fází, kterými jsou specifikace zadání, analýza a návrh, implementace, zavedení a testování informačního systému (Vymětal, 2009, s. 67–71).

3.4.1 Požadavky

Jedná se o první fázi životní cyklu, ve které je blíže představen cílový zákazník, pro něhož bude informační systém vyvíjen. V tento moment je důležité získat od zákazníka veškeré požadavky na systém, na základě kterých bude vyhotoven návrh i následná implementace výsledného informačního systému.

Taktéž je významné získat bližší informace o samostatném cílovém podniku, jeho rozsahu a složení, jelikož tyto informace jsou pro vývoj a zvolení architektury či přístupu klíčové (Gála, 2015, s. 200–203).



Obrázek 4: Fáze vývoje informačního systému (Arlow, 2007, s. 330)

3.4.2 Analýza a návrh

Po získání klientských požadavků přichází na řadu fáze analýzy, ve které jsou požadavky detailně rozebrány a rozděleny podle logických celků. Klíčovou funkcí je odkrytí případných chyb v požadavcích a dohodnutí s klientem na možných alternativách, neboť nedostatečné odhalení slabin vede v dalších fázích ke komplikacím a složitějšímu odstranění náhle objevených chyb (Arlow, 2007, s. 137–139).

Na základě vytvořené analýzy klientských požadavků je vytvořen návrh informačního systému, jenž vizualizuje použité funkce a data. Návrh je možné zhotovit podle strukturovaného či objektového přístupu, jejichž rozdíl byl popsán v kapitole 3.3. Výběrem přístupu dojde k definování diagramů, které poslouží pro detailnější specifikaci požadavků a celkového návrhu informačního systému (Arlow, 2007, s. 330–335).

3.4.3 Implementace

Fáze implementace je věnována samostatné realizaci informačního systému za pomoci programovacích jazyků a technologií. Programování je realizováno na základě výsledků a modelů získaných a zhotovených v předešlých fázích. Jsou tedy naprogramovány operace, definované vstupy a výstupy a celková funkčnost, která je vizualizovaná při návrhu ve funkčním diagramu (Sodomka, 2008, s. 96–97).

3.4.4 Zavedení a testování

Poslední fází životního cyklu je testování, které zjišťuje kvalitu a funkčnost výsledné aplikace a její nabízené služby. Jsou ověřovány základní požadavky a je testováno, zda logika aplikace vrací očekávané výsledky. Po úspěšném dokončení této fáze je vytvořený software připraven k předání klientovi a nasazení do produkce (Vymětal, 2009, s. 95).

4 Použité technologie

V této kapitole jsou představeny technologie, pomocí kterých byla vyhotovena implementace řízeného skladu pro podnik DOBEŠ-STAVBY s. r. o.

Programovací jazyk C#

C# (C Sharp) je objektově orientovaný programovací jazyk vytvořený společností Microsoft, který vývojářům umožňuje vytvářet širokou škálu bezpečných a robustních aplikací běžících na frameworku .NET, jenž umožňuje vytvářet klientské aplikace, webové služby či databázové aplikace (Microsoft Developer Network, 2015).

Vytvořený zdrojový kód je překládán do spravovaného kódu (managed code). Spravovaný kód je mezijazykem, jelikož je na půl cesty mezi jazykem nejvyšší úrovně (C#) a jazykem nejnižší úrovně (jazykem symbolických adres, někdy také strojový kód).

O překlad zdrojového kódu se stará utilita CLR, celým názvem Common Language Runtime. Jedná se o virtuální stroj, jenž je zakomponován ve frameworku .NET. Slouží k tvorbě klientských a webových aplikací s využitím databáze. Jednotlivé metody jsou předkládány až v čase spuštění aplikace, což je definováno jako kompilace Just in Time. Klíčovou výhodou CLR je automatická správa paměti pomocí komponenty garbage collection, v českém znění sběrač odpadu, která se stará o alokaci volné paměti a dealokaci již nepotřebného paměťového úseku pro opětovné nové využití (Nash, 2010, s. 23–24).

Framework .NET

Framework .NET je primárně integrován v operačním systému Microsoft Windows a slouží k rozvíjení nových softwarových aplikací a webových služeb. .NET poskytuje vývojářům podporu pro programování a rozvíjení aplikací na platformě Windows (Microsoft TechNet, 2006).

ASP .NET

Technologie ASP .NET slouží k dynamickému zobrazování webových stránek. Tato technologie je založena na frameworku .NET, kdy pro zobrazení webové HTML stránky používá syntaxe Razor.

Razor se využívá k vytvoření dynamických webových stránek za pomoci programovacího jazyka C# nebo Visual Basic .NET. Při volání webové stránky server zpracuje kód uvnitř a až poté vrátí stránku prohlížeči (W3schools, 1999–2017).

JavaScript

Jedná se o interpretovaný programovací jazyk, který je využíván moderními webovými stránkami a aplikacemi. Slouží k tvorbě interaktivního a dynamického obsahu,

případně k automatizování administrativních procesů. Vytvořené zdrojové kódy jsou překládány až v době spuštění a stažení na straně klienta (Flanagan, 2011a, s. 19).

jQuery

jQuery je jedna z nejrozšířenějších knihoven založená na jazyce JavaScript, která je stavěna na jednoduché syntaxi. Usnadňuje hledání prvků v dokumentu a jejich následnou manipulaci, jako je přidávání obsahu, editace HTML tagů, případně CSS vlastností či provádění animací. jQuery obsahuje AJAX (Asynchronní JavaScript a XML) nástroje určené pro dynamické odesílání HTTP požadavků a utility pracující s objekty (Flanagan, 2011b, s. 13–14).

TypeScript

Byl vytvořen jako nadstavba jazyka JavaScript, jenž je rozšířen o prvky objektově orientovaného programování. Dále je pak rozšířen o třídy, rozhraní, typovou kontrolu, IntelliSense a refaktorizaci kódu. Vytvořený zdrojový kód je z TypeScriptu vygenerován do JavaScriptu, ve kterém dochází ke kompilaci (Dajbych, 2013).

Bootstrap

Boostrap je webový open-source framework, který obsahuje CSS šablony a skripty jazyka JavaScript. Slouží pro front-end programování responzivních webových stránek. V dnešní době se jedná o nejpoužívanější framework. Při použití je programátorovi ušetřen čas i značná spousta kódu. Výsledkem je jednotný styl na všech dostupných internetových prohlížečích (Internet Explorer, Google Chrome, Mozilla Firefox a Opera) (Twitter, 2011–2017).

Entity Framework

Platforma .NET obsahuje nepřeberné množství různých ORM (Object relational mapping) frameworků, které zajišťují hladký převod mezi daty získanými v aplikaci a relační databází. Dříve byl hojně využíván framework NHibernate, od kterého se opouští a přechází se na, v dnešní době nejpoužívanější, ORM framework zvaný Entity Framework.

Ten nabízí dvě možnosti, jak řešit implementaci webové aplikace. První možností je navržení a vytvoření databáze, na jejímž základu bude implementace zhotovena. Druhou možností je tak zvaná metoda Code first, která dovoluje nejdříve napsat samotný kód aplikace a až poté zhotovit relační databázi (Entity Framework Tutorial, 2016).

5 Existující řešení

Před tvorbou návrhu a implementace vlastního řešení bylo zapotřebí nalézt dostupné informační systémy konkurenčních společností, zda nesplňují zmíněné zadání.

Hledány byly systémy, které nabízejí podporu pro řízený sklad a jsou určeny pro malé či střední podniky. Jelikož zákazníkovým požadavkem bylo zhotovení informačního systému jako webové aplikace, při které není nutná instalace na klientské stanici, byla snaha vybírat systémy, které by tento požadavek dokázaly uspokojit. Na vědomí bylo bráno plnění požadavků i finanční náročnost.

5.1 Pohoda 2017 Jazz

Prvním vybraným systémem je skladové hospodářství POHODA Jazz vyvíjeno společností STORMWARE s ručením omezením. Bohužel se nejedná o webovou aplikaci, nýbrž o desktopovou, která musí být nainstalována. Podporuje i mobilní verzi, kdy je možné aplikaci využívat na tabletech a chytrých telefonech.

Nabízí velké množství služeb zahrnující funkce pro sklad, fakturace a inventury. Dále poskytuje hotovostní a internetový prodej, automatické objednávání zásob či obsluhu zboží pomocí čteček čárových kódů.

Součástí tohoto modulu je i základní balík služeb, který zahrnuje oblast pro finance, elektronickou evidenci tržeb (EET), faktury a adresář, kvůli kterému je ovšem navýšena výsledná cena. Verze Pohoda 2017 Jazz se síťovou variantou pro 4 až 5 klientů je dostupná za 11 960 Kč (POHODA Jazz, 2017).

5.2 MS Dynamics 365

Dalším produktem je MS Dynamics 365, který má obrovskou výhodu ve snadnější integraci s dalšími produkty od společnosti Microsoft, případně možnost propojení s aplikací třetích stran. Naopak velkou nevýhodou je, že jsou data nahrávána pouze do cloudu nebo je cloud kombinován s lokálním serverem. Aplikace je přizpůsobena požadavkům zákazníka s individuálním cenovým rozsahem a na stránkách výrobce proto cena není uvedena (Dynamics Online, 2016).

5.3 Softwarový modul pro správu skladu

Altus Vario nabízí velké spektrum modulů, z nichž jeden je určen pro správu skladu. Tento software pokrývá náročné požadavky firem obchodujících se zbožím. Výhodou tohoto řešení je, že lze tento modul využívat naprosto samostatně, tudíž se neplatí za ostatní služby.

Modul nabízí širokou škálu služeb, týkající se skladu a jeho správy. Mezi služby patří řízení skladových zásob, evidence umístění skladových položek, evidence variant produktů a taktéž propojení s e-shopem.

Tato aplikace nabízí pouze desktopovou verzi. Cena pro malou až střední firmu obsahující deset klientů se pohybuje od 15 000 do 250 000 korun českých (Altus software, 2017).

5.4 Skladové hospodářství

S tímto modulem přichází akciová společnost ABRA Software, která nabízí přístup do skladu s aktuálním stavem, zobrazení objednávek a rezervací produktů. Možnost práce se sériovými čísly, sledování sériových čísel a zobrazování dokladů ze skladových karet. Navíc oproti ostatním systém podporuje reporty a analýzy obratů, které umožňují nastavení ideálního množství produktů na skladě. Společnost bohužel nenabízí informace o cenovém rozsahu tohoto systému (Abra Software, 2017).

5.5 Instantní sklad

Posledním vybraným systémem je Instantní sklad od společnosti CCV Informační systémy s. r. o. Jedná se o firmu s dlouholetou tradicí a velkým počtem zákazníků.

Instantní sklad je webová aplikace, jež nabízí spousty funkcí, mezi které patří vyhledávání zboží, grafický přehled využití skladu, evidence produktů s konkrétním sériovým číslem či jednotkou balení a dalšími doplňkovými vlastnostmi.

Systém je zpoplatněn měsíčním paušálem, kdy částka při nízkém počtu uživatelů činí 1 470 Kč, která ovšem s nárůstem uživatelů stoupá (CCV Informační systémy, 2017).

5.6 Závěr

Po nalezení a zhodnocení dostupných informačních systémů, které jsou určené pro skladové hospodářství, bylo zjištěno, že většina firem nabízí podobný základ modulu, který se liší pouze vylepšenými funkcemi a grafickým zpracováním. Hlavním rozdílem v uvedených systémech jsou vložené finance, které se mnohdy odvíjí od obsažených funkcí nebo doplňkových modulů. Některé společnosti cenu vytváří individuálně podle obsažených funkcí systému, jiné nabízí obecný systém pro malé až střední podniky s pevnou cenou pohybující se od 12 000 až po 250 000 Kč.

Pouze jeden zástupce z vybraných systémů je dostupný ve formě webové aplikace, která je ovšem velmi komplexní s velkým množstvím funkcí, které by zadavatelem DOBEŠ-STAVBY s. r. o. nebyly značně využity. Systém je poskytován jako služba s měsíčním paušálem, což je pro uvedený podnik vlastnící svou infrastrukturu neefektivní a nákladnější.

Bude proto navrhnut a implementován vlastní informační systém, který bude vytvořen podle konkrétních požadavků a přání zadavatele. Vyhotoven bude modul pro skladové hospodářství, jenž bude dostupný jako webová aplikace. Největší výhodou pro zadavatele je cena, jelikož systém bude navrhnut a zhotoven bez nutnosti finančního ohodnocení.

6 Metodika práce

Tvorba informačního systému bude zhotovena podle obecné konstrukce životního cyklu, který zahrnuje několik fází, jež byly detailně popsány v kapitole 3.4.

V prvotní fázi, která má za úkol specifikovat zadání a analyzovat potřeby, dojde k diskuzi s vedením podniku DOBEŠ-STAVBY s. r. o. Na základě získaných detailních požadavků a informací budou zjištěny potřebné funkce, které by měly být v novém informačním systému zahrnuty. Součástí analýzy bude taktéž rozbor momentálně využívaného informačního systému, který pro plnohodnotnou podporu řízení a chodu podnikání již není plně dostačující.

Hlavním požadavkem na nový informační systém, určený pro řízený sklad, je tvorba ve formě webové aplikace. Další získané požadavky budou v rámci analýzy rozděleny do dvou skupin, jimiž jsou funkční a nefunkční požadavky. Funkční požadavky znázorňují jednotlivé funkce a procesy, které jsou jádrem logiky celého systému. Naopak nefunkční požadavky uvádějí pouze technické a programové vlastnosti.

Ze zadaných požadavků bude namodelován návrh nového informačního systému za pomoci objektového přístupu. K modelování bude využit nástroj Visual Paradigm, jenž je podporován jazykem UML, který slouží pro grafický návrh a vizualizaci systémů.

Při návrhu dojde k vytvoření těchto diagramů:

- Diagram případu užití zobrazující vztahy mezi aktéry a službami systému. Jednotlivé případy užití budou specifikovány pomocí scénářů.
- Diagram tříd definující datové struktury pomocí tříd a vztahů mezi nimi.
- Sekvenční diagram znázorňující uspořádanou komunikaci mezi objekty.
- Diagram aktivit určující řízení toků pracovních postupů.
- Drátěný model pro vizuální představu výsledné webové aplikace.

Po zhotovení modelů bude zahájena realizace prototypu webové aplikace. Aplikace bude implementována pomocí technologie ASP .NET s využitím architektury MVC a s aplikováním frameworku .NET verze 4.5.2. Při implementaci bude využito databázových entit obsahujících základní vlastnosti, které jsou ukládány do databáze, a webových modelů, kdy jsou atributy získávány za pomoci matematického výpočtu nebo logických operací, případně jejich kombinací.

Aplikován bude ORM framework, který využívá Code First. Při implementaci bude využita lokální databáze, která je poskytována MSSQL serverem. Pro základní zobrazování bude použit jazyk Razor a pro generování dynamického obsahu jazyk TypeScript. Dále budou uplatněny technologie JavaScript, jQuery a C#.[1]

Poslední fází při tvorbě modulu pro řízený sklad bude testování, které zajistí ověření funkčnosti celého systému a eliminaci kritických chyb. K testování budou

[1] Využité technologie jsou jednotlivě vysvětleny v kapitole 3.4.3.

využity testovací účty, které odzkouší všechny funkce systému a zjistí nedostatky, které budou následně opraveny.

Po řádném otestování aplikace a zajištění bezchybnosti, bude možné představit tento prototyp podniku a po případné domluvě zajistit výměnu dosavadního informačního systému za nově vytvořený.

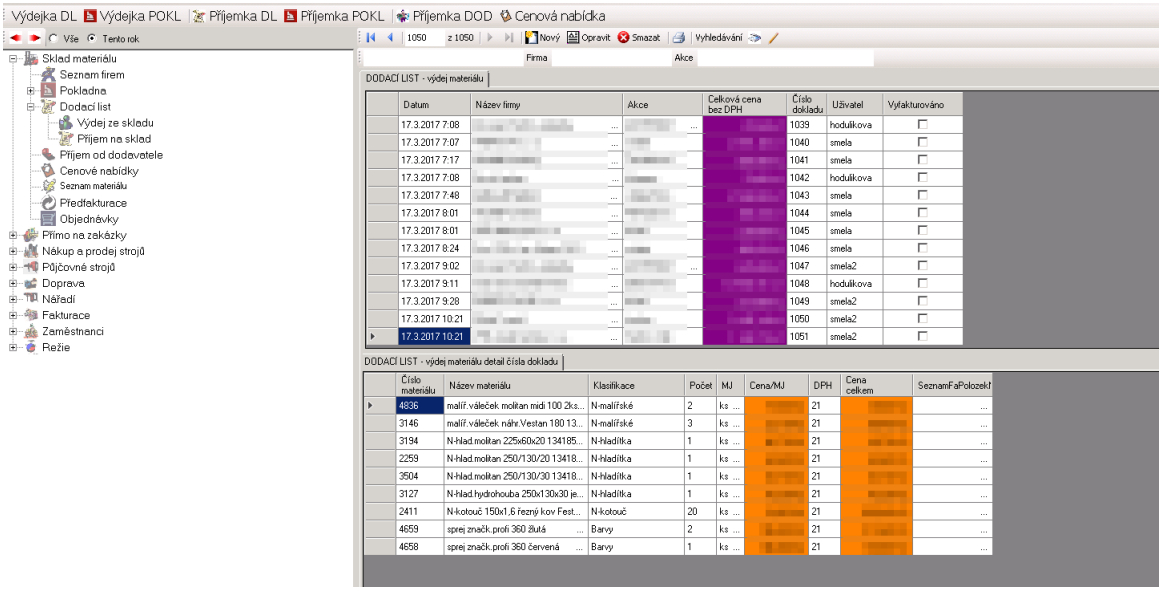
7 Stanovení požadavků

Informační systém byl vytvořen pro podnik DOBEŠ-STAVBY s. r. o., který se zabývá prodejem stavebního materiálu, výpůjčkou stavebních strojů a stavebními pracemi. Firma momentálně využívá podpůrný informační systém, který je nedostatečný z pohledu funkčních požadavků podniku na zpracování zakázek, a proto bylo potřeba zhotovit nový.

Klíčovým prvkem informačního systému je modul určený pro sklad, který slouží pro evidenci stavebního zboží s neustále se měnícím stavem.

7.1 Stávající informační systém

Stávající systém je tvořen staršími technologiemi, kdy byl vytvořen jako desktopová aplikace pomocí programovacího jazyka Visual Basic 2005. V dnešní době je to z pohledu plnění funkčních požadavků podniku méně efektivní, jelikož systém je přístupný pouze z pevného či přenosného počítače, na kterém je nainstalován operační systém Windows. Tím pádem není možné k informačnímu systému přistoupit pomocí mobilního zařízení. Vzdálený přístup k systému mimo podnikovou síť je velmi náročný a pro správu velmi nekomfortní.



Obrázek 5: Stávající informační systém (DOBEŠ-STAVBY s. r. o., 1995–2017)

Jelikož je desktopová verze nynějšího informačního systému založena na starších technologiích, práce s podnikovými daty a jejich modifikace je časově náročnější a dochází ke značnému zpoždění. Výhodou připojení k aplikaci pomocí interní sítě je zvýšená bezpečnost, neboť je systém méně ohrožen vnějšími útoky skrz Internet.

Nynější procesy patřící do stávajícího informačního systému se dají rozdělit na vnější a vnitřní. Do vnějších procesů lze zahrnout proces naskladnění zboží, do něhož vstupuje externí firma, která zboží obstarává v časovém úseku.

Vnitřní procesy zahrnují veškeré podnikové činnosti, které se skladu týkají. Mezi důležité procesy patří vkládání dodaného zboží na sklad (naskladnění) a jeho následné vyskladnění a dodání cílovému zákazníkovi. Dalšími vnitřními procesy jsou modifikace cen zboží, evidování informací o dodavatelích a případných stávajících odběratelích, vytváření dodacích listů a skladových karet.

7.2 Funkční požadavky

Po konzultaci s vedením podniku DOBEŠ-STAVBY s. r. o., byly sepsány funkční požadavky a bylo stanoveno, že systém může být využíván dvěma typy zaměstnanců, a to vedoucím a skladníkem.

- Zaměstnanci před vstupem do systému musejí zadat korektní, jim přiřazené přihlašovací údaje, které jsou složeny z e-mailové adresy a hesla.
- Po ukončení práce bude zaměstnanci umožněno odhlášení.
- Přihlášený zaměstnanec bude mít možnost změnit si své přihlašovací heslo.
- Přihlášený zaměstnanec bude mít možnost zobrazit si osobní údaje v profilu.
- Zaměstnanec s rolí *Vedoucí* bude moct evidovat nové zaměstnance a přiřazovat jim role.
- Vytvořenému zaměstnanci bude možné přiřadit nový účet, pod kterým se bude moci přihlásit a pracovat.
- Evidování základních kontaktních údajů o zaměstnancích bude možné po vstupu do detailu konkrétního zaměstnance. Všechny tyto údaje bude vyplňovat pouze *Vedoucí*, aby nedošlo ke zkreslení či zfalšování ze strany zaměstnance.
- Při ukončení pracovněprávního vztahu musí být zaměstnanec deaktivován pro zamezení přístupu.
- *Vedoucí* bude evidovat nové subjekty, právnické či fyzické osoby, které s podnikem spolupracují formou dovozu či odběru zboží.
- Subjektům bude možno nahlédnout do detailu a provádět úpravy osobních a kontaktních údajů, případně IČO a DIČ.
- Po ukončení spolupráce bude subjekt odstraněn z evidence.
- *Vedoucí* bude mít na starosti vytváření skladových karet pro jednotlivé produkty a nastavování údajů, jimiž jsou název karty a maximální množství produktů.
- Skladové karty budou editovatelné a odstranitelné.

- *Vedoucí* bude mít možnost nahlížet do skladových karet pro snazší přehled.
- Zaměstnanec typu *Skladník* bude mít možnost formou dodacích listů neboli příjemky a výdejky evidovat informace o materiálech, které byly zakoupeny na sklad nebo prodány.
- Evidované zboží v dodacím listě bude automaticky přiřazeno konkrétní skladové kartě. V případě výdejky bude přidán záznam o vyskladnění.
- *Skladníkovi* bude umožněno celý dodací list mazat, případně odebírat některé vložené záznamy.
- Při vkládání a odběru záznamů ze skladové karty bude docházet k automatické kontrole skladového limitu.
- V případě nízkých či vysokých zásob na skladě bude odesláno notifikační upozornění všem vedoucím na jejich uvedenou e-mailovou adresu.
- Po vytvoření dodacího listu do něho bude moct být vložen soubor, představující fakturu či doplňkové údaje.
- Objednávka zboží bude prováděna pomocí e-mailové adresy s výběrem konkrétního subjektu.

7.3 Nefunkční požadavky

V budově podniku jsou umístěny počítače a tiskárny, které jsou určeny pro práci zaměstnanců. Na počítačích jsou nainstalovány operační systémy od verze Windows 7 až po verzi Windows 10 od společnosti Microsoft. V rámci podnikové lokální sítě je umožněn přístup do Internetu, k čemuž slouží směrovač se zabudovaným firewallem pro zabezpečení provozu komunikace.

Vytvářený informační systém by měl být optimalizován pro běžné uživatele, kteří využívají standartní prohlížeče, jimiž jsou Google Chrome, Mozilla Firefox, Opera a Internet Explorer. Pro plnou funkci a podporu se od těchto prohlížečů očekává nejnovější verze, např. Internet Explorer verze 11.

Server

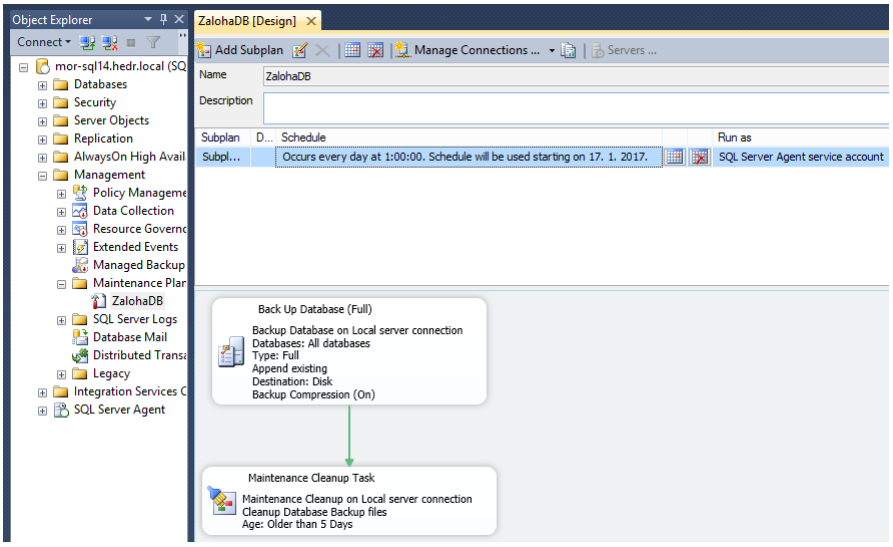
Podnik vlastní datové servery, které se starají o evidenci skladových zásob. Podnik pro databázi využívá Microsoft SQL Server 2014 Developer 6 zhotovený jako virtuální stroj, který je spuštěn na operačním systému Windows Server 2012R2 Standard. K serveru je možné přistoupit v lokální síti pomocí adresy *mor-sql14.hedr.local*.

Name	MOR-SQL14
Product	Microsoft SQL Server Developer (64-bit)
Operating System	Microsoft Windows NT 6.3 (9600)
Platform	NT x64
Version	12.0.5540.0
Language	English (United States)
Memory	15997 MB
Processors	4
Root Directory	C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL
Server Collation	Czech_CI_AS
Is Clustered	False
Is HADR Enabled	False
Is XTP Supported	True

Obrázek 6: Vlastnosti podnikového serveru (DOBEŠ-STAVBY s. r. o., 1995–2017)

Zálohování databáze

S vedením bylo domluveno, že záloha dat bude probíhat každý den, jednu hodinu po půlnoci, kdy jsou servery minimálně využívány a hrozí jen minimální riziko ztráty dat. Data budou ukládána ve formě komprimované zálohy do datového úložiště NAS Synology DS414+.



Obrázek 7: Nastavení zálohování (DOBEŠ-STAVBY s. r. o., 1995–2017)

8 Návrh řešení

V této části jsou prezentovány modely, které byly vytvořeny na základě zadaných požadavků podniku DOBEŠ-STAVBY s. r. o. Modely jsou vytvořeny se vztahem k webové aplikaci, která bude využívána dvěma typy zaměstnanců, mezi které patří role *Vedoucí* a *Skladník*.

8.1 Diagram případu užití

Diagram případu užití specifikuje aktéry využívající funkcionality systému. Kvůli vysoké složitosti je diagram dostupný v příloze D.

V modelu vytvářeného systému jsou obsaženy dva základní typy aktérů. Prvním aktérem je *Čas*, který obstarává zálohování dat v databázi.

Druhým aktérem je *Zaměstnanec*, do kterého lze zahrnout *Nepřihlášeného* a *Přihlášeného zaměstnance*, a slouží tedy jako obecný předek. *Nepřihlášený zaměstnanec* má možnost se pouze přihlásit do systému, jiná funkce mu není poskytnuta. Naopak *Přihlášeného zaměstnance* lze rozdělit na další dva potomky, jimiž jsou dvě požadované role *Vedoucí* a *Skladník*.

Aktér *Vedoucí* má právo provádět úkony nad zaměstnanci. Může tedy vytvářet a registrovat do systému nové zaměstnance a editovat jejich údaje po zobrazení detailu. Zaměstnanec může být i deaktivován, čímž ztrácí možnost vstupu do systému, ovšem údaje zaměstnance jsou ponechány v databázi z důvodu archivace a pozdějšího dohledání záznamů, které byly deaktivovaným zaměstnancem vytvořeny.

Další složka spadající pod aktéra *Vedoucí* jsou subjekty, které představují dodavatele a odběratele zboží. Subjekty se mohou dělit na fyzickou a právnickou osobu, kdy u právnické osoby musí být navíc evidováno IČO a DIČ.

U procesů se subjekty platí podobný princip jako u zaměstnanců. Subjekty je možné evidovat, ať už se jedná o fyzickou nebo právnickou osobu, upravovat, případně deaktivovat. Pro pozdější dohledání subjektu zůstává nadále uložen v systému.

Posledním dílem spadajícím pod *Vedoucího* jsou skladové karty, které modifikuje a vytváří, čímž umožňuje práci aktérovi *Skladník*. Jde o případy užití jako *Smazat skladovou kartu*, *Vytvořit skladovou kartu* a *Editovat skladovou kartu*.

Aktér *Skladník* pracuje s blokem dodacích listů, do kterého lze zařadit případy užití *Vytvořit dodací list* a *Vymazat dodací list*. Detail slouží pro aktéra *Vedoucí*, který tak kontroluje již vytvořené dodací listy.

Skladník, jenž vytvoří nový dodací list, má možnost do něho vkládat záznamy, které se posléze projeví ve skladové kartě. K dodacím listům je možno přidávat soubory, jež slouží jako podklady. Protože *Skladník* pracuje s dodacími listy, musí mít vztah i s částí skladových karet, které jsou vytvářeny aktérem *Vedoucí*.

Skladová karta je jedna z nejdůležitějších částí systému. V této části se evidují jednotlivé produkty, jejich cena, množství a typ a přidávají se jednotlivé produkty. Při dodání zboží, které je přidáváno právě pomocí dodacího listu, je zavolán případ užití *Zkontrolovat skladové zásoby*, který je vytvořen pro automatizovanou kontrolu

zásob. *Skladník* má navíc možnost *Přidat záznam do skladové karty* a *Odebrat záznam ze skladové karty*.

Po přidání nastane kontrola, zda není sklad přetížen, nebo naopak, zda skladu nedochází zásoby. Pokud takováto situace nastane, odesílá se notifikační e-mail všem *Vedoucím*, kteří jsou uvědoměni o nastalé situaci. Stejný případ užití je použit i při odebírání zboží ze skladové karty.

8.2 Scénář případu užití

Tabulka 1: Scénář odpovídající případu užití *Vytvořit dodací list*

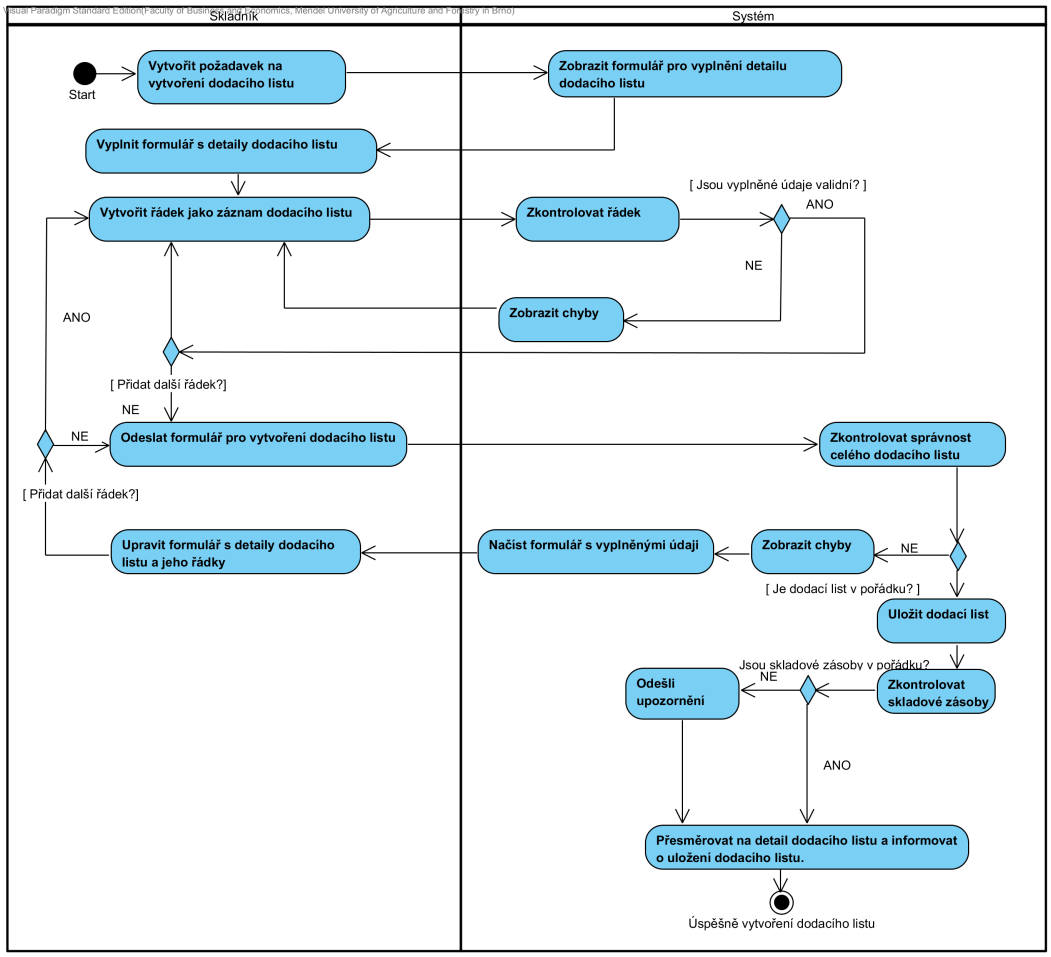
	Skladník	**Systém**
1	Vytvořit požadavek na vytvoření dodacího listu.	
2		Zobrazit formulář pro vyplnění detailu dodacího listu.
3	Vyplnění formuláře s detaily dodacího listu.	
4	Vytvořit řádek jako záznam dodacího listu.	
5		Zkontrolovat řádek.
6	Odeslat formulář pro vytvoření dodacího listu.	
7		Zkontrolovat správnost celého dodacího listu.
8		Uložit dodací list.
9		Zkontrolovat skladové zásoby.
10		Přesměrovat na detail dodacího listu a informovat o uložení dodacího listu.
5a		Zobrazit nalezené chyby.
7a		Zobrazit nalezené chyby.
7b		Načíst formulář s vyplněnými údaji.
7c		Upravit formulář s detaily dodacího listu a jeho řádky.
9a		Odeslat upozornění na e-mail.

Při návrhu musel být pro každý případ užití ve vytvořeném diagramu případu užití specifikován scénář, který detailně popisuje jeho funkcionalitu probíhající mezi aktérem a systémem.

V práci je uveden scénář, který náleží případu užití jménem *Vytvořit dodací list*, který je obsloužen aktérem *Skladník*. V něm probíhá tvorba nového dodacího listu a vkládání jednotlivých záznamů, po jehož uložení dojde systémem ke kontrole odeslaných dat z formuláře i skladových zásob. Celkový průběh scénáře je v tabulce 1. Na základě uvedeného scénáře byl vytvořen sekvenční diagram a diagram aktivit.

8.3 Diagram aktivit

Zobrazuje činnosti, které jsou zařazeny do plavecké dráhy aktéra *Skladník* nebo do dráhy informačního systému. Diagram aktivit je vytvořen na základě hlavního scénáře, který může být rozvětven scénářem alternativním, kvůli čemuž vznikají rozhodovací bloky.



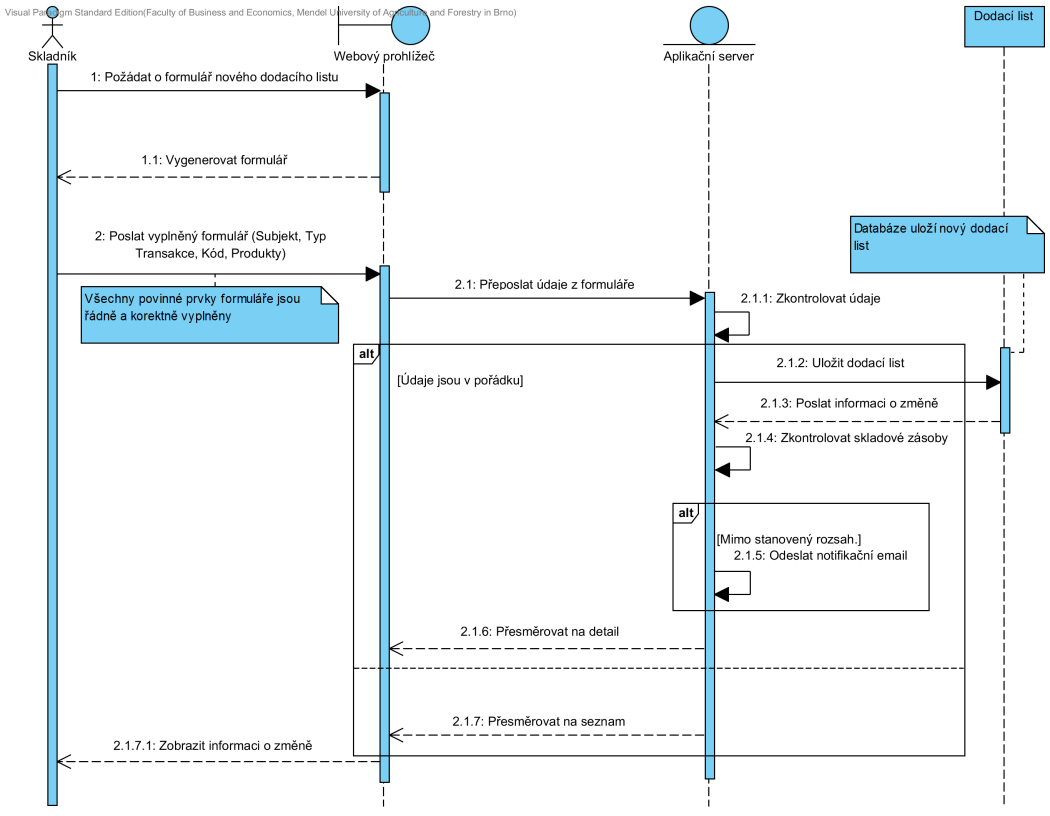
Obrázek 8: Diagram aktivit pro případ užití *Vytvořit dodací list*

Jak je vidět na obrázku 8, diagram začíná iniciativou *Skladníka*, kdy požaduje vytvoření nového dodacího listu, na což musí systém reagovat zobrazením formuláře. Poté je *Skladníkovi* umožněno vyplnění detailů dodacího listu a přidání řádku, který představuje naskladněné zboží. Po přidání řádku dojde systémem ke kontrole zadaných údajů a v případě nevalidních dat jsou chyby vypsány uživateli.

Počet řádků je neomezený a po ukončení vkládání záznamů dojde ke kontrole celého odeslaného formuláře a v případě nenalezení chyby i k jeho uložení. Poté systém provede kontrolu skladových zásob. Pokud jsou zásoby v limitu, dojde k přesměrování a zobrazení informativní hlášky. Ovšem pokud jsou zásoby mimo stanovený rozsah, je na e-mailovou adresu *Vedoucích* odesláno upozornění.

8.4 Sekvenční diagram

Druhý diagram vytvořený z nadefinovaného scénáře ze strany 33 je diagram sekvenční, který se skládá ze čtyř čar života. První čára života reprezentuje uživatele, jímž je *Skladník*, poté následuje webový prohlížeč, aplikační server a databáze.



Obrázek 9: Sekvenční diagram pro případ užití *Vytvořit dodací list*

V prvním kroku je *Skladníkem* vyslán požadavek na webový prohlížeč, který vygeneruje nový formulář pro zadávání údajů nového dodacího listu. Po vyplnění formuláře jsou získaná data zaslána přes webový prohlížeč aplikačnímu serveru, který zajistí zkontrolování údajů.

Při úspěšné kontrole jsou data uložena do databáze, provedeno ověření rozsahu skladových zásob a webovému prohlížeči zaslán požadavek na přesměrování stránky na detail nově vytvořeného dodacího listu. Pokud dojde k chybě již při kontrole údajů získaných z databáze, je provedeno přesměrování na seznam dodacích listů, bez uložení dat do databáze.

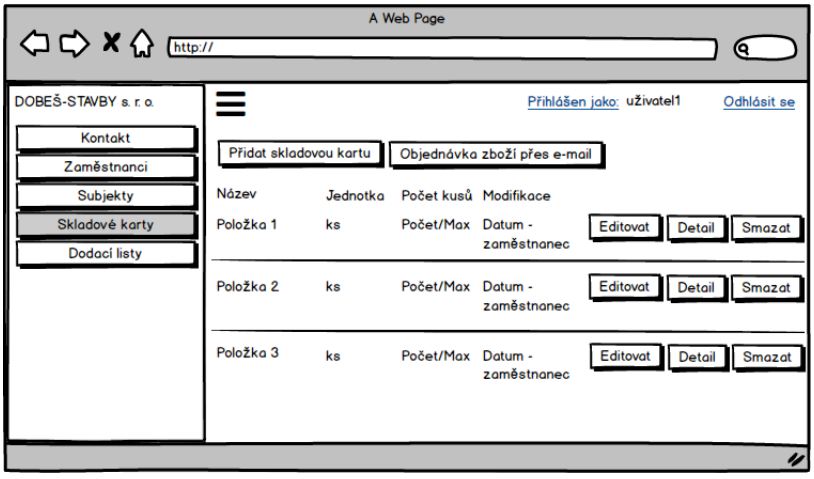
8.5 Diagram tříd

Součástí analýzy je i tvorba diagramu tříd, který znázorňuje typy objektů v systému a jejich vzájemné vztahy. Tento diagram je základem pro implementování aplikace. Kvůli obsáhlosti je diagram obsažen v příloze C.

8.6 Drátěný model

Pro vizuální představu rozmístění prvků webové aplikace bylo vytvořeno několik drátěných modelů, které byly prezentovány vedení podniku.

Na obrázku 10 je zobrazen model, který představuje pohled na aplikaci s rolí *Vedoucí* a návrh pro výpis skladových karet. Jak bylo uvedeno v požadavcích, tato role má možnost manipulovat jak se zaměstnanci a subjekty, tak i se skladovými kartami, a může nahlédnout i do dodacích listů. Ostatní vytvořené drátěné modely jsou součástí přílohy.

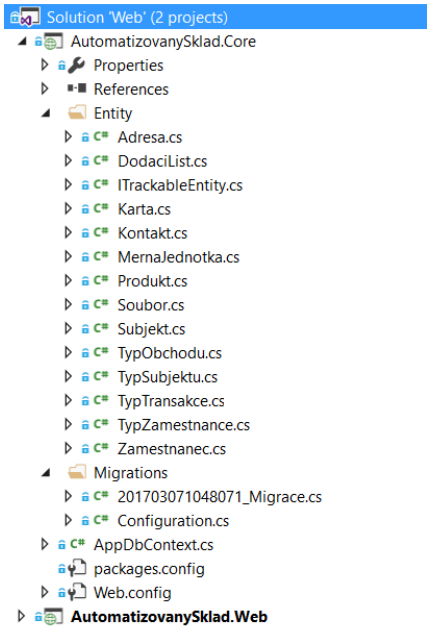


Obrázek 10: Drátěný model pro zaměstnance *Vedoucí*

9 Implementace

9.1 Struktura aplikace

Z důvodu snadného rozšíření či modifikace aplikace v budoucnu byl projekt rozdělen do dvou vrstev. První vrstva obsahuje objekty neboli entity, které jsou pomocí Entity Framework mapovány do databáze. Tato vrstva je v projektu pojmenována jako AutomatiozvanySklad.Core, která má funkci knihovny tříd (Class Library). Entity byly vytvořeny na základě předem navrhnutého diagramu tříd.



Obrázek 11: Rozdělení aplikace

Druhá vrstva zahrnuje veškeré webové modely, kontroléry a pohledy spadající do architektury MVC, dále JavaScript soubory, kaskádové styly a další konfigurační soubory.

9.1.1 Migrace databáze

Pomocí vytvořených entit v první vrstvě projektu byly vytvořeny databázové tabulky. K tomuto vytvoření bylo zapotřebí spuštění konzole Package Manager Console s následným příkazem Enable-Migrations, díky kterému došlo k povolení migrace první vrstvy do databáze.

Poté se musela nastavit třída AppDbContext, která se stará o celkové propojení jednotlivých tříd s databází, jak je vidět v následujícím ukázkovém kódu. Posledním krokem bylo vytvoření migrace za pomoci příkazu Add-Migration, která definuje strukturu databáze.

```
using System.Data.Entity;
using AutomatizovanySklad.Core.Entity;

namespace AutomatizovanySklad.Core {
  public class AppDbContext : DbContext {
    public AppDbContext() : base("SkladDb") {

    }
    public DbSet Adresy { get; set; }

    public DbSet Karty { get; set; }

    public DbSet Kontakty { get; set; }

    public DbSet Produkty { get; set; }

    public DbSet Subjekty { get; set; }

    public DbSet Zamestnanci { get; set; }

    public DbSet Soubory { get; set; }

    public DbSet DodaciListy { get; set; }
  }
}
```

Dalším krokem je nastavení ConnectionString, který určuje přesnou cestu k databázi, jejíž jméno musí být shodné se jménem určeným ve třídě AppDbContext. Ten je obsažen v konfiguračním souboru web.config, který je součástí druhé vrstvy.

9.1.2 Autentizace a autorizace

Jelikož je přístup do informačního systému omezen a povolen pouze oprávněným uživatelům, byl využit již existující framework AspNet Identity 2.2.1., který ulehčuje správu a tvorbu uživatelských účtů a zabezpečuje hesla pomocí hašovací funkce.

Pro tyto potřeby bylo zapotřebí propojit uživatelské e-mailové adresy, pomocí kterých je umožněn přístup do systému, s e-mailovými adresami uloženými v tabulce AspNetUsers. A také propojení uživatelských účtů s jednotlivými identifikačními čísly (ID) s tabulkou AspNetUserRoles. Díky tomu lze použít atribut kontrolující práva vstupujícího uživatele.

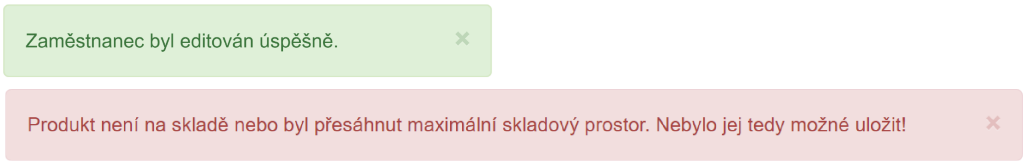
```
[Authorize(Roles = "Vedouci,Skladnici")]
  public class KartaController : BaseController {
```

9.2 Funkce informačního systému

Informační systém je zhotoven jako webová aplikace, která reaguje na uživatelské vstupy a požadavky, při kterých jsou volány události a prováděny funkce zajišťující chod celé aplikace.

Informativní hláška

Po každém uživatelském zásahu je vypisována informativní hláška, která je zpětnou vazbou pro uživatele. Tato hláška informuje o úspěšném provedení akce či nastalé neočekáváné chybě. Typy těchto hlášek jsou vyobrazeny na obrázku 12.

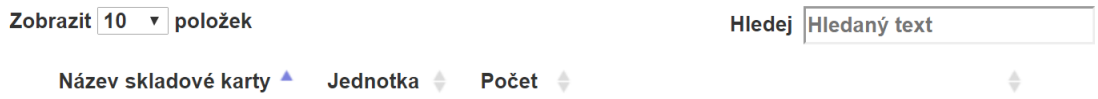


Obrázek 12: Informativní hlášky

Řazení seznamu

Výpis všech seznamů, například uživatelů či produktů, je doplněn o funkci řazení. Výchozí řazení je nastaveno podle prvního sloupce, který je seřazen podle abecedy.

Každý sloupec má možnost seřazení, a proto jsou vedle názvu sloupce umístěny šipky (obrázek 13). Šipka směřující nahoru sloupec seřadí sestupně a směřující dolů vzestupně.



Obrázek 13: Řazení seznamu

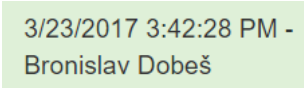
Dále je možné u seznamu zobrazovat položky po omezeném množství, případně zobrazit celý výpis pouze na jednu stranu, čímž dochází ke zhoršení přehlednosti. Výchozí hodnota je nastavena na deset záznamů na jednu stránku. Pod seznamem jsou zobrazeny očíslované listy obsahující dostupné záznamy (obrázek 14).

Obrázek 14: Stránkování seznamu

Pro tuto funkci byl využit již existující plug-in s názvem DataTables, což je zásuvný modul pro knihovnu jQuery. Tento modul byl přidán do projektu vrstvy AutomatizovanySklad.Web, který je vyvíjen v aplikaci Visual Studio Community. V ní byla otevřena konzole, vybrán zdroj nuget.org a vložen příkaz Install-Package jquery.datatables, který zajistí stažení tohoto balíčku.

Informace o modifikaci

Při každé modifikaci záznamu je zachycena informace o uživateli, který změnu provedl. Spolu s ním je evidováno i časové razítko. Tyto zaznamenané informace jsou součástí záznamů, se kterými jsou rovněž vypisovány.



Obrázek 15: Informace o modifikaci

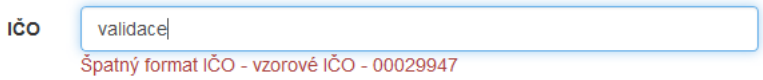
Časové razítko se vytváří nebo mění na základě metod *PridejDatumVytvoreni* nebo *PridejDatumZmeny*, jež jsou prezentovány v následujícím kódu. Do metod vstupuje model implementovaný pomocí rozhraní *ITrackableEntity*, díky kterému je možné nastavit vlastnosti. Čas je ukládán ve formátu UTC (Coordinated Universal Time), a proto musí být před vypsáním převeden na lokální čas.

```
protected virtual TModel PridejDatumVytvoreni(TModel entita)
   where TModel : ITrackableEntity {
    entita.DatumVytvoreni = DateTime.UtcNow;
    entita.DatumUpravy = DateTime.UtcNow;
    ApplicationUser uzivatel =
       _userManager.FindByNameAsync(User.Identity.Name).Result;
    entita.VytvorilId = uzivatel.VybranyZamestnanec;
    entita.UpravilId = uzivatel.VybranyZamestnanec;
    return entita;
}

protected virtual TModel PridejDatumZmeny(TModel entita) where
   TModel : ITrackableEntity {
    entita.DatumVytvoreni = entita.DatumVytvoreni;
    entita.DatumUpravy = DateTime.UtcNow;
    ApplicationUser uzivatel =
       _userManager.FindByNameAsync(User.Identity.Name).Result;
    entita.VytvorilId = entita.VytvorilId;
    entita.UpravilId = uzivatel.VybranyZamestnanec;
    return entita;
}
```

Validace vstupu

Každý formulář je validován z hlediska správnosti formátů a vyplnění všech povinných vstupů. Při zadávání hodnot, jako je e-mailová adresa či heslo, bylo využito validace, kterou nabízí již implementovaná validační knihovna ASP .NET. Naopak u vstupů, jako je poštovní směrovací číslo či IČO a DIČ, bylo zapotřebí vytvořit vlastní regulární výrazy, které zaručují správnou kontrolu vstupů.



Obrázek 16: Validace vstupu

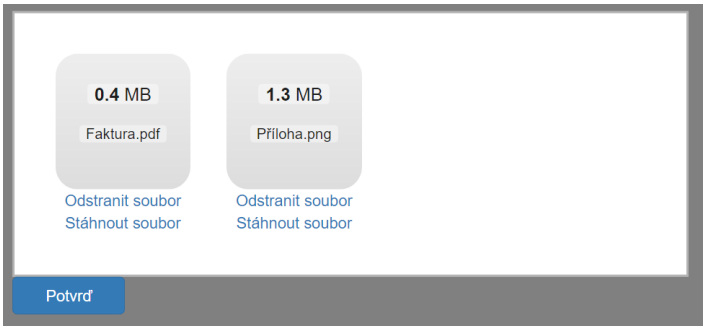
Taktéž byla ošetřena kontrola duplicitních názvů u skladových karet, kdy nesmí být vytvořeny skladové karty se stejným identifikačním jménem. U zaměstnanců musí být zajištěna unikátnost e-mailových adres. Při jakékoliv chybě ve validaci je uživatel upozorněn chybovou hláškou, která je zobrazena u vstupního boxu.

Přidávání faktur

Do informačního systému lze vkládat faktury či jiné dokumenty, které jsou takto elektronicky zálohovány. V návrhu je tento proces zaznačen jako *Přidat soubor*. Přidávání faktur je řešeno pomocí dvou způsobů.

První možností je přidání samostatného produktu, k němuž může být přiřazen pouze jediný soubor v různém formátu. Pokud by v tomto případě bylo zapotřebí vložit více souborů, je umožněno vložení archivu s příponou .zip či .rar.

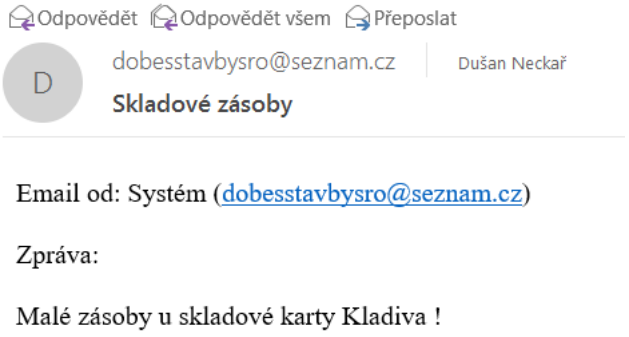
Druhou možností je přidání produktu pomocí dodacího listu, ve kterém je dovoleno přiřadit jednomu dodacímu listu více faktur. Kvůli uživatelsky přívětivějšímu prostředí byla využita komponenta dropzoneJS, která dovoluje nahrávat soubory pouhým přetažením neboli drag and drop do vyhrazeného prostoru. Případně je možné na tuto plochu kliknout a zobrazí se okno průzkumníka, pomocí kterého je soubor nalezen a vybrán. U této metody je možné vybrat více souborů najednou, které jsou posléze uloženy do databáze.



Obrázek 17: Přidávání faktur

Odesílání e-mailů

Odesílání e-mailů může nastat ve více situacích, a to v případě *Přidat záznam do skladové karty* a při *Odebrat záznam ze skladové karty*, jak lze vidět z diagramu případu užití, který je obsažen v příloze D.



Obrázek 18: E-mail

Po přidání záznamů s produkty jsou přiřazeny do určitých skladových karet, kdy proběhne kontrola množství. Při nízkém množství se odešle notifikační e-mail, který upozorňuje na snížení skladových zásob v konkrétní skladové kartě. Obdobná zpráva se odešle při vysokém množství zásob, pouze s rozdílným upozorněním. Tento proces je v diagramu případu užití zaznačen jako *Zkontrolovat skladové zásoby*.

Tato funkcionalita byla vytvořena za pomoci třídy MailMessage, která je obsažena v knihovně tříd .NET Framework. Třída byla využita k bezproblémovému odesílání e-mailových zpráv pomocí SMTP (Simple Mail Transfer Protocol) klienta. Po vytvoření instance této třídy a předání vstupních parametrů do vlastností, musel být vytvořen objekt třídy SmtpClient. Tomuto objektu byl přiřazen SMTP host a port na Seznam.cz.

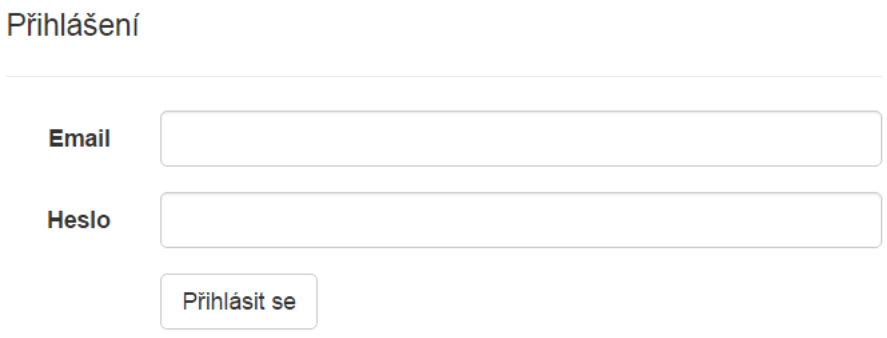
```
protected void OdesliEmail(string zprava, string odEmail,
    string odJmeno, string komuEmail, string predmet,
    string heslo, HttpPostedFileBase fileUploader) {
      var body = "Email od: {0} ({1})Zpráva:{2}";
      var message = new MailMessage();
      message.To.Add(new MailAddress(komuEmail));
      message.From = new MailAddress(odEmail);
      message.Subject = predmet;
      zprava = IsNullOrEmpty(zprava) ? "0" : zprava;
      message.Body = Format(body, odJmeno, odEmail, zprava);
      message.IsBodyHtml = true;
      if (fileUploader != null) {
        string fileName =
           Path.GetFileName(fileUploader.FileName);
```

```
        message.Attachments.Add(new
            Attachment(fileUploader.InputStream, fileName));
      }
      try {
        using (var smtp = new SmtpClient()) {
          var credential = new NetworkCredential {
            UserName = odEmail,
            Password = heslo
          };
          smtp.Credentials = credential;
          smtp.Host = "smtp.seznam.cz";
          smtp.Port = 587;
          smtp.EnableSsl = true;
          smtp.Send(message);
        }
      } catch (Exception e) {
        Danger("Chyba internetu: "+ e);
    }
}
```

Přihlášení

Přihlášení uživatele je vstupním rozhraním do aplikace, které zajišťuje autentizaci a autorizaci. V návrhu je tento případ užití pojmenován jako *Přihlásit se*.

Uživatel se přihlašuje pod zaregistrovaným e-mailem, který musí být unikátní v celém systému, a jeho heslem, které musí být tvořeno šesti znaky, z nichž minimálně jeden znak musí být velké písmeno a taktéž musí být použit znak speciální. Při nedostatečném počtu znaků či neobsažení požadovaného znaku je zobrazeno chybové upozornění a vyžádáno opětovné zadání.



Obrázek 19: Přihlášení

Odhlášení

Opakem přihlášení je odhlášení, kdy návrh obsahuje případ užití *Odhlásit se*. Tato akce nastává pří kliknutí na tlačítko odhlásit se nebo při změně osobních zásadních údajů, jako je e-mailová adresa či heslo. Při této akci je zapotřebí vymazat a znovu načíst soubory cookies.

9.3 Funkcionalita systému z pohledu *Vedoucího*

Role *Vedoucí* má na starosti administraci zaměstnanců, subjektů a skladových karet. Tyto administrativní činnosti jsou blíže rozepsány a představeny.

Vytvořit a registrovat nového zaměstnance

Vytvoření nového zaměstnance je provedeno v sekci *Zaměstnanci*, pod tlačítkem *Přidat nového zaměstnance*. Poté je umožněno zadání uživatelských údajů, kdy je potřeba vyplnit jméno a jeho roli, případně e-mail a poznámku. E-mailová adresa musí být v systému jedinečná, a proto je před uložením provedena kontrola duplicity. Tímto dojde k uložení nového zaměstnance do databáze.

Pokud je u zaměstnance k dispozici jeho e-mailová adresa, je možné mu vytvořit i účet do systému. To je provedeno pomocí registrace, ve které je vybrán konkrétní zaměstnanec a přiřazeno heslo, které si uživatel po přihlášení může změnit podle povinně stanoveného formátu.

Vedoucí má možnost evidovat do systému i zaměstnance, kteří nebudou mít přidělený přístup do informačního systému kvůli jeho nevyužití či úmyslnému zamezení. Tento přístup je vhodný při budoucí změně nebo přidání rolí, například při povýšení zaměstnance. Registrace je v diagramu případu užití uvedena jako *Registrovat zaměstnance*.

Editovat zaměstnance

Při výpisu všech zaměstnanců je u každého z nich k dispozici tlačítko *Upravit*, po kterém je umožněna úprava uživatelských údajů. Při změně e-mailové adresy je taktéž provedena kontrola duplicit. Ve výpisu jsou zobrazeny pouze údaje jméno a příjmení a pro zobrazení detailních informací je k dispozici tlačítko *Detail*.

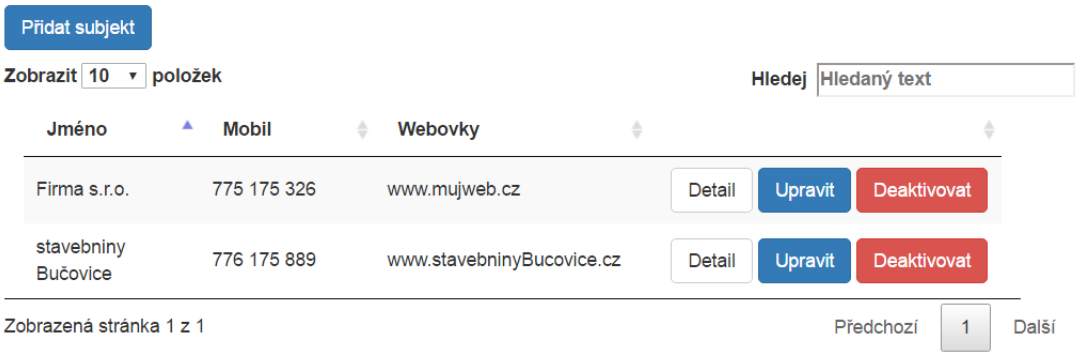
Deaktivovat zaměstnance

Při zobrazení zaměstnanců je možno provést jejich deaktivaci, při které jsou odebrány autentizační údaje. Ovšem veškeré zaměstnancovy údaje jsou ponechány v databázi pro pozdější dohledání a pro výpis modifikací, které v systému provedl.

Vytvořit subjekt

U vytváření subjektu se zadávají pouze základní nutné údaje, kdy je možné zvolit, zda se jedná o fyzickou nebo právnickou osobu. U fyzické osoby se zadává název nebo jméno a taktéž lze vyplnit město. U právnické osoby se vyplňují stejné informace jako u fyzické s pouhým rozšířením o IČO a DIČ.

Po vytvoření subjektu lze dodat další doplňující údaje, kterými jsou kontaktní a adresní údaje a případná poznámka.



Obrázek 20: Seznam subjektů

Deaktivovat subjekt

Při zobrazení subjektů je možné subjekt deaktivovat, čímž dojde k zamezení jeho dalšího využívání a manipulace. Stejně jako deaktivovaný zaměstnanec, tak i subjekt je po deaktivaci dále ponechán v databázi pro pozdější zpětné dohledání.

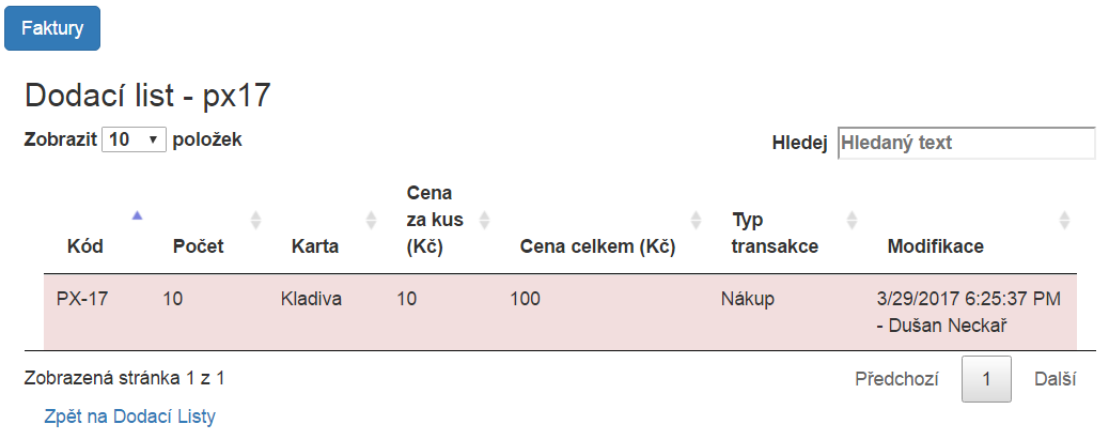
Vytvořit a editovat skladovou kartu

Skladové karty představují jednotlivé kategorie produktů, do kterých jsou *Skladníkem* přidávány položky z dodacích listů. Tvorba je možná v sekci Skladové karty, po kliknutí na tlačítko *Přidat skladovou kartu*, kdy je nutné vyplnit název karty, maximální možný počet naskladnění a měrnou jednotku.

Po zhotovení je možné údaje skladové karty upravit přes tlačítko *Upravit* ve výpisu všech skladových karet, případně celou skladovou kartu odstranit.

Zobrazit detail dodacího listu

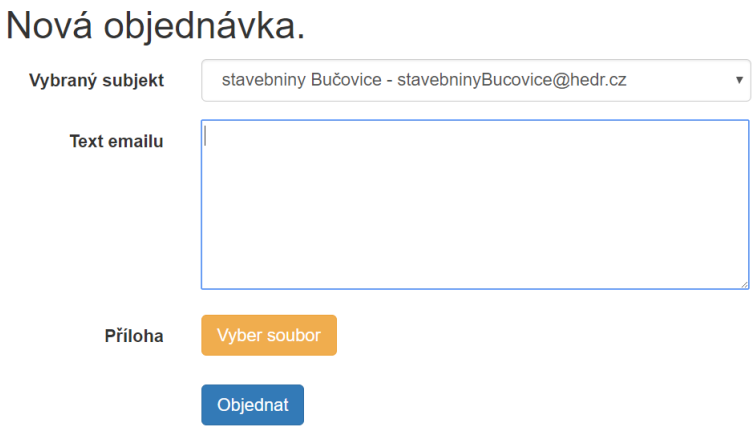
Pro přehlednější výpis položek dodacího listu slouží *Detail dodacího listu*. Ten zobrazuje rozšířené informace o položkách, což je specifický kód položky, typ skladové karty, počet a cena za kus. Tato funkcionalita platí i pro *Skladníka*. V detailu konkrétního dodacího listu je umožněno vkládání faktur způsobem popsaným na straně 41.



Obrázek 21: Detail dodacího listu

Objednat zboží

V případě úbytku skladových zásob je *Vedoucímu*, případně *Skladníkovi*, umožněna objednávka zboží. Ta je provedena po přesměrování z výpisu skladových karet a zobrazení formuláře, ve kterém je možno vybrat konkrétní subjekt s uvedením jeho e-mailové adresy a okno pro napsání zprávy. K objednávce lze přiložit i přílohu.



Obrázek 22: Objednávka zboží

9.4 Funkcionalita systému z pohledu *Skladníka*

Přidat záznam do skladové karty

Pokud je již skladová karta zhotovena od *Vedoucího*, je možné do ní přidávat produkty v podobě záznamů, které mohou být dvou typů. Prvním typem je naskladnění produktů do skladu a druhým typem je jejich vyskladnění.

Každá tato akce je kontrolována proti neplatnému zásahu, jako je odebrání produktu ze skladové karty, který již není dostupná. Při přidávání je kontrolováno, zda zadávaná hodnota nepřesahuje maximální povolenou kapacitu skladu, která byla předem určena *Vedoucím*. Tuto hodnotu je možné přesáhnout maximálně o dvacet procent.

Pokud by došlo k převýšení či nedostatečnému stavu skladu, je odeslán informativní e-mail *Vedoucím* podniku. Záznamy lze editovat se stejnou funkčností.

Odebrat záznam ze skladové karty

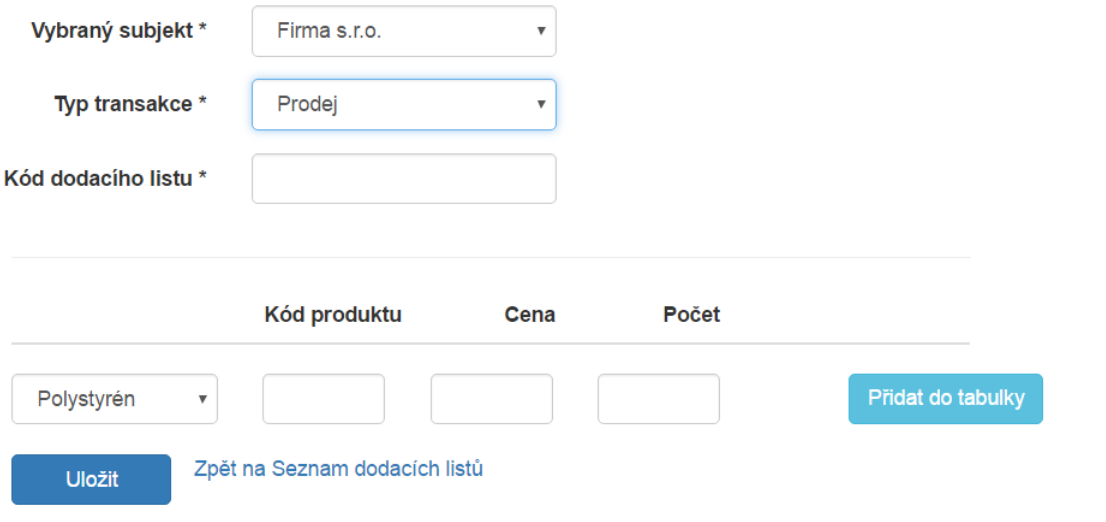
Při odebírání záznamů ze skladových karet je prováděna validace, ověřující stav skladových zásob. Tato kontrola ověřuje stav skladových zásob, zda při odebírání nedošlo k nekorektním nebo záporným hodnotám zásob.

Vymazat dodací list

Tato operace vymaže kompletně celý dodací list i s obsahujícími skladovými zásobami, kdy je opět zanesena kontrola správnosti.

Vytvořit dodací list

Za předpokladu, že je *Vedoucím* vytvořena alespoň jedna skladová karta a minimálně jeden subjekt lze vytvořit dodací list. Po vybrání subjektu a vložení jedinečného identifikačního kódu, lze přidávat importované či exportované zboží pomocí jednotlivých položek dodacího listu.



Obrázek 23: Vytvoření dodacího listu

Pro snadnější manipulaci je zaměstnanec po prvotní editaci přesměrován na detail dodacího listu, kde může po stisku na tlačítko *Faktury* přidat fakturu k vytvářenému dodacímu listu. Limit faktur je prozatímní a je možno vložit maximálně čtyři faktury. Nahrané faktury je možné stahovat a mazat, kdy tuto možnost má i *Vedoucí*.

Je počítáno i s variantou, že by zaměstnanec při vytváření zadal nesprávné údaje, a proto je možné vytvořený dodací list editovat. Také zde je zajištěna kontrola správnosti, aby nemohlo dojít k zadání nesprávných hodnot či údajů.

10 Testování

Po ukončení implementace bylo potřeba aplikaci řádně otestovat. Cílem bylo odhalit veškeré kritické i méně závažné chyby. Kritickými chybami jsou myšleny ty, které by znepřístupňovaly nebo jakýmkoliv způsobem ohrožovaly chod systému.

Testování bylo provedeno na straně vývojáře, kdy bylo k ověření všech funkcí a služeb využito fiktivně vytvořených přihlašovacích účtů, jež jsou uvedeny v tabulce 2. Bylo využito dostupných internetových prohlížečů, jako je Internet Explorer verze 11 či Google Chrome verze 57.0.2987.110 (64 bit).

Nejdříve bylo k testování využito lokální databáze a lokálního serveru, kvůli snadnější manipulaci a testování dat. V pozdější fázi testování byla použita serverová databáze a produkční server, na který byla přidělena doména *http://automatizovanysklad.hedr.cz*.

Tabulka 2: Testovací údaje

Role	**Přihlašovací jméno**	**Heslo**
Vedoucí	dobeststavbysro@seznam.cz	Heslo123*
Skladník	dusan.neckar@hedr.cz	Heslo123*

Při testování bylo nalezeno velké množství nedostatků v implementaci. Mezi nejkritičtější chybu patřilo chybné zapisování získaných dat do databáze a jejich následné mazání. Dále byly odhaleny aplikační nedostatky, které byly při vývoji zanedbány. Každá objevená chyba byla řádně opravena a znovu otestována.

Dalším krokem zahrnutým v testování bylo vytvoření krátkého návodu k seznámení systému, který byl předán pověřenému vedoucímu cílového podniku. Ten poté sám aplikaci vyzkoušel a ohodnotil míru uživatelské přívětivosti a intuitivního ovládání.

Po konzultaci s vedením bylo dohodnuto, že každý stávající i nový zaměstnanec by měl projít základním školením objasňujícím práci v novém informačním systému.

10.1 Uniscan

K otestování aplikace byl využit nástroj Uniscan, který testuje odolnost webových aplikací vůči různým typům útoků. Mezi tyto typy patří SQL injection, Cross-Site Scripting (XSS) a mnoho dalších. Nástroj byl spuštěn na operačním systému speciálně určeném pro penetrační testování s názvem Kali linux verze 2016.2.

Výsledek testu dopadl zdařile, neboť žádný provedený útok nebyl úspěšný, a nebyla tak prolomena ani ohrožena bezpečnost testované webové aplikace. Výpis z nástroje po ukončení testu je zobrazen na obrázku 24.

Obrázek 24: Uniscan testování

11 Diskuze a závěr

V bakalářské práci byla rozebrána problematika vytváření informačního systému pro podnik, který se zaobírá prodejem stavebního materiálu. Cílem bylo vytvořit modul řízeného skladu, který by ulehčil zaměstnancům a vedoucím práci ve skladu. Na základě získaných požadavků byla dohodnuta tvorba systému formou webové aplikace.

V úvodní části byly objasněny pojmy spojené s informačními systémy, základní architektury, přístupy k analýze a životní cyklus vývoje. Posléze byly popsány technologie použité při implementaci modulu pro řízený sklad. Vybranou technologií pro vyhotovení modulu byl ASP .NET s architekturou MVC a pro podporu dynamiky aplikace byl použit programovací jazyk JavaScript a jQuery. Hlavním důvodem výběru této technologie byla snadná budoucí rozšířitelnost zhotovené webové aplikace do mobilní či desktopové verze a jednoduchá konfigurace podnikového serveru s vytvořenou aplikací.

V další části byly sepsány zákazníkovi požadavky, na základě kterých byl proveden návrh a namodelovány diagramy sloužící pro podporu implementace. Taktéž byl navržen drátěný model pro vizuální představu budoucího systému, který byl konzultován se zákazníkem. Podle namodelovaného diagramu tříd byly vytvořeny objekty, které v architektuře MVC představují modely. Poté byly propojeny s kontroléry obstarávající komunikaci mezi modely a pohledy. Do systému je povolen vstup pouze autorizovaným uživatelům, kteří jsou rozděleny na dvě role, a to na vedoucí a skladník. Tyto dvě role mají odlišnou funkcionalitu a každý má v systému povolené své procesy.

Systém byl proveden s automatickou kontrolou skladových zásob, kdy při každé akci je kontrolován stav zásob a popřípadě odeslána systémová notifikace na e-mail vedoucích podniku. Po implementaci aplikace bylo provedeno ruční testování, které odhalilo chyby, jež byly následně opraveny. Dále bylo provedeno testování pomocí nástroje Uniscan, který slouží k odhalování rizik a testuje existující webové útoky. Z výsledků bylo patrné, že při testování zhotoveného řízeného skladu nebyla nalezena žádná hrozba a systém je plně zabezpečen.

V případě potřeby je možné aplikaci v budoucnu snadno rozšiřovat či upravovat podle dalších přání klienta a zdokonalovat tak chod celého podniku. Typy pro zdokonalení, o které by aplikace mohla být vylepšena jsou:

- Filtrování produktů či dodacích listů dle vymezených kritérií.
- Evidence zboží pomocí čtečky čárových kódů.
- Rozšíření o další doplňkové moduly a jejich vzájemné propojení.
- Export dat do různých formátů pro snazší převod do jiných aplikací (.xml) nebo kvůli zachování struktury výsledného dokumentu (.pdf).
- Vytvoření mapy skladu a zobrazování naskladněných produktů.

- Vytváření reportů a dashboardů ze získaných dat.

Po předání vytvořeného řízeného skladu zadavateli DOBEŠ-STAVBY s. r. o. byla taktéž navrhnuta tato vylepšení, o které by podnik v případě rozšíření skladových prostor měl zájem. Jedná se zejména o vytvoření přehledné mapy skladu, která by zobrazovala schéma fyzického umístění skladových zásob a obsazenost částí. Dynamická správa skladu by byla zajištěna i s pomocí čteček čárových kódů, které by při zvětšení skladu i zásob značně usnadnily a urychlily práci zaměstnanců.

V případě implementace mapy skladu by muselo dojít ke konzultaci s vedením podniku a získání půdorysu, podle kterého by bylo možné vytvořit grafické schéma skladu. Tato mapa by zobrazovala zboží naskladněné ve skladu a informovala o obsazenosti jednotlivých skladových částí. Pokud by firma chtěla aplikovat i čtečky čárových kódů, muselo by z hlediska aplikace dojít ke značným úpravám. Musel by být vybrán a aplikován framework určený přímo pro práci s těmito čtečkami, se kterým by aplikace pracovala.

Závěrem práce lze říci, že vyhotovený řízený sklad byl sestaven podle požadavků zákazníka a taktéž byl splněn stanovený cíl práce.

12 Reference

Abra Software. *Skladové hospodářství* [online]. Praha: ABRA Software, © 2017 [cit. 2017-04-03]. Dostupné z: https://www.abra.eu/informacni-systemy/moduly/skladove-hospodarstvi/.

Altus software. *Altus Vario* [online]. Praha: Altus software, © 2017 [cit. 2017-04-03]. Dostupné z: http://www.vario.cz/moduly/sklad/.

Arlow, J. a I. Neustadt. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky.* 2., aktualiz. a dopl. vyd. Brno: Computer Press, © 2007. ISBN 978-80-251-1503-9.

Bruckner, T. a kol. *Tvorba informačních systémů: Principy, metodiky, architektury.* Praha: Grada Publishing, © 2012. ISBN 978-80-247-4153-6.

CCV Informační systémy. *Instantní sklad* [online]. Brno: CCV Informační systémy, © 2017 [cit. 2017-04-03]. Dostupné z: https://www.ccv.cz/instantni-sklad/.

Dajbych, V. *K čemu je dobrý TypeScript.* In: Zdroják.cz [online]. Praha: Devel.cz Lab, 2013 [cit. 2017-04-04]. ISSN 1803-5620. Dostupné z: https://www.zdrojak.cz/clanky/k-cemu-je-dobry-typescript/.

Dickerson, C. E. a D. N. Mavris *Architecture and principles of systems engineering.* Boca Raton: CRC Press, © 2010. ISBN 978-1-4200-47253-2.

DOBEŠ-STAVBY s. r. o. *Informační system.* Vyškov: DOBEŠ-STAVBY s. r. o., 1995–2017.

Dynamics Online. *MS Dynamics 365* [online]. Praha: WEBCOM, © 2016 [cit. 2017-04-03]. Dostupné z: http://www.msdynamics365.cz/.

Entity Framework Tutorial. *What is Entity Framework?* [online]. EntityFrameworkTutorial.net, © 2016 [cit. 2017-04-04]. Dostupné z: http://www.entityframeworktutorial.net/what-is-entityframework.aspx.

Flanagan, D. *JavaScript: The Definitive Guide.* 6th ed. Sebastopol, CA: O'Reilly, © 2011. ISBN 978-0-596-80552-4.

Flanagan, D. *jQuery Pocket Reference.* CA: O'Reilly, © 2011. ISBN 978-144-9397-227.

Freeman, A. *Pro ASP.NET MVC 5 Platform.* New York, USA: Springer, © 2014. ISBN 978-1-4302-6542-9.

Gála, L. *Podniková informatika: počítačové aplikace v podnikové a mezipodnikové praxi.* 3. vyd. Praha: Grada, © 2014. ISBN 978-80-247-5457-4.

KOSSIAKOFF, A. A KOL. *Systems Engineering: Principles and Practice*. 2nd ed. Hoboken: New Jersye: Wiley, © 2011. ISBN 978-0-470-40548-2.

KOŠUT, V. *Nejčastější problémy v řízení skladového hospodářství.* Digitovárna.cz [online]. Praha: MM Průmyslové spektrum, 2016 [cit. 2017-04-04]. Dostupné z: http://www.digitovarna.cz/clanek-84/nejcastejsi-problemy-v-rizeni-skladoveho-hospodarstvi.html.

MICROSOFT DEVELOPER NETWORK. *Introduction to the C# Language and the .NET Framework.* Microsoft Developer Network [online]. Redmond (Washington): Microsoft, 2015 [cit. 2017-04-04]. Dostupné z: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx.

MICROSOFT TECHNET. *Chapter 1: Introduction to .NET.* [online]. Redmond (Washington): Microsoft, 2006 [cit. 2017-04-04]. Dostupné z: https://technet.microsoft.com/en-us/library/bb496996.aspx.

MUNRO, J. *ASP.NET MVC 5 with Bootstrap and Knockout.js: Building Dynamic, Responsive Web Applications*. USA: O'Reilly, © 2015. ISBN 978-149-1914-397.

NASH, T. *C# 2010: rychlý průvodce novinkami a nejlepšími postupy*. Brno: Computer Press, © 2010. ISBN 978-80-251-3034-6.

POHODA JAZZ. *Účetní program POHODA* [online]. Jihlava: STORMWARE, © 2017 [cit. 2017-03-28]. Dostupné z: https://www.stormware.cz/pohoda/jazz.aspx.

RÁBOVÁ, I. *Podnikové informační systémy a technologie jejich vývoje*. 1. vyd. Brno: Tribun EU, 2008. ISBN 978-80-7399-599-7.

SODOMKA, P. A H. KLČOVÁ. *Informační systémy v podnikové praxi.* 2. akt. a roz. vyd. Brno: Computer Press, © 2010. ISBN 978-80-251-2878-7.

SOMMERVILLE, I. *Software engineering*. 9. vyd. Boston: Pearson, © 2011. ISBN 978-0-13-703515-1.

TWITTER *Bootstrap* [online]. Twitter, © 2011–2017 [cit. 2017-04-04]. Dostupné z: http://getbootstrap.com/.

VYMĚTAL, D. *Informační systémy v podnicích: teorie a praxe projektování*. Praha: Grada, © 2009. ISBN 978-80-247-3046-2.

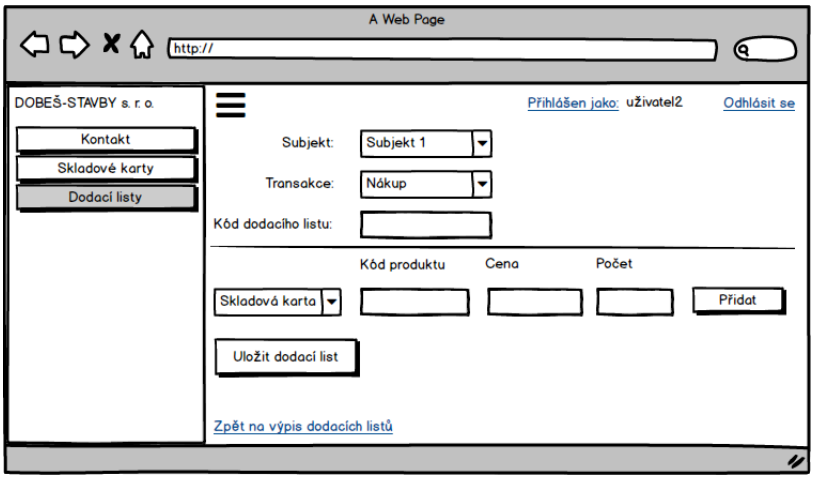
W3SCHOOLS *ASP.NET Razor - C# and VB Code Syntax* [online]. Refsnes Data, © 1999-2017 [cit. 2017-04-04]. Dostupné z: https://www.w3schools.com/asp/razor_syntax.asp.

Přílohy

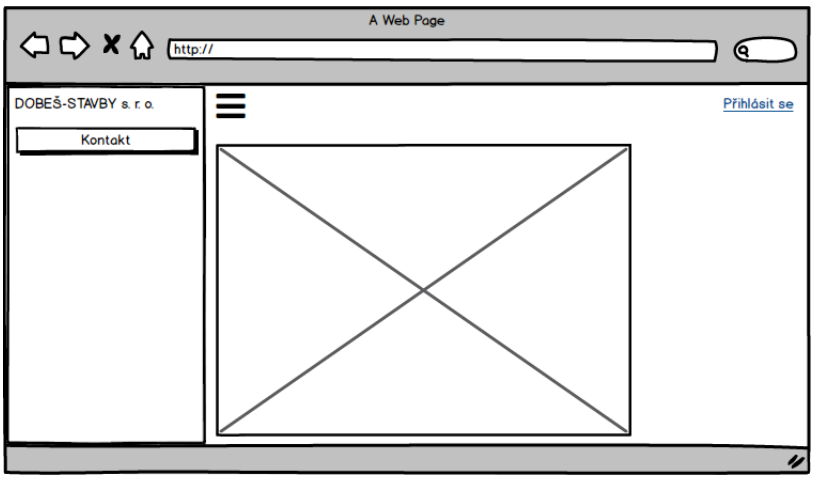
A Přiložené CD

Přiložené CD obsahuje výslednou aplikaci se zdrojovými kódy, které se nacházejí ve složce *Web*. Dále jsou vytvořené diagramy, využité v práci přiloženy ve složce *Diagramy*.

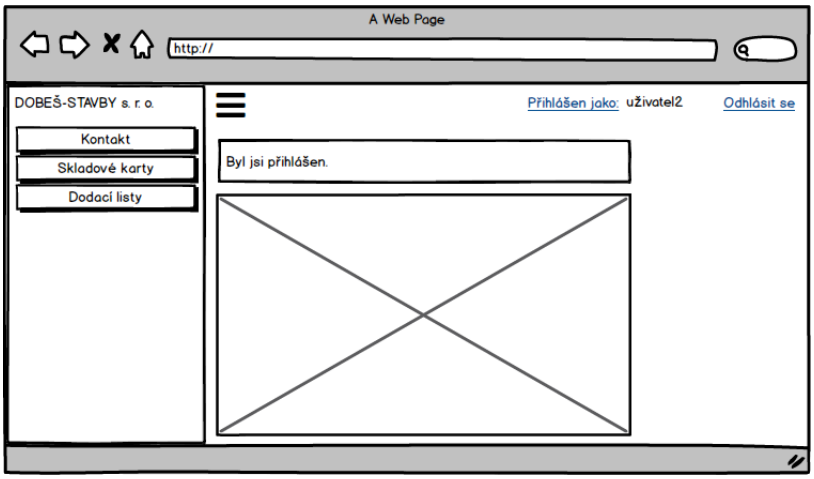
B Drátěný model



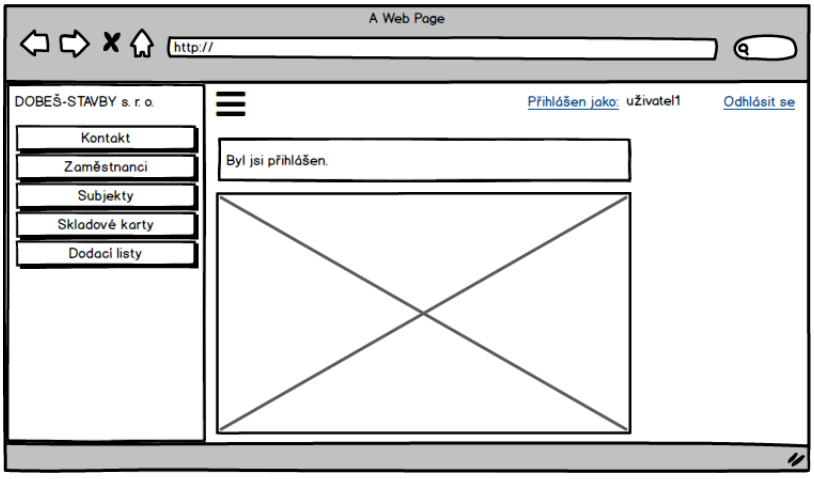
Obrázek 25: Drátěný model pro *Zobrazit dodací listy*



Obrázek 26: Drátěný model pro *Nepřihlášeného zaměstnance*

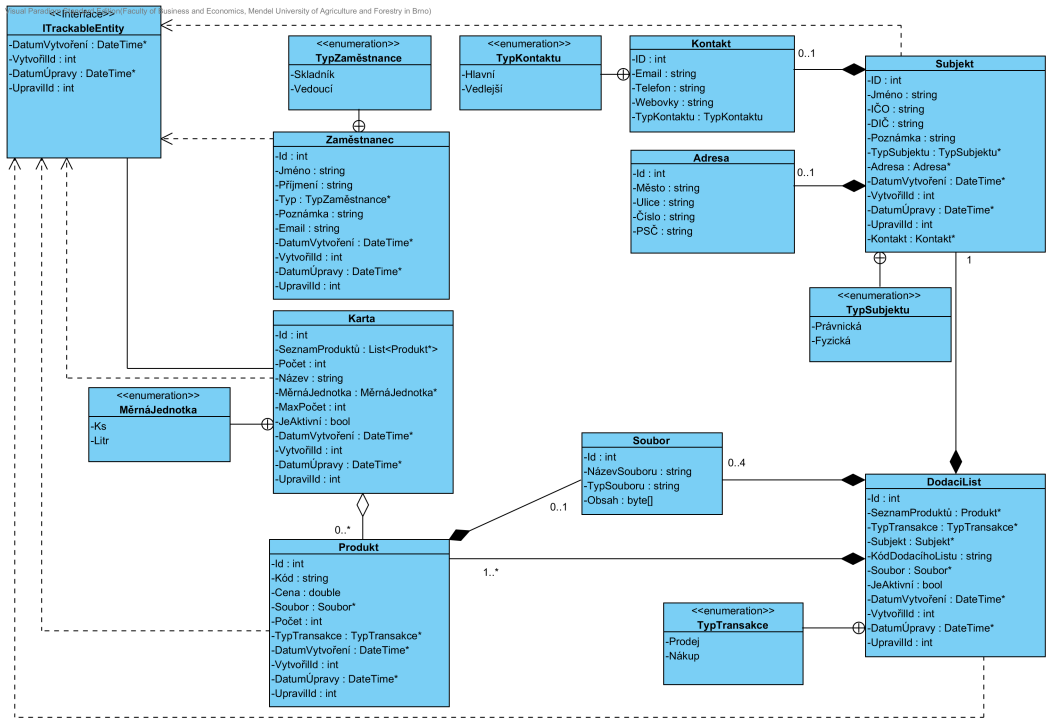


Obrázek 27: Drátěný model pro zaměstnance *Skladník*



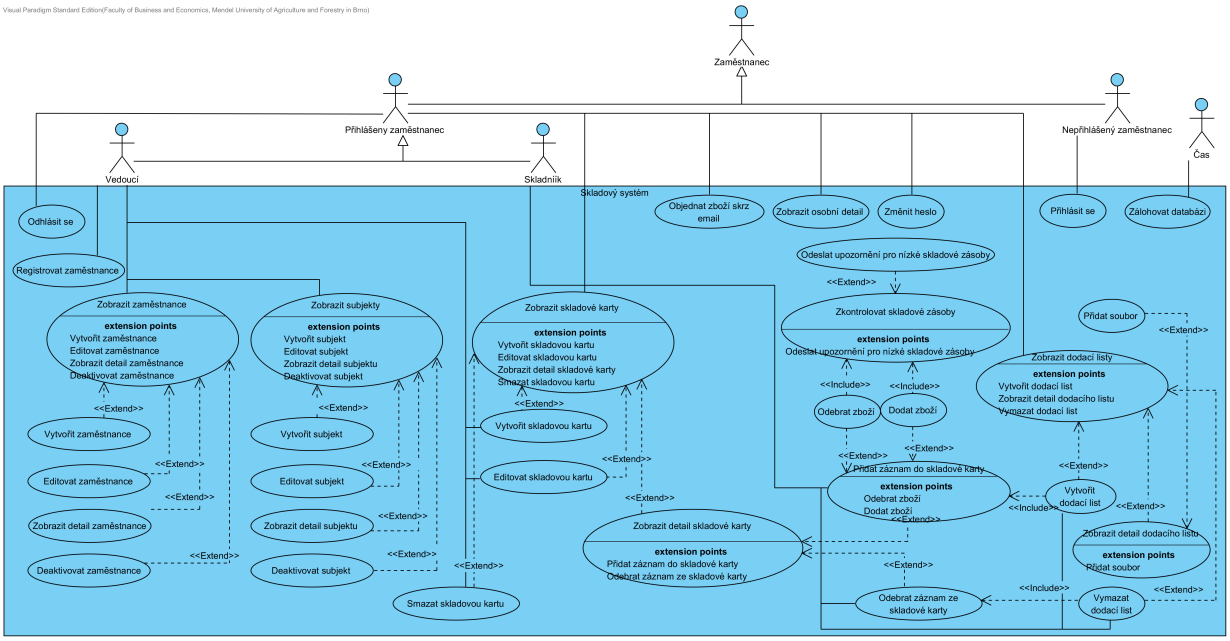
Obrázek 28: Drátěný model pro zaměstnance *Vedoucí*

C Diagram tříd



Obrázek 29: Diagram tříd

D Diagram případu užití



Obrázek 30: Diagram případu užití