Mendelova univerzita v Brně
Provozně ekonomická fakulta

Automatizace importu příjemek do ES Pohoda

Diplomová práce

Vedoucí práce:

Ing. Jan Přichystal, Ph.D. **Bc. Ondřej Charvát**

Brno 2017

Tímto bych chtěl poděkovat vedoucímu diplomové práce Ing. Janu Přichystalovi, Ph.D. za ochotu, cenné rady a odborný dohled. Rovněž bych chtěl poděkovat vedení a zaměstnancům firmy ASSEC Computers, za vstřícnost a ochotu.

Čestné prohlášení

Prohlašuji, že jsem tuto práci Automatizace importu příjemek do ES Pohoda vypracoval/a 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/a, 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 2. ledna 2017 _____

Abstract

Charvát, O. The automation import of receipt cards to ES Pohoda. Diploma thesis. Brno: Mendel University, 2017.
The diploma thesis deals with design and implementation of application to automation import to ES Pohoda. In the theoretical part is introduced automation, economic system Pohoda and company for which the solution was processed. The thesis also analyses the current state of the issue within the company and outside.
The practical part of the thesis deals with requirements analysis, design, implementation and testing of the application. In this part are closer introduced
the technologies used in the development. In the conclusion of the thesis is application critically assessed and possible improvements are suggested.

Keywords

Automation, business processes, ASSEC Computers, ES Pohoda, requirements analysis, Visual Basic, .NET, application.

Abstrakt

Charvát, O. Automatizace importu příjemek do ES Pohoda. Diplomová práce. Brno: Mendelova univerzita v Brně, 2017.
Diplomová práce se zabývá návrhem a implementací aplikace pro automatizaci importu příjemek do ES Pohoda. V teoretické části práce je představena automatizace, ekonomický systém Pohoda a firma, pro kterou bylo řešení zpracováno. Dále práce analyzuje současný stav problematiky ve firmě i mimo ni. Praktická část práce se zabývá analýzou požadavků, návrhem, implementací a testováním aplikace. V této části jsou blíže představeny technologie, použité při vývoji aplikace. V závěru práce je aplikace kriticky zhodnocena a jsou navržena možná vylepšení.

Klíčová slova

Automatizace, podnikové procesy, ASSEC Computers, ES Pohoda, analýza požadavků, Visual Basic, .NET, aplikace.

Obsah

1 Úvod a cíl práce

1.1 Úvod

Člověk se vždy od ostatních tvorů, žijících na Zemi, vyznačoval „něčím navíc“. Byl to tvor obdařený rozumem a schopností myslet. To je nejspíše důvod, proč lidé již od svých počátků vymýšleli věci a postupy, které jim přinášely „něco navíc“, a pomohly jim uplatnit svoji inteligenci v praxi. Z počátku se jednalo o věci, které lidem zajišťovaly základní potřeby, jako bylo bezpečí, potrava atd. S rozvojem společnosti však nové vynálezy začaly přinášet i jiný užitek, jako bylo pohodlí, vojenská převaha atd.

Současná lidská civilizace je typická neustálou snahou o zrychlování. Tuto snahu můžeme pozorovat na každém kroku, ať už se jedná o neustálý vývoj výpočetní techniky, který umožní zkrácení doby výpočetních operací nebo snaha o zkrácení doby přepravy z jednoho místa na druhé, pomocí rychlejších dopravních prostředků. Účinným nástrojem toho trendu je také Internet, díky kterému je šíření informací opravdu snadné a který nám umožňuje provádět spoustu činností z pohodlí domova.

Za vynález člověka se dá rovněž považovat automatizace. Již v pravěku lidé využívali různé automatizované procesy, jako byla sterilizace, zpracování kovů atd., tehdy se však jednalo převážně o procesy přírodní. Automatizace prošla různými etapami vývoje, až do dnešní podoby. Podstatným aspektem automatizace dnešní doby jsou počítače. Počítač, je jednoduše řečeno zařízení, které zpracovává data, podle předem vytvořeného programu. Počítač tedy provádí něco, co dříve musel provádět člověk sám, automatizuje jeho činnost. Člověku však nestačí, že za něj počítač provádí složité výpočty. Lidé často touží po tom, aby za ně počítač i přemýšlel. Proto vzniká řada řešení, které simulují umělou inteligenci.

O oblíbenosti automatizace svědčí i fakt, že neustále vzniká celá řada nových řešení, jež automatizaci provádí. Ať už se jedná o jednoduché aplikace, které za člověka v určitý čas vypnou počítač nebo o komplexní řešení, která dokáží řídit celý složitý systém. Důkazem oblíbenosti automatizace je bezesporu i neustála snaha o vytvoření dokonalé umělé inteligence a robota, který bude co nejvěrněji imitovat chování člověka.

Jedním z důvodů, proč je automatizace ve firmách zaváděna, je bezesporu i fakt, že se firmy chtějí soustředit především na jejich klíčovou aktivitu, tedy činnost, která jim generuje zisk. Automatizace ve firmě často provádí aktivity, které zisk sice negenerují, ale jsou pro její chod rovněž důležité. Lidé, kteří tuto činnost dříve prováděli, se pak mohou soustředit na jiné činnosti, které mají pro firmu větší přínos nebo takové, které automatizovat nelze.

1.2 Cíl práce

Hlavním cílem diplomové práce je navrhnout a implementovat aplikaci, která bude ve firmě ASSEC Computers automatizovat import příjemek do ekonomického systému Pohoda. Aplikace by měla především omezit počet chyb při importu, generovaných člověkem. Uživatelé budou mít díky automatizaci více času na ostatní činnosti.

Za dílčí cíle této práce lze označit analýzu současného stavu zpracovávané problematiky, analýzu požadavků na vytvářené řešení, návrh tohoto řešení a jeho implementaci. Následně je potřeba provést testování vytvořené aplikace, nasadit ji do provozu a zaškolit personál v jejím používání.

2 Metodika

Pro zpracování této práce, resp. vývoj aplikace, která je její součásti, jsem se rozhodl využít některé prvky metodiky Rapid Application Development neboli RAD. I přesto, že je tato metodika určena spíše pro vývoj ve skupině, tak myslím, že mi některé její aspekty pomohou s vývojem mojí aplikace. Ostatně, nenašel jsem žádnou metodiku pro vývoj softwaru, která by byla určena výhradně pro samostatného vývojáře.

Na dnešním, stále více konkurenčním trhu, přináší metodika RAD možnost efektivnější spolupráce zákazníka a vývojáře, a umožňuje tak rychlejší dodávání softwaru. Poptávka po nových aplikacích stále roste, avšak schopnost IT odvětví tyto požadavky absorbovat není dostačující. Podle výzkumu společnosti Mendix asi 70 % vývojářských týmů nestíhá termíny a není schopno uspokojit stále rostoucí požadavky trhu. Metodika RAD tak může být způsobem, jak tuto krizovou situaci řešit. (Rapid Application Development, 2016)

Metodika RAD je založena na vytváření prototypů a iterativním vývoji, bez složitého počátečního plánování. Prototyp je funkční model, který svou funkcionalitou odpovídá části kompletního systému. Díky nepřetržité integraci jsou redukovány problémy, které se jinak mohou objevit až při finálním nasazení vytvářeného řešení. (Gottesdiener Ellen, 2016)

Na rozdíl od vodopádového vývoje softwaru, který se někdy až puntičkářsky soustředí na dokonalou specifikaci a plánování, metodika RAD umožňuje vývoj se stále se měnícími požadavky. Nové požadavky jsou pak mnohdy získávány až v průběhu vývoje, kdy se zákazníkovi dostane do rukou prototyp výsledného řešení. Tradiční metodiky vývoje softwaru dodržují přísný model s velkým důrazem
na analýzu požadavků ještě před samotným psaním kódu. Tento přístup vytváří zbytečný tlak na zákazníka, který musí definovat požadavky na projekt, který vlastně ještě nezačal a na produkt, se kterým ještě dlouhou dobu nepřijde do styku. (SDLC - RAD Model, 2016)

Díky opakujícím se cyklům vývoje a rychlému dodání prototypů je zajištěna brzká zpětná vazba od zákazníka. Při každé iteraci tak mohou vývojáři ověřit a upřesnit požadavky zákazníka, které se mohou v průběhu vývoje měnit. Díky neustálému zapojení zákazníka v procesu vývoje se redukují problematická místa a zvyšuje se šance na úspěšné dodání finálního produktu. (Gottesdiener Ellen, 2016)

Kromě rychlého vývoje poskytuje metodika RAD ještě další výhody. Mezi ty nejdůležitější patří například tyto:

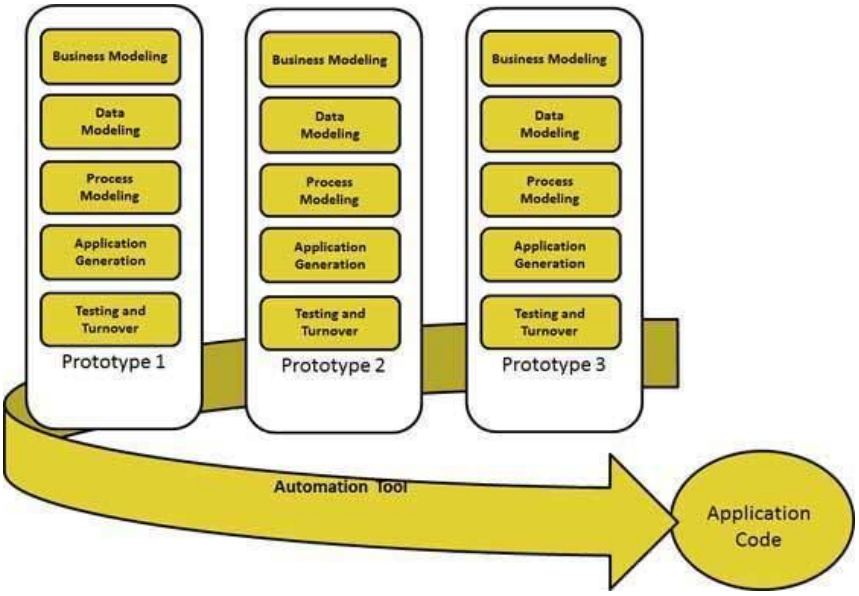
- Snížení rizika – schopnost rychle vytvořit a sdílet fungující prototyp, umožňuje zákazníkovi kontrolovat funkcionalitu již od počátku vývoje, což snižuje šanci na to, že bude nutné celou aplikaci razantně předělat

- Zvýšení kvality – využití prototypů a testování funkcionality během vývoje aplikace také zvyšuje kvalitu softwaru, jelikož mohou být na základě zpětné vazby zákazníka ověřovány a upřesňovány požadavky
- Rychlejší nasazení – rychlejší vytvoření funkční aplikace znamená, že zákazník může využívat výhod aplikace ihned, zatímco vývoj další funkcionality stále pokračuje (Rapid Application Development, 2016)

RAD model se skládá z analýzy, návrhu, tvorby a testování, prováděných v krátkých sériích iterativně. Metodiku RAD lze rozdělit do několika fází:

- Business modelování – identifikuje tok informací mezi různými podnikovými funkcemi
- Datové modelování – informace získané v business modelování jsou využity k definování datových objektů, jež jsou k podnikání potřeba
- Procesní modelování – datové objekty definované v datovém modelování jsou upraveny pro dosažení podnikových informací, díky kterým je dosaženo specifických podnikových cílů
- Tvorba aplikace – vývojové nástroje jsou použity pro převedení procesního modelu na kód a konkrétní systém
- Testování a opakování – v RAD je celkový testovací cyklus redukován a prototypy jsou testovány při každé iteraci (Definition of Rapid Application Development, 2016)

Vývoj pomocí metodiky RAD znázorňuje i následující schéma. Na začátku vývoje proběhne rychlá analýza a návrh řešení, aby měl vývojář na čem stavět. Následuje iterace, kdy je vyvíjena část aplikace, ta je následně představena zákazníkovi a na základě testování a jeho zpětné vazby jsou upraveny požadavky. Pokud je daná část systému v pořádku, je přidána, k již fungujícímu prototypu. Vývojář postupně zpracovává jednotlivé části, dokud není aplikace zcela hotova.



Obr. 1 Schéma metodiky RAD (Rapid Application Development Model, 2016)

Výhody metodiky Rapid Application Development shrnuje následující výčet:

- Zkracuje trvání vývoje
- Zvyšuje znuvupoužitelnost jednotlivých komponent
- Možnost rychlého počátečního zhodnocení
- Vybízí zákazníka ke zpětné vazbě
- Integrace od počátku vývoje redukuje problémy s koncovou integrací (Gottesdiener Ellen, 2016)

3 Teoretická východiska práce

Tato práce by nemohla vzniknout bez nějakého podnětu. Něčeho, co vnukne prvotní myšlenku na její zpracování. Když pominu podmínku nutnou pro úspěšné absolvování magisterského studia, tedy nutnost zpracovat diplomovou práci, byl hlavním podnětem, pro tvorbu této diplomové práce požadavek, vznesený mým vedoucím pracovníkem, v současném zaměstnání.

Požadavek zněl jasně, vytvořit nástroj, který bude automaticky zpracovávat faktury generované hlavním dodavatelem firmy a bude je ve formě příjemek importovat do ekonomického systému Pohoda, který firma využívá pro vedení účetnictví.

Jakmile jsem znal tento prvotní požadavek, vytvořil jsem si v hlavě hrubou představu jeho řešení. Budu muset vytvořit nějakou formu automatizace, která bude získávat data z webu dodavatele, upraví je do použitelné podoby a provede import do ES Pohoda. Jelikož se jedná o velice citlivá data, musí být zajištěna bezchybná funkcionalita a bezpečnost aplikace.

Ještě předtím, než jsem začal práci psát, musel jsem nastudovat, jakým způsobem jsou data na stránkách dodavatele uložena, jak je bude možné z jeho stránek získat a dostat do požadované podoby. Musel jsem taky zjistit, jakým způsobem je možné komunikovat s ES Pohoda nějakou automatizovanou cestou.

V průběhu psaní jsem potom nastudoval problematiku automatizace, musel jsem zjistit bližší informace o firmě Assec Computers a rovněž jsem se zabýval metodikou vývoje softwaru nazvanou RAD, jíž se budu při vývoji aplikace držet. Nastudoval jsem některé techniky pro analýzu požadavků a musel jsem se naučit některé pokročilé věci jazyka VB.NET.

V této kapitole bude nejprve představena automatizace, její vývoj, význam pro společnost a předpokládané trendy do budoucna. V další části bude představena firma Assec Computers a ES Pohoda, který firma používá.

Pro hledání knih a odborných článků, věnujících se zpracovávané tématice, jsem kromě tištěných knih, použil rovněž následující databáze:

- ScienceDirect[1]
- Google Knihy[2]
- Google Scholar[3]
- Springer Link[4]
- Microsoft Academic[5]

[1] http://www.sciencedirect.com/

[2] https://books.google.cz/

[3] https://scholar.google.cz/

[4] http://link.springer.com/

[5] https://academic.microsoft.com/

3.1 Automatizace

Slovník současné češtiny, od společnosti Lingea[6], popisuje slovo automatizace jako „zavádění samočinných zařízení do výroby“. Dle slovníku spisovného jazyka českého[7], zpracovaného lexikografickým kolektivem Ústavu pro jazyk český ČSAV je význam tohoto slova „automatizace pracovních procesů, výroby“, případně ho lze vyjádřit slovy „ustálení“, „zevšednění“ nebo „mechanizace“. Dle slovníku cizích slov portálu ABZ.cz[8], jde o „jeden z hlavních směrů soudobého technického vývoje, založeného na zavádění a užívání samočinných zařízení bez obsluhy“.

Automatizace obecně znamená fungovat, pracovat, jednat nebo se přizpůsobovat a to nezávisle, tedy bez lidského zásahu. Tento termín pochází z řeckého slova „αυτοματος“, což znamená sám o sobě jednající nebo jednající z vlastní vůle čili spontánně. Automatizace zahrnuje stroje, nástroje, zařízení a systémy, jež jsou vyvinuty člověkem, aby prováděli nějakou činnost samostatně, tedy bez zásahu člověka. Existuje ovšem celá řada definic tohoto slova. Před rokem 1950, než se automatizace dostala do současného stavu, byla mechanizace nejčastější formou automatizace. Jakmile se k mechanizaci přidalo automatizované řízení, reprezentující prvek inteligence, rozdíly mezi mechanizací a automatizací, a především potom výhody automatizace začaly být zřetelnější. (Nof Shimon, 2009)

3.1.1 Stručná historie automatizace

Jak již bylo řečeno, slovo automat je řeckého původu a znamená sám o sobě jednající. Již ve starověku si lidé uvědomovali, že schopnost člověka samostatně tvůrčím způsobem jednat a řídit mnohé děje reálného světa je v přírodě zcela mimořádná. Proto lidé obdivovali vše, co se jim podařilo uměle vytvořit, a mělo podobné automatické vlastnosti. Ve starověku, a dokonce i později ve středověku byly takové výtvory považovány za kouzla a zázraky. (Lacko Bronislav, 2000)

Podle publikace (Nof Shimon, 2009) se dá vývoj automatizace rozdělit do 3 etap: generace před automatizovaným řízením, generace před počítačovým řízením a generace počítačového řízení.

První generace automatizace byla typická tím, že ještě nevyužívala automatizovaného řízení. Tyto prvotní náznaky automatizace již měly jisté prvky autonomie procesů a základního autonomního rozhodování, nebraly však vůbec v potaz zpětnou vazbu a nedokázaly tak reagovat na průběh automatizace. Raná automatizace byla typická tím, že si lidé snažili podmanit procesy přírody, aby je mohli opakovat dle vlastního uvážení. Automatizace působila lidem požitek a taky zvyšovala produktivitu při menším úsilí a nižším riziku. (Nof Shimon, 2009)

V knize (Lacko Bronislav, 2000) je vývoj automatizace rozdělen do historických epoch lidstva. Kniha popisuje, jak již ve starověku, 200 let př. n. l., ve městě Alexandria, využívali učenci sílu páry a teplého vzduchu pro otevírání těžkých

[6] http://www.nechybujte.cz/

[7] http://ssjc.ujc.cas.cz/

[8] http://slovnik-cizich-slov.abz.cz/

bronzových vrat a další v té době nepochopitelné věci. Jelikož v té době byla automatizace zastřena rouškou tajemna, lidé si neuvědomovali, že ji vlastně vidí všude kolem sebe, a působila na ně jako něco nadpřirozeného. Ve středověku byly potom středem pozornosti různé mechanické „hračky“. Vrcholným představitelem řemeslnické zručnosti a dovednosti byl hodinový stroj, který obsahoval řadu důmyslných regulátorů. Příkladem zručnosti a umu tehdejších hodinářských mistrů jsou orloje, které dokážou očarovat i v současnosti. (Lacko Bronislav, 2000)

Druhá generace přinesla automatizované řízení, automaty řízené počítačem však byly stále velkou neznámou. Automatizované řízení, poskytovalo lepší stabilitu a spolehlivost, možnost komplexnějších rozhodovacích procesů, a především lepší kontrolu nad kvalitou automatizace. Tato generace zahrnuje období od 15. století až do 40. let 20. století. (Nof Shimon, 2009)

Dobrým příkladem automatizovaného řízení je vylepšení parního stroje Jamesem Wattem. Tento anglický mechanik přidal do parního stroje odstředivý regulátor otáček, čímž několikanásobně zvýšil jeho výkon. Přínosem tohoto řešení bylo především odstranění ručního ovládání rozvodu páry, čímž ukázal v praxi význam automatického řízení, při ovládání složitých strojů. (Lacko Bronislav, 2000)

Dalším průkopníkem v oblasti automatizace byl Joseph Marie Jacquard. Tento francouzský tkadlec a vynálezce opatřil své tkalcovské stavy možností programového řízení. Pomocí pásu tuhého kartonu, který procházel čtecím zařízením, předával stavu informace o tom, co má dělat. Tento „program“ bylo přitom možné velice snadno vyměnit za jiný. Tento vynález byl předzvěstí děrných štítků a později děrných pásek, které se využívaly jako prostředek pro vstup programů a dat do prvních počítačů. Jacquard byl za tento vynález uveden do francouzské Akademie věd. (Lacko Bronislav, 2000)

S příchodem kapitalismu se obrátila pozornost vědců, vynálezců a konstruktérů k řešení automatů, které umožňovaly zvyšovat produktivitu lidské práce. Právě takováto řešení se v této době dočkala největšího obdivu, pro jejich autory však jen málo kdy znamenala zdroj příjmů. Modernizace továren pomocí těchto automatů však přinášela vyšší zisky jejich majitelům. (Lacko Bronislav, 2000)

Během první průmyslové revoluce, typické hromadným zaváděním strojů do výroby, došlo k velkému technologickému a sociálně-ekonomickému převratu. V manufakturách rozšířená rukodělná práce byla stále více nahrazována prací strojů, které dokázaly v porovnání s ní produkovat několikanásobně větší množství výrobků. Zavádění strojů do výroby však mělo značný sociální dopad. Vyšší výkonost strojů připravovala dělníky o práci a tím i o mzdu, potřebnou k obživě. Ve středověku byly automaty obdivovány jak šlechtou, tak prostým lidem. Novodobé stroje však byly rozbíjeny dělníky, kteří v nich mylně viděli příčinu své bídy. Zavádění čím dál vyššího stupně automatizace výroby, které v té době mělo vrchol v pásové výrobě Fordových automobilových závodů, tak přineslo současně velký sociální problém. Vize dosažení blahobytu, která na začátku strojové velkovýroby a automatizace byla velmi rozšířená, byla velmi brzy vystřídána deziluzí a zklamáním. (Nof Shimon, 2009)

Vývoj informačních a komunikačních technologií má významný vliv na sofistikovanost a efektivnost automatického řízení. Třetí generace vývoje automatizace se datuje do 40. let 20. století a pokračuje až dodnes. Na automatizaci měla významný vliv druhá světová válka, ta vedla ve svých důsledcích ke zvýšení požadavků na zbrojní výrobu a dále zvýšila požadavky na ještě rozsáhlejší automatizaci výroby. Jelikož se výrobní stroje stávaly stále složitějšími, bylo nutné najít taková řešení, která by umožňovala automatizovat složité řídící funkce. (Nof Shimon, 2009)

Byly to především vojenské účely, co vedlo k dalším požadavkům na urychlení výpočtů. Ty přivedly skupinu amerických vědců v čele s profesorem N. Wienerem kolem roku 1945 ke zjištění řady shodných rysů v chování složitých řídicích systémů a lidského mozku. Dosažené výsledky popsali v knize „Kybernetika neboli řízení a sdělování v živých organismech a strojích“. Obecné principy automatického řízení, které kybernetika jako nový vědecký obor popsala, pomohly konstruktérům prvních amerických samočinných počítačů – reléového počítače MARK I (1937) a elektronkového počítače ENIAC (1946). Do počátku éry samočinných počítačů stavěl člověk stroje, které znásobovaly jeho sílu, rychlost, vidění apod., a automatizovaly fyzickou práci. Nyní však dovedl postavit stroj, který do určité míry dovedl rychle napodobit duševní práci člověka, a mohl jej využít k realizaci složitých řídicích systémů. (Lacko Bronislav, 2000)

Současná nízká cena automatizačních prvků a prostředků dovoluje využít automatizaci nejen v průmyslu, ale i v domácnostech. Řada výrobků spotřebního zboží každodenního použití v domácnostech má automatizovány některé svoje funkce. Každý z nás se setkává se spotřebiči, jako jsou žehličky, mikrovlnné trouby, automatické pračky, rychlovarné konvice atd. (Lacko Bronislav, 2000)

Kniha (Lacko Bronislav, 2000) na vývoj automatizace nahlíží ještě z jiného pohledu. Vývoj není rozdělen striktně podle časového rámce, nýbrž podle míry nahrazování lidské práce. Podle tohoto rozdělení existují tři stupně vývoje automatizace:

- 1. úroveň – instrumentace, kdy je pracovní proces vybavován ručními nástroji
- 2. úroveň – mechanizace, kdy je fyzická lidská práce nahrazována činností strojů
- 3. úroveň – automatizace, kdy je duševní a řídící práce nahrazována činností strojů

3.1.2 Důvody automatizace

Stejně jako každá racionální činnost je i automatizace založena na rozumných důvodech. Tyto důvody obvykle sledují nějaký cíl. Cílů automatizace může být více, avšak obecně se dá říci, že hlavním cílem je nahradit činnost člověka. Dle knihy (Lacko Bronislav, 2000) lze důvody k automatizaci rozdělit do třech kategorií. Zapojení automatizace může být v některých případech nezbytně nutné, potom lze automatizaci považovat za vynucenou. Další důvody sledují cíle tržního hospodářství, tedy generování zisku. Do poslední kategorie se potom řadí všechny ostatní

důvody. Do první skupiny tedy patří případy, kdy je automatizace vynucena jistými skutečnostmi. Může jít například o tyto situace:

- V některých případech je nutné člověka z procesu odstranit, protože jeho bezprostřední přítomnost představuje často smrtelné nebezpečí. Může jít například o práci ve velkých hloubkách, při vysokých teplotách nebo manipulaci s radioaktivními látkami
- V jiných situacích potřebujeme člověka z procesu vyřadit, protože jeho fyzické a duševní schopnosti nejsou uzpůsobeny zvládnout danou činnost dostatečně rychle, přesně či v daném rozsahu. Jde například o řízení průběhu řetězové reakce v jaderném reaktoru, navádění raket apod.
- Dále taky můžeme člověka z procesu vyřadit, protože automaticky řízené stroje zvládnou požadované úkony vykonat s vyšší jakostí než člověk. Jedná se například o situaci, kdy robot vede stříkací pistoli rovnoměrnou rychlostí po složité dráze tak dobře, že vytvoří na povrchu karosérie kvalitní a stejnoměrnou vrstvu laku
- Automatické zařízení můžeme taky využít v případech, kde není nebo nemůže být člověk přítomen, aby požadované činnosti vykonal. Příkladem může být například obsluha kosmických sond nebo signalizačních bójí v moři
- Důvodem k nasazení automatizace mohou být také činnosti, ke kterým si nemůžeme dovolit vázat odpovídající množství lidské práce. Jde například o automatické spojování telefonních hovorů, automaty, které obsluhují zákazníky i v nočních hodinách atd.

Do další skupiny potom patří důvody, které sledují cíle tržního hospodářství. Jde například o maximalizaci zisku, snižování nákladů atd. Některé ukázky jsou zmíněny v následujícím výčtu:

- Automatické zařízení představuje snížení výrobních nákladů ve srovnání s neautomatizovanou výrobou. Může se jednat o úsporu přímých mzdových nákladů, díky menší potřebě lidské práce nebo také o úsporu materiálů, kdy v důsledku přesnější výroby vzniká menší odpad
- Díky automatickému zařízení je zvýšena produktivita práce a objem výroby. V porovnání s neautomatizovanou výrobou je tak za danou časovou jednotku vyrobeno větší množství výrobků
- Použití automatizace dodává výrobku užitečné vlastnosti, kterých si zákazník cenní. Díky tomu má výrobek pro zákazníka větší hodnotu a vzniká tak po něm větší poptávka
- Automatizace může také umožňovat produkovat výrobky o vyšší jakosti, což může firma následně promítat do zvýšení ceny výrobků

V poslední skupině jsou potom důvody, které nejsou nezbytné, ani se netýkají tržního hospodářství. Často se jedná o důvody, které pro někoho nemusí mít příliš významnou hodnotu, někdo jiný si jich však může cenit. Jedná se například o tyto důvody:

- Automatizace může být použita například z důvodu prestiže. V takovém případě může firma chtít poukázat na své konstrukční, technické či finanční schopnosti a možnosti
- Automatizace může také zvyšovat pohodlí člověka. Jde například o automatické otevírání vrat garáže, automatické ovládání různých elektronických zařízení, automatické spouštění oken v autě a další
- V neposlední řadě může být automatizace rovněž zdrojem zábavy. Jde o různé herní automaty a automatizované dětské hračky

3.1.3 Přínosy automatizace

Přínosy automatizace jsou značně odvozeny od důvodů, které k ní vedly. V tržní ekonomice se všechny kroky hodnotí z pohledu nákladů a přínosů, avšak toto hodnocení nemusí být vždy rozhodující. Mezi hlavní přínosy automatizace patří:

- zkrácení doby výroby a možnost rychle reagovat na požadavky zákazníka
- podstatné zvýšení jakosti výrobků
- snížení výrobních nákladů
- optimalizace výrobního procesu a další (Lacko Bronislav, 2000)

3.1.4 Trendy v automatizaci

Automatizace je velice rychle se vyvíjejícím oborem a v posledních letech to platí obzvlášť. Proto je obtížné odhadovat, kam bude její další vývoj směřovat. Podle aktuálních trendů v automatizaci a ve společnosti lze však určité předpoklady vyvodit. Pokouší se o to i autor knihy (Lacko Bronislav, 2000), který uvádí hned několik směrů, kterými by se měla automatizace vyvíjet. Jelikož je tato publikace již z roku 2000, můžeme rovnou i vyhodnotit, jak daleko měli tyto predikce od skutečnosti.

Automatizace je jedním z nejdynamičtějších technických oborů. Dnes je více než zřejmé, že se jedná o mezioborovou disciplínu. Automatizace využívá nejnovější mikroelektronické součástky a rovněž pracuje s výsledky z různých oborů, především elektroniky a techniky počítačů, informačních a komunikačních technologií a dalších. Automatizace již v dnešní době není zdaleka takovou výsadou jako dříve a je využívána v mnoha oborech. Automatizační technika se bude dále rozšiřovat do různých oborů a také do domácností.

Automatizace se blízce spojena mikroelektronikou, jejíž vývoj umožní další zmenšování rozměrů a spotřeby elektrické energie automatizačních prostředků. Rovněž dojde ke zvýšení složitosti programovatelných funkcí automatů, razantně se zvýší spolehlivost a sníží cena automatizačních prostředků.

Díky poklesu cen bude automatizační technika používána pro bezprostřední měření zdraví člověka. Očekává se rovněž produkce osobních robotů, starajících se o domácnost, či pomáhajících při navigaci během jízdy v autě. V dnešní době již existují chytré náramky, které monitorují například tep uživatele. Rovněž již existují chytré vysavače, které zajišťují úklid domácnosti. Pokroků bylo rovněž dosaženo i v automatické navigaci a řízení osobních automobilů.

Umělá inteligence doznala v posledních letech rovněž velkého pokroku. V knize (Lacko Bronislav, 2000) bylo predikováno, že činnosti jako rozpoznávání obrazů, robotické vidění, komunikace strojů v přirozeném jazyce a další, budou postupně přicházet do běžného provozu. V dnešní době se s těmito věcmi setkáváme skutečně velmi často a nepřijde nám to vůbec zvláštní. Očekává se rovněž rozšíření expertních systémů[9], pro podporu složitých činností.

3.2 Představení firmy

Firma ASSEC Computers (dále jen „firma“), pro kterou je toto řešení zpracováváno, působí v oblasti informačních technologií. Firma poskytuje svým zákazníkům široké spektrum IT služeb, od prodeje informačních technologií, až po správu počítačových sítí. Informace o firmě uvedené v této kapitole jsou čerpány především z vlastních poznatků a zkušeností, ale také z webových stránek firmy (ASSEC Computers, 2012).

Obr. 2 Logo společnosti ASSEC Computers (ASSEC Computers, 2012)

Firma působí především v Hustopečích, kde má své sídlo, a blízkém okolí. Její zákazníky však můžeme najít po celé České republice. Jedná se o malou firmu s 5 zaměstnanci pracujícími na plný úvazek a několika dalšími zaměstnanci s částečným úvazkem. Firemní struktura je velice jednoduchá. Ve vedení firmy je pouze jeden člověk, její zakladatel, který současně vykonává i pozici servisního technika. Ostatní zaměstnanci vykonávají různé pozice, jako je prodejce, účetní, servisní technik atd.

Prodej zboží probíhá dvěma způsoby, buď přímo na firemní prodejně v Hustopečích, nebo na internetovém e-shopu[10]. Firma je partnerem společnosti Comfor[11], takže nabízený sortiment se částečně odvíjí od toho jejich. Na firemní prodejně je rovněž nabízen spotřební materiál, jako jsou tonery a papír do tiskáren, kabely atd. Dalším partnerem firmy je společnost Stormware[12], což je softwarová společnost, zabývající se vývojem především ekonomického softwaru pro platformu Microsoft Windows. Od toho se odvíjí i služby nabízené firmou Assec Computers, jež zahrnují prodej, nasazení, správu a konzultace, pro produkty firmy Stormware.

[9] Systémy napojené na bázi dat, ze které získávají znalosti pro řešení složitých problémů.

[10] http://eshop.assec.cz/

[11] http://www.comfor.cz/

[12] http://www.stormware.cz/

Pro své fungování využívá firma rozličné hardwarové prostředky. Někteří zaměstnanci využívají notebooky, jiní stolní počítače, firma však nemá žádného výhradního dodavatele či výrobce hardwaru. U programového vybavení je situace poněkud odlišná. Napříč celou firmou se používá software společnosti Microsoft. Všechny počítače ve firmě využívají operační systém Windows, v současnosti nejčastěji Windows 10. Servery běží na operačním systému Windows Server, firemní mobilní telefony využívají operační systém Windows Mobile. Všichni zaměstnanci využívají kancelářský balík MS Office, pro správu mailového účtu využívají všichni klienta MS Outlook a takto bych mohl pokračovat dále.

Pro vedení účetnictví využívá firma ekonomického systému Pohoda, vyvíjeného společností Stormware. Využívání tohoto softwaru je odůvodněno především partnerstvím se společností Stormware. Díky partnerství má firma zajištěnu lepší podporu ze strany vývojářů a v neposlední řadě také výhodnější cenu v porovnání s běžnými zákazníky. V roce 2015 bylo ve firmě nasazeno CRM od společnosti Vistos[13], které je s ES Pohoda provázáno. Tento informační systém přinesl spoustu nových funkcí jak pro vedení, tak pro běžné zaměstnance. Vedení dostalo silný nástroj pro přehled nad firemními procesy a výkonností zaměstnanců. Pro zaměstnance pak tato změna znamená značné usnadnění každodenních úkonů, jako je vykazování práce atd.

3.3 Ekonomický systém Pohoda

Pohoda je ekonomický a účetní software vyvíjený firmou Stormware. Jde o českou softwarovou firmu, která se zabývá tvorbou produktů pro platformu Microsoft Windows. Sídlo společnosti je v Jihlavě, své pobočky má však po celé České republice a na Slovensku. Vlajkovou lodí společnosti je ekonomický software Pohoda, kromě něj však firma vyvíjí i další produkty:

- PAMICA – personalistika a mzdy
- TAX – daňová přiznání
- POHODA BI – reporting dat z ES Pohoda
- GLX – kniha jízd a cestovní příkazy
- WINLEX – zákony a předpisy
- mPOHODA – mobilní fakturace a prodej (Stormware, 2016)

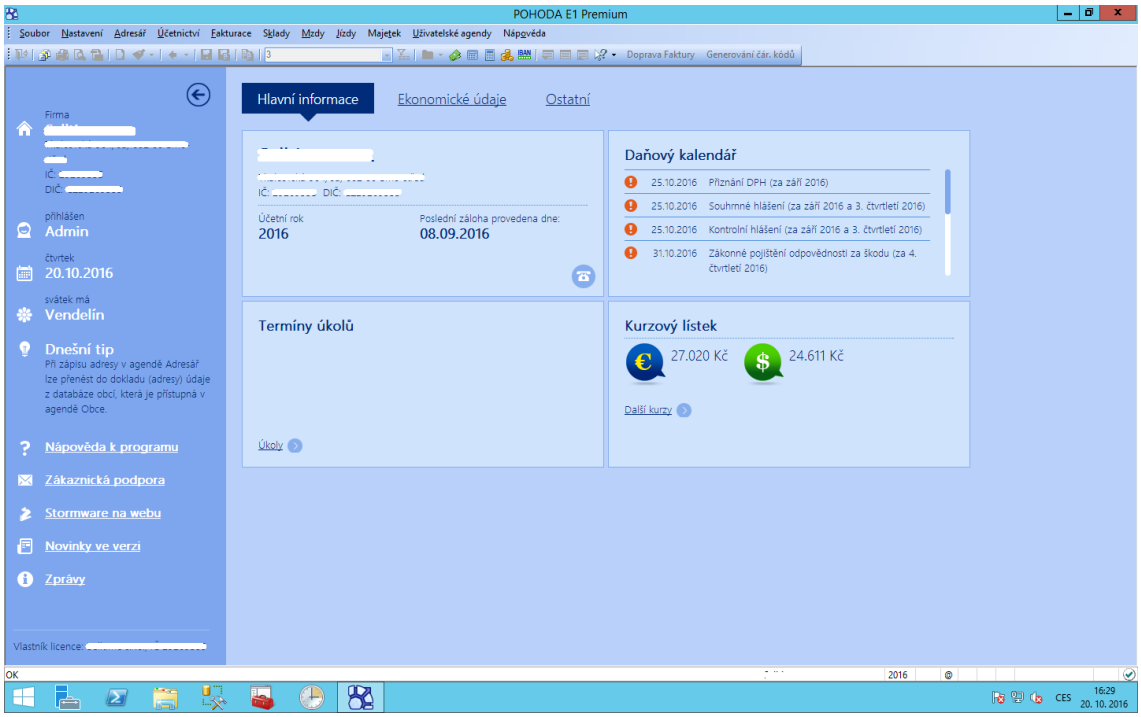
Jak již bylo řečeno, hlavním produktem firmy Stormware je ekonomický systém Pohoda. Pohodu lze pořídit ve třech základních řadách, podle potřeb zákazníka. Ve své základní řadě obstará Pohoda všechny běžné úkony, proto je mezi zákazníky nejrozšířenější. Specifičtější požadavky pak zvládnou obsloužit její vyšší řady.

- POHODA – základní řada, slouží pro živnostníky a menší firmy, dokáže obsloužit vše od účetnictví po daňovou evidenci

[13] http://vistoscrm.cz/

- POHODA SQL – oproti základní řadě umožňuje souběžnou práci více uživatelů, a proto nabízí také správu přístupových práv a vyšší zabezpečení dat
- POHODA E1 – oproti nižším řadám nabízí navíc možnost přidávání vlastních agend[14] a datových polí[15], větší možnosti správy přístupových práv, rozšířené funkce skladů a další (Stormware, 2016)

Prostředí ES Pohoda je postaveno na modulové struktuře a jednotlivé moduly se zde nazývají agendy. Agendy rozčleňují funkcionalitu Pohody do logických celků. Základní verze Pohody obsahuje agendy dostačující pro plnohodnotné vedení účetnictví. Ve verzi Pohoda E1 je možné přidávat další agendy, které dále rozšíří funkcionalitu podle specifických požadavků zákazníka. Mezi základní agendy patří například Přijaté objednávky, Vydané faktury, Příjemky, Sklad, Výdejky a další.



Obr. 3 Úvodní obrazovka ES Pohoda

Na obrázku 3 můžete vidět úvodní obrazovku ES Pohoda. Prostředí aplikace působí na první pohled trochu zastarale a po stránce vzhledu má jistě co dohánět. Na úvodní obrazovce je přehled základních informací o účetní jednotce, přihlášených uživatelích a několik dalších obecných informací. Jelikož nejsem v účetnictví příliš zběhlý, netroufnu si hodnotit, jak je na tom ES Pohoda v porovnání s konkurencí po stránce funkční a obsahové.

Jak již bylo řečeno, ES Pohoda je rozdělen do agend. Vyvíjená aplikace bude při své činnosti pracovat především s agendami Zásoby a Příjemky.

[14] agenda – část systému ES Pohoda, rozšiřující funkcionalitu

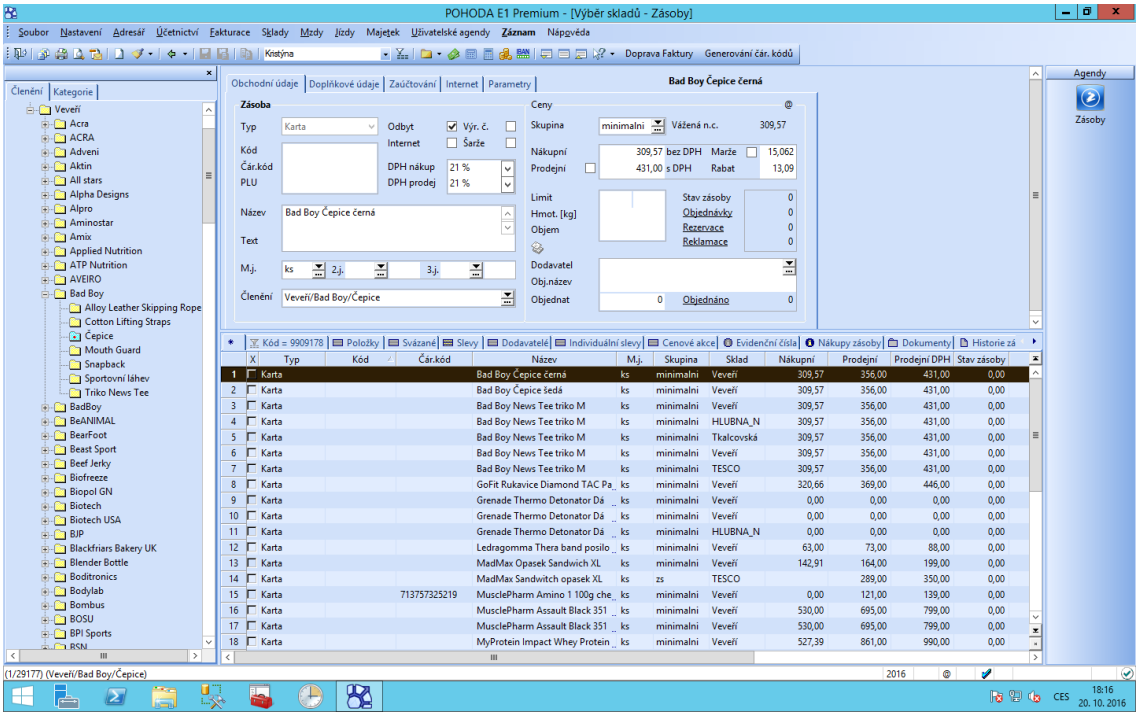
[15] datové pole – atribut účetního dokladu v ES Pohoda

3.3.1 Agenda Zásoby

Jak již bylo řečeno, agenda je část ES Pohoda, která se věnuje určité problematice. Agenda Zásoby tvoří jakýsi přehled vašeho skladu. Každá zásoba je reprezentována skladovou kartou, u které jsou evidovány údaje, jako je název, množství zásoby na skladě, nákupní a prodejní cena a další. Dokonce i zásoba, která má na skladě nulové množství, může mít stále skladovou kartu.

Zvyšování množství zásoby na skladě probíhá pomocí příjemek. Jakmile uživatel vytvoří příjemku, na níž figuruje určitá skladová karta, bude skladové množství této zásoby navýšeno. Z toho lze logicky odvodit, že pokud chceme naskladnit nějaké zboží, musí pro něj existovat skladová karta. Podobným způsobem probíhá i snižování množství zásoby na skladě. Jakmile uživatel vytvoří například vydanou fakturu s nějakou skladovou kartou, sníží se její množství na skladě.

Každá účetní jednotka[16] může mít hned několik skladů, přičemž každý sklad se dá rozdělit skladovým členěním na menší části. Každá skladová karta má přiřazeno skladové členění, které reprezentuje umístění zásoby na skladě. Toto členění usnadňuje orientaci na skladě, umožňuje pracovat pouze s některými zásobami atd. Strukturu skladu můžete vidět na obrázku 4 v levé části obrazovky. Pro editaci skladového členění má ES Pohoda vlastní agendu.



Obr. 4 Agenda Zásoby

Na obrázku 4 můžeme vidět vzhled okna s otevřenou agendou Zásoby. V levé části je možné vybírat konkrétní sklady, případně skladová členění. Zcela vpravo potom

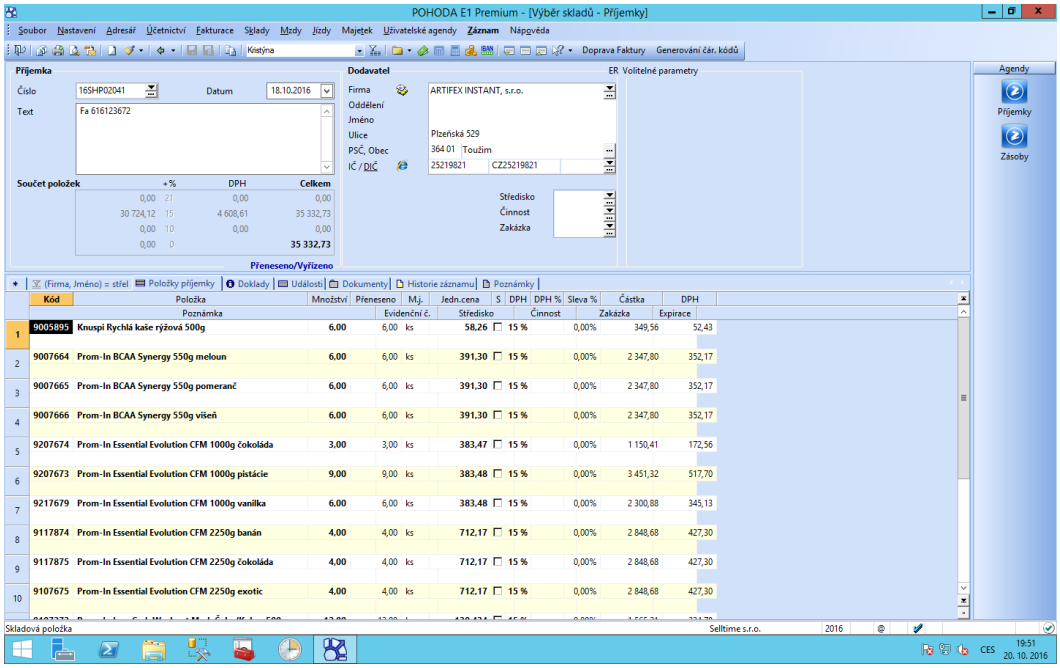
[16] účetní jednotka reprezentuje nejčastěji účetnictví jedné firmy

můžeme vidět otevřené agendy, mezi kterými lze pak snadněji přepínat. Aktuálně máme otevřenou pouze agendu Zásoby, jakmile jich otevřeme více, budou se řadit pod sebe. V prostřední části obrazovky jsou potom samotné zásoby. Dole můžeme vidět seznam zásob, mezi kterými lze procházet a libovolně je filtrovat. Uživatel si může přehled zásob přizpůsobit a zobrazit pouze atributy (reprezentované sloupci), které ho zajímají. Jakmile vybereme nějakou zásobu, informace o ní se nám zobrazí v horní části uprostřed, hned nad seznamem, ze kterého jsme vybíraly. U každé skladové karty jsou vedeny informace jako je její název, měrná jednotka, množství na skladě, cenová skupina a další.

3.3.2 Agenda Příjemky

Tato agenda slouží k manipulaci s účetními doklady nazvanými „příjemky“. Jak již bylo naznačeno v předchozí kapitole o agendě Zásoby, příjemky vytváříme, pokud chceme přijmout zboží, nebo prostě zvýšit množství zásoby na skladě. Jak již bylo rovněž napsáno, důležitým předpokladem k tomu, abychom mohli zásobu naskladnit pomocí příjemky, je mít pro tuto zásobu skladovou kartu. U každé příjemky je rovněž vhodné uvést dodavatele. Dodavatele lze vybrat z adresáře, což je další agenda, ve které spravujeme kontakty (dodavatele, zákazníky atd.).

Agenda Příjemky vypadá téměř totožně, jako agenda Zásoby, ale její rozhraní je o něco jednodušší. V levé části například není potřeba spravovat skladové členění a rovněž množství informací o příjemce není tak rozsáhle jako u zásob. Agenda Příjemky se však oproti agendě Zásoby liší především v tom, že každá příjemka, obsahuje seznam položek jiné agendy, tedy skladových karet. Tyto položky jsou provázány se skladovými kartami v agendě Zásoby.



Obr. 5 Položky příjemky v agendě Příjemky

Zásoby je na příjemku možné vkládat i v textové podobě. V takovémto případě ovšem přicházíme o jejich provázání s agendou Zásoby a obecně se tento postup nedoporučuje.

3.3.3 XML komunikace s ES Pohoda

Ekonomický systém Pohoda umožňuje použití komunikačního rozhraní XML, díky kterému je možné komunikovat s různými externími aplikacemi, včetně internetových obchodů. Tato forma komunikace se na rozdíl od jiných forem importu a exportu neustále vyvíjí, proto její možnosti, výkonnost a použitelnost neustále rostou. Výhody této komunikace jsou především její otevřenost, nezávislost, dostupnost a v případě použití transformačních XSL souborů také přenositelnost mezi libovolnými systémy. XML komunikace může být spouštěna pomocí dávkového zpracování, které plně automatizuje výměnu informací a umožňuje tak implementovat XML komunikaci s ES Pohoda do různých externích nástrojů. (XML komunikace, 2016)

Podrobnější informace o XML komunikaci a o jejím využití v mojí práci se dočtete v kapitole 6.3, nazvané rovněž XML komunikace s ES Pohoda.

4 Současný stav problematiky

Tato kapitola poskytne čtenáři lepší představu o tom, jaký je aktuální stav problematiky ve firmě, i mimo ni. První část kapitoly se věnuje aktuálnímu řešení daného problému ve firmě. V další části bude představeno několik komerčních řešení, věnujících se zpracovávané problematice. Budou analyzovány aspekty těchto řešení, které by mohly být pro moji práci přínosem, případně pak takové, kterým bude lepší se vyvarovat. Prostudoval jsem rovněž několik studentských prací, které dle mého názoru s touto problematikou souvisí. Všechny poznatky, získané při analýze současného stavu problematiky, jak ve firmě, tak mimo ni, nakonec shrnu v poslední části této kapitoly.

Při hledání studentských prací s podobným tématem, jsem využil následující databáze studentských prací:

- Závěrečné práce MENDELU[17]
- Závěrečné práce – VUT v Brně[18]
- Závěrečné práce MUNI[19]

Při analýze studentských prací mě několik z nich zaujalo, a proto jsem je prostudoval důkladněji. Konkrétně šlo o následující práce:

- CHALUPOVÁ, L. 2016. Automatizace kontroly kódu během vývoje
- LOSKOT, J. 2012. Systém automatizace zasílání balíků
- MACÁKOVÁ, G. 2015. Návrh a implementace aplikace pro správu pracovních cest

4.1 Stav problematiky ve firmě

Aktuální stav řešení ve firmě je důvodem, proč tato práce vznikla. Tento způsob zpracování sice funguje, avšak lidem, kteří s ním pracují příliš nevyhovuje. V této kapitole bude čtenář nejprve obecně seznámen s aktuálním stavem problematiky ve firmě, dále budou analyzována některá konkrétní čísla, týkající se importu dokladů, a na závěr bude navrženo možné zlepšení aktuálního stavu.

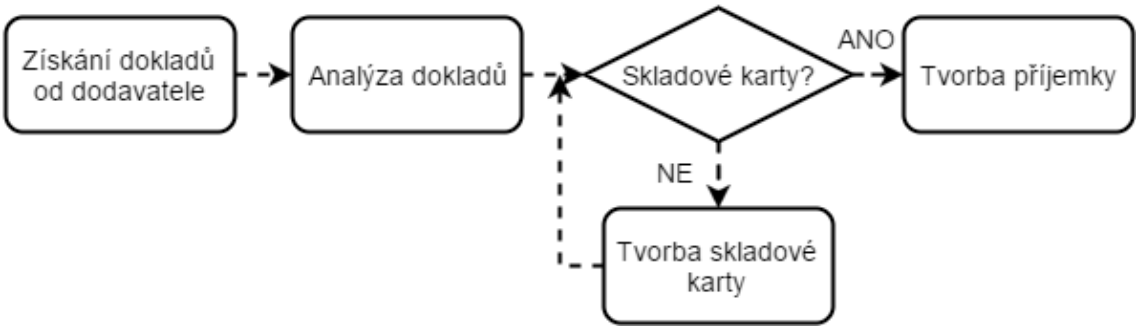
4.1.1 Seznámení s aktuálním stavem

V současné době je ve firmě Assec Computers prováděn import příjemek do ES Pohoda ručně. Zodpovědný zaměstnanec musí provést sérii několika kroků, které jsou zjednodušeně zobrazeny na následujícím diagramu.

[17] https://is.mendelu.cz/zp/

[18] https://www.vutbr.cz/studium/zaverecne-prace

[19] http://is.muni.cz/thesis/



Obr. 6 Diagram zpracování dokladů

Firma nejprve objedná u dodavatele zboží. Jakmile je toto zboží dostupné, dodavatel ho odešle na firmu. K dané objednávce je vytvořena faktura, dodací listy, případně další doklady, které pro nás nejsou při zpracování příliš důležité. Jakmile dojde zásilka fyzicky na firmu, je třeba zboží naskladnit. Informace pro naskladnění zboží získá zaměstnanec z faktury a dodacích listů. Tyto doklady musí dohledat na webovém portálu dodavatele[20], po přihlášení do zákaznické sekce. Zaměstnanec musí nejprve ověřit, zdali souhlasí informace na dokladech se skutečností. Jestliže je vše v pořádku, může začít pracovat s ES Pohoda. V ekonomickém systému musí zaměstnanec nejprve ověřit existenci skladových karet pro všechny položky faktury. Jestliže zaměstnanec narazí na zboží, pro které ještě neexistuje skladová karta, je potřeba ji vytvořit. Na skladové kartě je potřeba zadat informace jako je název, čárový kód či umístění zboží na skladě. Vytváření nových skladových karet je o to složitější, že zaměstnanec musí při této činnosti kombinovat informace o zboží jak z faktury, tak z dodacích listů. Existující skladové karty je potřeba prověřit opravdu důkladně, aby nedošlo k hromadění duplicitních skladových položek. Jakmile jsou vytvořeny skladové karty pro všechny položky faktury, může zaměstnanec přikročit k tvorbě samotné příjemky. U příjemky je potřeba vybrat její položky z agendy Zásoby a k nim přidat informace o dodavateli z agendy Adresář. Dodavatelem firmy je ve většině případů společnost AT Computers, a právě pro ni bude automatizace zpracována.

4.1.2 Analýza aktuálního stavu

Aktuální řešení importu příjemek do ES Pohoda ve firmě ASSEC Computers, představené v předešlé kapitole, má 2 hlavní slabiny.

- časová náročnost
- stereotypní práce vedoucí k chybovosti

Toto jsem tušil již při zadávání práce, ale ještě více jsem se o tom přesvědčil při hlubší analýze této problematiky. Informace jsem čerpal částečně z vlastních zkušeností a částečně z konzultací s pracovníky firmy. Díky tomu, že se mnou sdíleli své osobní zkušenosti, mohl jsem tento problém pochopit o něco lépe.

[20] http://www.atcomp.cz/

Analyzoval jsem data za poslední rok a zjistil jsem, že každý den je potřeba zpracovat 1–2, výjimečně i 3 příjemky od firmy AT Computers. Počet příjemek, i počet zboží na nich jsou závislé na tom, o jaký typ objednávky jde. Můžou to být běžné každodenní objednávky pro zákazníky či potřeby firmy, může jít ale také o velké objednávky, které pravidelně doplňují zboží na skladě. Tabulka uvádí typy objednávek, včetně jejich obvyklé doby zpracování a četnosti.

Tab. 1 Typy objednávek

	Každodenní objednávka	**Velká objednávka**
Četnost	každý den 1–3x	1x za měsíc
Časová náročnost	30 až 90 minut	2–4 hodiny

Zdroj: interní zdroje firmy

Pravidelné měsíční objednávky dodávají tzv. akční zboží, jež koresponduje s akčními letáky dodavatele pro daný měsíc. Často jde o nové zásoby, které ještě nemají skladovou kartu, a tak je potřeba ji vytvořit. Jelikož jde o akční zboží, tak i když již skladovou kartu náhodou má, je ve většině případů potřeba upravit její prodejní cenu, případně jiné atributy.

Druhý nedostatek aktuálního způsobu zpracování, tedy chybovost člověka, při zadávání dat, se mi nepodařilo při analýze nějakým způsobem změřit. Zaměstnankyní, která import provádí, jsem byl ujištěn, že tyto chyby nenastávají příliš často, ale jednou za čas k nim dojde. Jelikož jde o citlivá data, bylo by ideální, kdyby k nim nedocházelo vůbec. Právě tomu se budu svojí aplikací snažit co nejvíce přiblížit.

4.1.3 Možnosti zlepšení

V předešlých dvou podkapitolách byl představen aktuální stav problematiky ve firmě. Rovněž bylo řečeno, jaké jsou hlavní nedostatky tohoto řešení, tedy časová náročnost a chybovost. Právě tyto problémy by měla vyřešit automatizace, konkrétně nějaká aplikace, která bude automatizovat činnost pracovníka, a omezí tak chyby, které plynou například z nepozornosti.

Při provádění této činnosti má počítač oproti člověku velkou výhodu. Počítač dokáže provádět složité výpočty a hledání mnohem rychleji než člověk. Navíc odpadá chybovost v důsledku únavy či frustrace. Počítač provádí činnost na základě zadaných instrukcí, a to bez ohledu různé rušivé vlivy. Díky tomu je počítač ideálním prostředkem pro provádění této činnosti.

Kromě omezení chybovosti, při zadávání dat do ES Pohoda, se rovněž zkrátí čas při dohledávání dat, jak v Pohodě, tak na webu dodavatele.

4.2 Analýza komerčních aplikací

Při hledání komerčních aplikací, zabývajících se stejnou nebo podobnou problematikou jsem zjistil, že tato řešení jsou poměrně rozšířená. Existuje celá řada firem, které se zabývají automatizací zpracování dodavatelských faktur. V této kapitole

uvedu některá z těchto řešení. Jelikož však nemám v plánu za tato komerční řešení platit, uvedu vždy pouze tolik informací, kolik lze získat bez zakoupení placené licence. Pokud bude možné získat nějakou bezplatnou verzi, vyzkouším ji.

4.2.1 Infomatic

Infomatic je konzultační a technologická společnost umožňující ostatním firmám využití informačních technologií k rozvoji jejich podnikání. Tato firma se specializuje na digitalizaci dokumentů z různých vstupních formátů, BI, software pro procesní řízení firmy, prodej dokumentových skenerů a další činnosti. (Infomatic, 2016)

Toto komerční řešení jsem neměl možnost vyzkoušet v praxi, avšak na základě nastudování dostupných materiálů mohu konstatovat, že se jedná o poměrně komplexní řešení. Některé části funkcionality, tohoto řešení, budu implementovat i ve své práci.

Moje aplikace bude rovněž digitalizovat dokumenty, resp. je bude nejprve hledat na stránkách dodavatele a následně je převádět do požadovaného formátu. V tomto je moje řešení omezenější, jelikož se soustředí pouze na jeden způsob získávání dokumentů. Na tuto činnost bude navazovat automatizace importu získaných dokumentů do ES Pohoda. Řešení firmy Infomatic zahrnuje rovněž automatizaci firemních procesů, z dostupných zdrojů se mi však o této funkcionalitě nepodařilo zjistit žádné bližší informace. Předpokládám proto, že jde vždy o konkrétní požadavek zákazníka, který firma analyzuje a následně dále řeší.

4.2.2 Qinve

Jedná se o nástroj, pro rychlé a efektivní zpracování dodavatelských faktur. Uživateli stačí dostat papírovou fakturu do elektronické podoby, k dispozici jsou formáty PDF, JPG či PNG, a nahrát ji do aplikace. Ta sama rozpozná důležité údaje a vloží je do správných polí. Fakturu v takovémto formátu pak můžete importovat do podporovaných systémů. Qinve umí pracovat se systémy Helios, Pohoda, K2, Abra a dalšími. (Qinve, 2016)

Jedná se o jednoduché řešení, které usnadňuje uživateli přepisování dokumentů do elektronické podoby. Aplikace Qinve podporuje celou řadu systémů, jeho funkcionalita je však poměrně omezena. I když bude moje aplikace omezena pouze na jeden systém, do kterého bude moci doklady importovat, jeho funkcionalita bude o něco komplexnější. Uživatel mojí aplikace nebude muset zdlouhavě dohledávat potřebné faktury, aplikace to udělá za něj. Dále potom, stejně jako Qinve, převede získané dokumenty do elektronické podoby a naimportuje je do ekonomického systému.

4.2.3 FlexiInvoice

Toto řešení má velice podobnou funkcionalitu jako to předchozí, tedy Qinve. Na rozdíl od něj se však jedná o webovou aplikaci, což přináší řadu výhod. Aplikace umožňuje online zpracování přijatých faktur s automatickým rozpoznáváním in-

formací na nich. Po nahrání dokladu do aplikace, může být vyžadována kontrola zodpovědným pracovníkem, který ověří správnost tohoto dokladu. Dále aplikace rozpozná data na faktuře a zpracuje je. Jestliže je vše v pořádku, faktura může být zaúčtována. (FlexiInvoice, 2016)

Jak již bylo řečeno u aplikace Qinve, moje řešení bude pokrývat podobnou funkcionalitu, avšak navíc bude ještě automatizovat získávání faktur z webu dodavatele. Řešení FlexiInvoice neumožňuje přímý import do žádného z ekonomických systémů, jež byly zmíněny u Qinve. Na místo toho toto řešení komunikuje s aplikací FlexiBee, což je online účetní software, vyvíjený totožnou firmou jako FlexiInvoice. Toto řešení není tudíž příliš otevřené, pracuje pouze s vlastním účetním softwarem, a proto neumožňuje využití u širšího spektra zákazníků.

4.2.4 plusSystem

PlusSystem je komplexní řešení od firmy ITFuture s.r.o., jež vytváří nadstavbu pro ES Pohoda. Tato firma se specializuje na poskytování komplexních služeb v oblasti skladových systémů a logistiky – od konzultace po školení a podporu. PlusSystem je sada aplikací a nástrojů k ES Pohoda, mající za cíl urychlit, zjednodušit a zpřesnit provádění běžných operací. Je to jediné, ze zde analyzovaných řešení, se kterým jsem se v praxi setkal. (plusSystem, 2016)

Aplikace balíku plusSystem lze rozdělit do 4 skupin, podle jejich použití. První skupina jsou obecné nástroje, které využije každý, kdo v ES Pohoda provádí alespoň fakturaci. Druhá skupina je určená do skladů, tudíž pro různé operace spojené s příjmem a výdejem zboží, vyřizováním objednávek atd. Třetí skupinou jsou výrobní procesy, a aplikace v poslední skupině usnadňují práci účetním firmám. (plusSystem, 2016)

I přesto, že tato sada obsahuje velké množství aplikací, s širokým spektrem využití, aplikace pro automatizaci importu příjemek mezi nimi není. To ještě více poukazuje na fakt, jak je tato práce specifická. Moje řešení je zpracováváno pro konkrétní firmu, bude zpracovávat doklady od konkrétního dodavatele a příjemky budou importovány do jednoho konkrétního ekonomického systému. Řešení, které by odpovídalo těmto požadavkům, jsem při svém hledání nenašel. Proto je nutné vytvořit řešení vlastní.

4.2.5 Shrnutí analýzy komerčních aplikací

Nelze říci, že by všechna analyzovaná řešení pokrývala celý proces zpracování dodavatelských faktur. Například Qinve či FlexiInvoice zcela vypouští automatizované získávání dokladů, uživatel je musí vkládat sám. Řešení FlexiInvoice je na rozdíl od ostatních vybudováno jako webová služba, což usnadňuje jeho nasazení a používání. Na druhou stranu, nutnost internetového připojení může být někdy značně omezující. Pro přehlednost shrnu v tabulce některé klady a zápory analyzovaných řešení.

Tab. 2 Srovnání komerčních řešení

	Klady	Zápory
Infomatic	+ komplexní řešení + digitalizace dokumentů	- chybí automatické dohledávání dokladů
Qinve	+ digitalizace dokumentů + podporuje nejrozšířenější ekonomické softwary	- chybí automatické dohledávání dokladů
FlexiInvoice	+ snadné nasazení a dostupnost + digitalizace dokumentů	- chybí automatické dohledávání dokladů - nutnost internetového připojení - podporuje pouze vlastní ekonomický software
plusSystem	+ komplexní řešení + soustředí se na ES Pohoda	- automatizace zpracování příjemek chybí

I přesto, že řešení plusSystem nenabízí funkcionalitu, kterou zpracovávám, rozhodl jsem se jej do výběru zařadit také. Toto řešení je vyvíjeno pro platformu Windows a ES Pohoda, což má s mojí aplikací společné.

Řešením Infomatic, Qinve a FlexiInvoice chybí automatické dohledávání dokladů, které se mají zpracovat. To je do značné míry způsobeno tím, že tato řešení jsou otevřená a použitelná pro různé dodavatele. Zapracovat automatické dohledávání dokladů pro všechny možné dodavatele je prakticky nemožné. Jelikož moje aplikace bude zpracovávat doklady pouze od jednoho dodavatele, mohu se na tuto funkcionalitu více soustředit a rovněž ji automatizovat. Tato řešení nabízejí místo toho alternativní způsoby vkládání dokladů.

Výhodou i nevýhodou zároveň je pro řešení FlexiInvoice zpracování jako webová služba. To umožňuje snadnější nasazení i používání, zároveň však vyžaduje internetové připojení, které nemusí být vždy dostupné.

4.3 Shrnutí aktuálního stavu problematiky

Při analýze studentských prací jsem si ještě více uvědomil, jak je zadání mojí práce specifické. Jelikož je toto téma vytvořeno pro konkrétní firmu a jde o konkrétní požadavek, který tato firma potřebuje vyřešit, je téměř nemožné najít studentskou práci, která by se zabývala stejnou problematikou. Z analyzovaných prací jsem se poučil v některých aspektech jejich zpracování, proto pro mě jejich nastudování bylo rozhodně přínosem.

U komerčních řešení, která jsem analyzoval, jsem si uvědomil, jak může být tato problematika komplexní. Pokud porovnám řešení Infomatic s aktuálním stavem problematiky ve firmě a tím, co musím vytvořit, bude moje aplikace pokrývat pouze část funkcionality tohoto řešení. To je však na druhou stranu jeho výhodou. Jelikož bude aplikace zpracovávat doklady pouze od jednoho dodavatele, může

se soustředit na automatizaci získávání těchto dokladů, což uživateli ještě více usnadní jeho práci.

Řešení FlexiInvoice je zpracováno jako webová služba, což usnadňuje jeho nasazení a používání. K práci s ním postačí uživateli pouze internetové připojení. To je však zároveň i jeho nevýhodou, jelikož bez Internetu je toto řešení nedostupné.

5 Analýza požadavků

V nejjednodušším slova smyslu, je požadavek vlastnost, jež musí být splněna, aby byl stanovený problém vyřešen. Při vývoji software je to tedy taková vlastnost, kterou musí vyvíjený systém mít, případně činnost, kterou musí být schopen provést. (Sawyer Pete a Kotonya Gerald, 2001)

Metodika RAD, kterou jsem při vývoji aplikace využil, je typická tím, že se příliš nesoustředí na počáteční plánování a důslednou analýzu požadavků. Namísto toho jsou od zadavatele získány pouze základní požadavky, díky kterým může vývojář začít s tvorbou prototypu. Jakmile má zákazník možnost si požadovaný produkt „osahat“, dokáže potom lépe formulovat své požadavky. (SDLC - RAD Model, 2016)

Jako vývojář jsem potřeboval nejprve získat základní představu o tom, jak si zákazník aplikaci představuje, jakou by měla mít funkcionalitu a jak má vypadat. Proto jsem si potřeboval ujasnit, jakým způsobem budu požadavky od zákazníka sbírat, jak je budu členit, jak je zaznamenám a co vlastně potřebuji vědět, abych mohl začít s vývojem.

Jelikož jsem se chtěl soustředit na tvorbu prototypů, rozhodl jsem se získané požadavky rozdělit pouze na funkční a nefunkční. Funkčními požadavky jsou myšleny takové vlastnosti a funkce, které jednoduše musí být splněny. Druhou skupinou požadavků jsou nefunkční požadavky, jejichž úkolem není provádět nezbytné úkony, podporující business cíle firmy, ale i přesto je důležité na ně při vývoji dbát. Jedná se o požadavky jako je výkon, spolehlivost, rozšiřitelnost a další. Pokud je cílem vývojáře vybudovat kvalitní a konkurenceschopný software, měl by na tyto požadavky dbát stejně tak jako na ty funkční. (Systems Engineering Fundamentals, 2001)

Jakmile jsem měl představu o tom, co vlastně od zákazníka potřebuji vědět, uskutečnil jsem několik konzultací, při kterých jsem od něj získal základní požadavky.

5.1 Základní požadavky

Základními požadavky jsou myšleny takové požadavky, které jsem zajistil ještě před samotným vývojem od zákazníka. Tyto požadavky jsem zjišťoval konzultacemi s majitelem a zaměstnanci firmy, kteří budou aplikaci využívat. Procházel jsem s nimi posloupnost kroků, které musí být vykonávány ručně a snažil jsem se zjistit, jaká je jejich představa o automatizaci daných kroků. Rovněž mě zajímala jejich obecná očekávání od vyvíjené aplikace.

5.1.1 Funkční požadavky

Funkční požadavky definují takové vlastnosti systému, které musí být splněny. Často jde o činnosti, které přímo navazují na business cíle firmy. Z prvních konzultací jsem nakonec vybral následují funkční požadavky:

- aplikace bude co možná nejvíce automatizovat import příjemek
- příjemky bude vytvářet z nezpracovaných vydaných faktur a dodacích listů
- vydané faktury a dodací listy bude hledat na webu dodavatele AT Computers
- příjemky budou importovány do ES Pohoda
- aplikace bude pracovat na OS Windows

První požadavek je zcela jasný a není potřeba ho příliš vysvětlovat. Aplikace má v co možná největší míře automatizovat činnost, která je ve firmě nyní prováděna ručně. Touto činností je import příjemek do ES Pohoda.

Druhý požadavek říká, že příjemky budou vytvářeny z vydaných faktur a dodacích listů, které ještě nebyly zpracovány. Na to navazuje další požadavek, že tyto doklady budou získávány na webu dodavatel AT Computers. Úkolem aplikace bude tedy stáhnout doklady z webu AT Computers v použitelném formátu, získat z nich údaje potřebné pro vytvoření příjemky a dostat tato data do požadované podoby pro další použití.

Další požadavek uvádí, že tyto příjemky budou importovány do ekonomického systému Pohoda, který je ve firmě používán. Bude nutné analyzovat možné způsoby importu do ES Pohoda a najít takový, který bude pro moje účely nejvíce vyhovovat. Bude nutné zjistit, v jakém formátu je Pohoda schopna data přijímat.

Poslední požadavek stanovuje prostředí pro nasazení aplikace. V kapitole 3.2 byla představena firma ASSEC Computers a bylo analyzováno HW i SW vybavení, které firma využívá. Jelikož všechny stroje ve firmě běží na OS Windows, bude i tato aplikace vyvíjena pro OS Windows.

5.1.2 Nefunkční požadavky

Tyto požadavky mají mnohdy také kritický vliv na aplikaci. Jejich úkolem ale není přímo podporovat business podniku, ale spíše zajistit kvalitní a stabilní aplikaci. (Wiegers Karl Eugene, 2008)

Během prvotních konzultací jsem si uvědomil, že při vývoji takovéto „malé“ aplikace zákazník zřejmě příliš nefunkčních požadavků mít nebude. Nakonec jsme se shodli na tom, že by aplikace měla mít především tyto vlastnosti:

- bezpečnost
- spolehlivost

Bezpečností je myšleno především to, aby aplikaci mohl používat pouze uživatel s pověřením. Ostatní uživatelé nebudou moci v aplikaci provést prakticky nic. Tím se zamezí nepovolenému přístupu a v samotném důsledku pak nesrovnalostem v účetnictví.

Pod spolehlivostí si zákazník představuje, že systém bude spolehlivý i při větší zátěži, nebude padat a zvládne udělat to, co se od něj očekává.

5.2 Prototyp a další požadavky

Jakmile jsem měl hrubou představu o požadavcích na aplikaci, začal jsem s jejím vývojem. V první fázi jsem nebral v potaz vzhled aplikace, ale soustředil jsem se spíše na posloupnost činností, které musí být provedeny. Funkcionalitu aplikace jsem si rozdělil do těchto 7 částí:

1. Nalezení poslední nezpracované faktury
2. Nalezení dodacích listů svázaných s fakturou
3. Stažení nalezených dokladů
4. Nalezení údajů ve stažených dokladech
5. Ověření existence skladových karet pro zboží na faktuře
6. Vytvoření chybějících skladových karet
7. Vytvoření příjemky ze získaných údajů

Při vývoji jsem postupoval od 1. části až k té poslední. Dával jsem dohromady funkční aplikaci, ke které jsem postupně přidával nové části funkcionality. Po zapracování každé části funkcionality, jsem tento postup testoval a konzultoval se zákazníkem. Ve většině případů měl zákazník vždy nějaké připomínky a nové požadavky, které si předtím nedokázal ani představit.

5.2.1 Nalezení poslední nezpracované faktury

Zákazník si přál, aby bylo v aplikaci možné nějakým způsobem změnit číslo poslední nezpracované faktury. Tato funkcionalita bude použitelná v případech, kdy bude potřeba některé faktury přeskočit a nezpracovávat, nebo naopak když bude potřeba některou fakturu zpracovat opakovaně.

Vydané faktury jsou na webu dodavatele rozděleny do stránek po 30 fakturách. Pro starší faktury je potřeba jít na další stránky tohoto přehledu. Aplikace musí umět dohledat faktury i na dalších stránkách, ale zároveň musí být stanovena nějaká hranice, pro případ, kdy by hledané číslo faktury vůbec neexistovalo.

5.2.2 Nalezení dodacích listů svázaných s fakturou

Zákazník mě informoval o tom, že k faktuře nemusí být svázán ani jeden dodací list, ale taky mohou existovat faktury, které mají dodacích listů více. Aplikace musí být schopna vyřešit všechny tyto situace.

Aplikace musí nalézt všechny dodací listy ke zpracovávané faktuře, jelikož dodací listy jsou nutné ke správnému vytvoření příjemky. Na samotných dodacích listech ani fakturách nejsou nikdy uvedeny všechny potřené údaje, proto je nutné dat dohromady údaje z obou těchto dokumentů.

5.2.3 Stažení nalezených dokladů

Zjistil jsem, že přímo ze stránek dodavatele je doklady možné stahovat pouze ve formátu PDF. Po konzultaci se zákazníkem jsme se dozvěděl o existenci webu[21], ze kterého se dají stahovat doklady vystavené firmou AT Computers v různých dalších formátech, například ve formátu XML. Právě tento formát mi připadal jako nejlépe použitelný pro extrakci dat.

5.2.4 Nalezení potřebných údajů ve stažených dokladech

Jak již bylo zmíněno, pro vytvoření příjemky musí být spojeny údaje jak z vydaných faktur, tak z dodacích listů. Aplikace musí být schopna procházet XML soubory, které stáhne z webu dodavatele, a nalézt v nich všechny potřebné údaje. Aplikace bude získané údaje ukládat do datových struktur a připraví je do formátu vhodného pro import do ES Pohoda.

5.2.5 Ověření existence skladových karet pro zboží na faktuře

Ověření existence skladových karet pro zpracovávané zásoby probíhá podobným způsobem, jako samotný import příjemky. Na základě údajů získaných z dokladů se vytvoří soubor s požadavkem. Jakmile je tento soubor „poslán“ do Pohoda, ta vrátí XML soubor s odpovědí. Z tohoto souboru lze zjistit, jestli skladová karta s hledanými údaji existuje či nikoliv.

Aplikace musí tedy umět generovat XML soubor s požadavkem, na základě kterého mu ES Pohoda pošle odpověď. Tu musí aplikace umět analyzovat a vyhodnotit, které zásoby již skaldovou kartu mají, a které nikoliv.

5.2.6 Vytvoření chybějících skladových karet

Na základě předchozího kroku bude vytvořen další XML soubor s požadavkem. Tentokrát však bude sloužit k vytvoření skladových karet. Těm zásobám, které neměly skladovou kartu, bude vytvořena.

5.2.7 Vytvoření příjemky ze získaných údajů

Jakmile aplikace ověří, že pro všechny zpracovávané zásoby existuje skladová karta, může začít import příjemky. Údaje uložené v datových strukturách aplikace bude potřeba upravit do tvaru, který koresponduje s definovanou strukturou XML požadavku. Opět je potřeba vytvořit XML soubor s požadavkem, na kterém budou informace o dodavateli a položkách příjemky.

[21] http://ws.atcomp.cz/dokumenty.asmx

6 Vlastní řešení

Tato kapitola čtenáře seznámí s návrhem aplikace a výběrem vhodných technologií. Dočte se v ní o tom, jak probíhala implementace, bude uvedeno několik klíčových částí implementace a problémy, se kterými jsem se při vývoji setkal. V závěru kapitoly bude čtenář seznámen s testováním a nasazením aplikace do provozu.

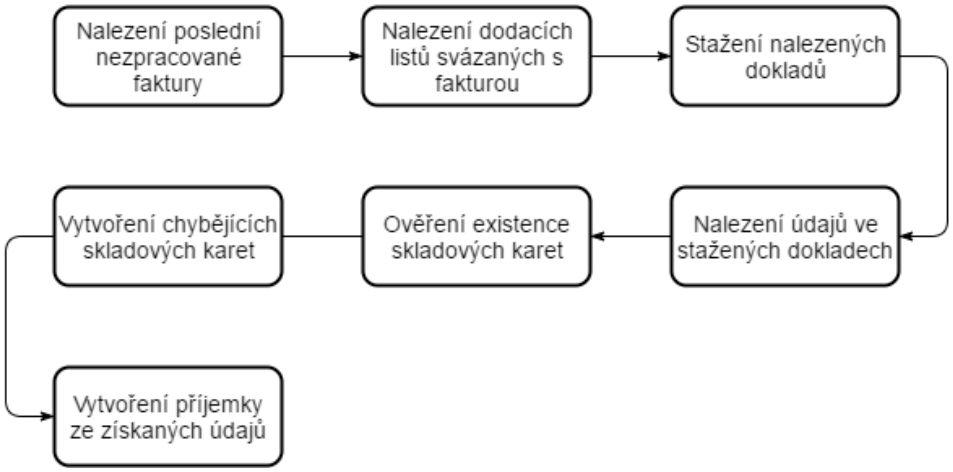
6.1 Návrh řešení

Při vývoji aplikace jsem se snažil držet metodiky RAD, jež se soustředí na rychlé vytváření prototypů, a proto jsem se příliš důkladně nezabýval jejím úvodním návrhem a modelováním.
se spíše na rychlý vývoj funkčních prototypů a jejich konzultaci se zákazníkem. Nebylo by ovšem moudré, začít s vývojem aplikace, bez jakékoliv představy o její struktuře a fungování. Vytvořil jsem proto diagramy, které mi pomohly si tyto věci lépe představit a nyní v tomto směru pomohou i vám. Pro tvorbu diagramů jsem použil Visual Paradigm, jehož licenci máme na fakultě k dispozici zdarma, a webový nástroj pro tvorbu diagramů Draw.io[22].

6.1.1 Úvodní návrh řešení

Aplikaci jsem si nejdříve rozdělil do menších bloků, které jsem již nastínil v kapitole 5.2. Díky tomu, že jsem si aplikaci rozdělil na menší části, byl pro mě jednodušší její návrh, implementace i závěrečné testování.

[22] https://www.draw.io/

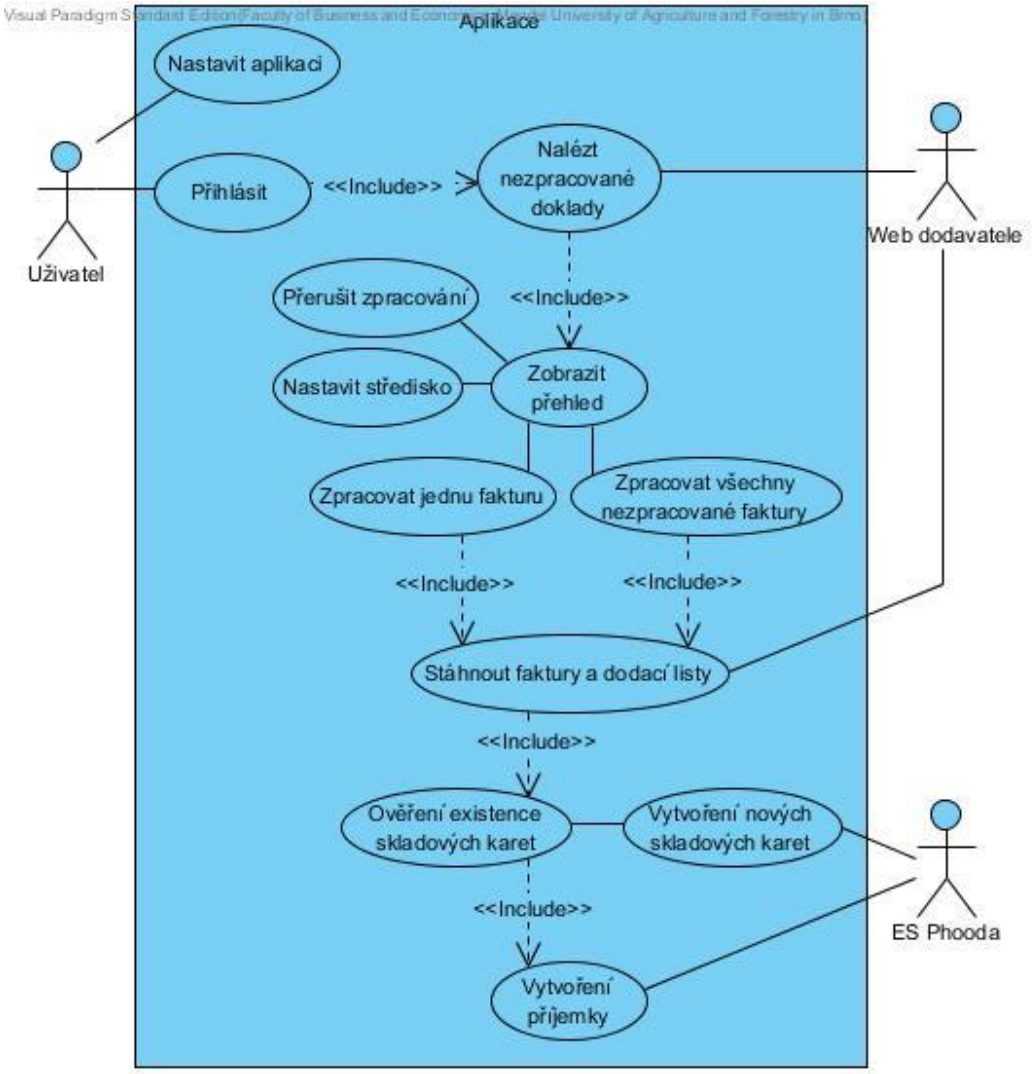


Obr. 7 Rozdělení implementace aplikace

Obrázek 7 rekapituluje jednotlivé části implementace a jejich návaznost. Návaznost je v tomto procesu důležitá. Například bez nalezení poslední nezpracované faktury nelze dohledat dodací listy s ní svázané, nebo tyto doklady stáhnout.

6.1.2 Diagram případů užití

Pro lepší představu o fungování vyvíjené aplikace jsem vytvořil jednoduchý diagram případů užití, jež můžete vidět na obrázku 8.



Obr. 8 Diagram případů užití

Tento diagram přehledně zachycuje vnější pohled na budoucí aplikaci. Prvním aktérem je zde uživatel. Pro práci s aplikací musí mít uživatel v ES Pohoda přidělena práva pro XML komunikaci. K aplikaci bude přistupovat pouze jeden typ uživatele. Aplikace nebude mít vytvořenu žádnou hierarchii přístupových práv, která by jednomu uživateli umožňovala vykonávat něco víc než jinému. Druhým aktérem je web dodavatele, na kterém aplikace hledá a stahuje nezpracované doklady. Třetím aktérem je ES Pohoda, se kterým aplikace komunikuje při vytváření nových skladových karet a příjemek.

Případy užití na sebe často navazují, bez možnosti zásahu uživatele. Prováděné činnosti většinou odpovídají těm, které byly již naznačeny na obrázku 7. Před přihlášením bude mít uživatel možnost nastavení aplikace. Nastavení se bude týkat převážně připojení k databázi ES Pohoda a webu dodavatele. Mimo to bude ještě možné nastavit číslo poslední zpracované faktury.

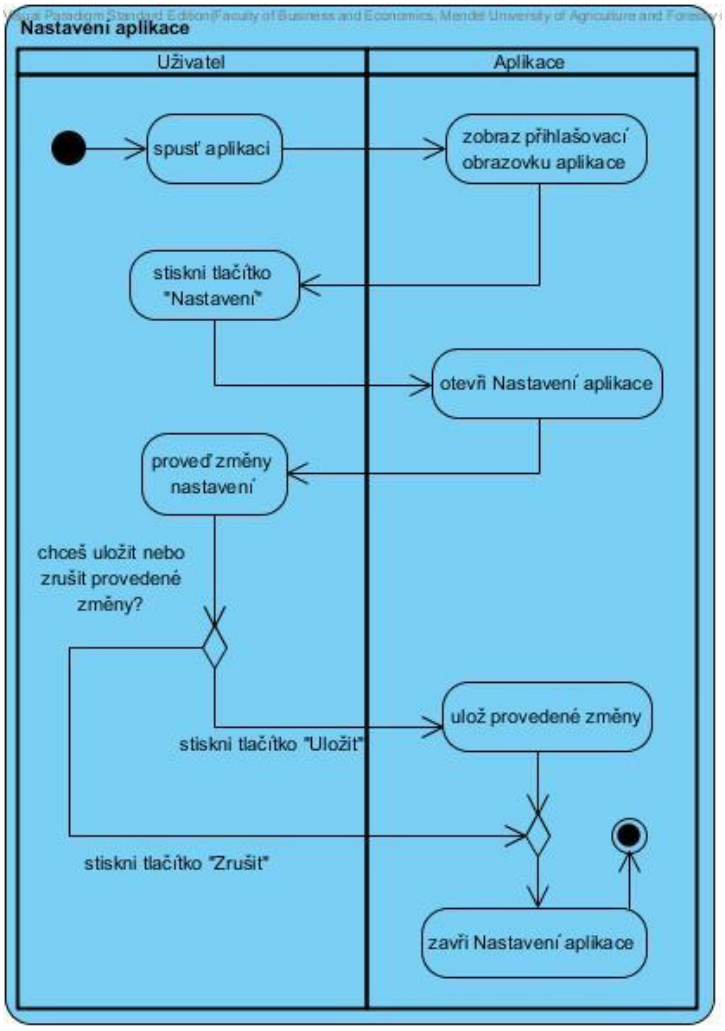
Jakmile bude uživatel úspěšně přihlášen, aplikace automaticky dohledá nezpracované doklady na webu dodavatele. Po dokončení hledání se zobrazí počet nezpracovaných faktur a číslo faktury, která se bude aktuálně zpracovávat. V této fázi bude moci uživatel nastavit středisko, pod kterým budou příjemky vytvářeny. Jakmile tak učiní, může začít zpracovávat dohledané faktury. Aplikace může zpracovat buď jednu, nebo všechny dosud nezpracované faktury. U obou těchto možností dojde ke stažení dokladů z webu dodavatele, ověření existence skladových karet, jejich případnému vytvoření, a nakonec k vytvoření příjemky v účetní jednotce ES Pohoda. Pokud aplikace bude zpracovávat všechny dosud nezpracované doklady, bude se činnost prováděná pro zpracování jedné faktury cyklicky opakovat. Hromadné zpracování může uživatel kdykoliv přerušit.

6.1.3 Diagram aktivit

Pro některé části funkcionality vyvíjené aplikace, jsem rovněž vytvořil diagram aktivit. Tento diagram se používá pro popis dynamických aspektů systému, a ještě důkladněji se věnuje konkrétním činnostem aplikace. Každou aktivitu lze vyjádřit jako sekvenci jednotlivých kroků, které mohou být reprezentovány jako akce (dále nedělitelná činnost) nebo vnořené aktivity (mohou být vyjádřeny dalším diagramem aktivit).

První diagram aktivit popisuje poměrně banální činnost, kterou je nastavení aplikace. Jsou zde zakresleny pouze dvě „swimlanes“, které rozdělují diagram na aktivity provedené uživatelem a aktivity provedené aplikací. Podle toho, kdo aktivitu provádí, je umístěna v patřičné části diagramu. Diagram je na obrázku 9.

Jakmile uživatel aplikací spustí, zobrazí se přihlašovací obrazovka. Pro nastavení aplikace musí uživatel stisknout tlačítko „Nastavení“, načež se mu okno s nastavením otevře. Uživatel může provést libovolné změny, které může následně buď uložit, nebo zrušit. Pokud jsou provedené změny v pořádku a uživatel je chce uložit, aplikace provede jejich uložení a uzavře okno s nastavením.



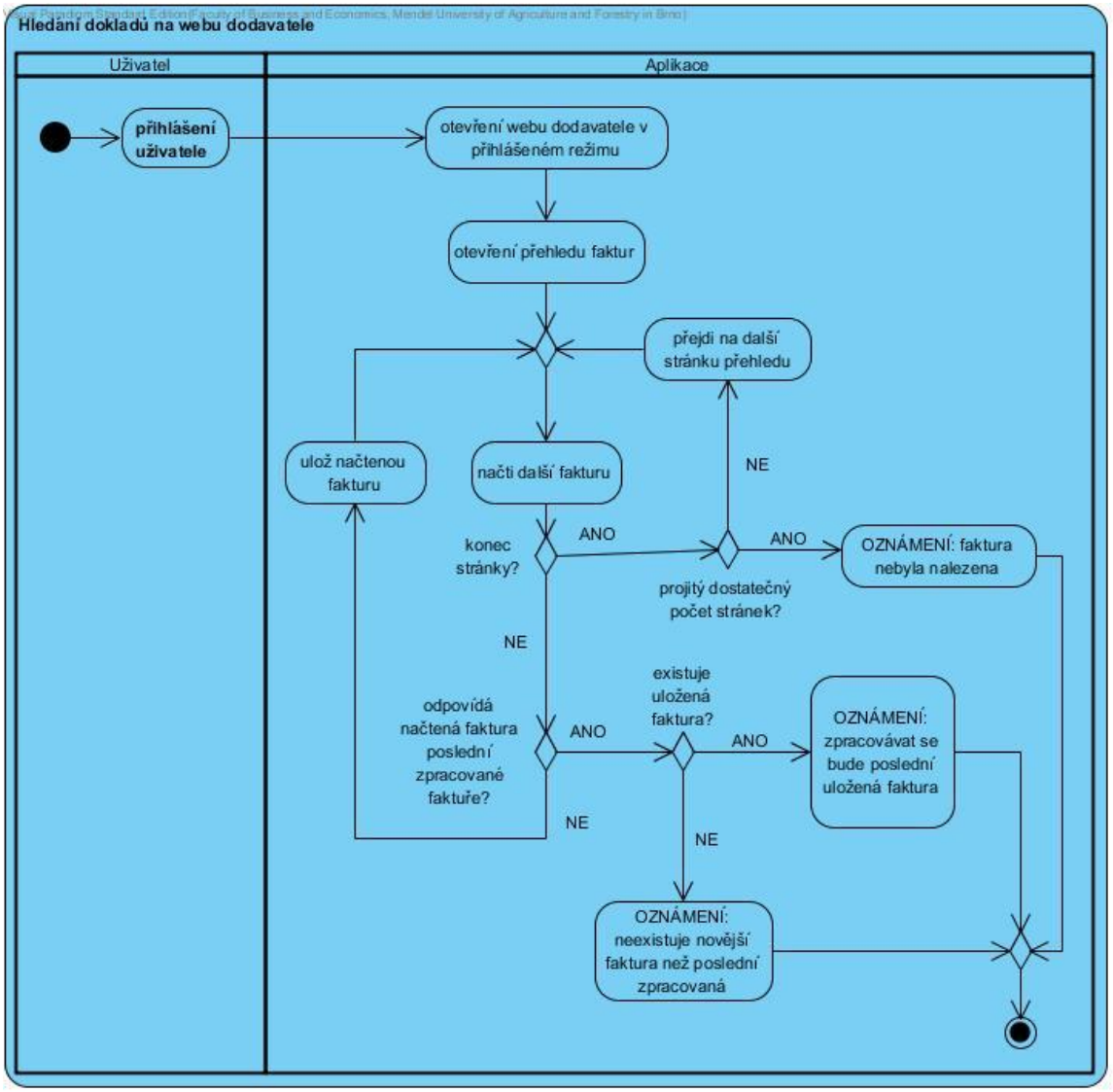
Obr. 9 Diagram aktivit pro nastavení aplikace

Další diagram aktivit jsem zpracoval pro hledání faktur na webu dodavatele. Můžete ho vidět na obrázku 10. Při této aktivitě se budou na webu dodavatele dohledávat pouze čísla dokladů, jelikož samotné stahování bude probíhat z jiného webu, který umožňuje tyto doklady stáhnout v požadovaném formátu.

I přesto, že v této aktivitě jistým způsobem vystupuje web dodavatele, rozhodl jsem se tyto činnosti vyjádřit jako činnosti aplikace. Tento krok jsem si odůvodnil tak, že i přesto, že jsou doklady hledány na webu dodavatele, provádí je samotná aplikace, a nikoliv web.

Tato aktivita bude prováděna ihned po přihlášení uživatele a poté následně při každém dokončení importu příjemky. Jakmile se uživatel do aplikace přihlásí, provede aplikace přihlášení na web dodavatele a přesune se na přehled vydaných faktur. Dále se pokusí načíst poslední nepřečtené číslo faktury. Pokud aplikace nenarazí na poslední fakturu na aktuální stránce, zjistí, jestli se aktuálně přečtená faktura rovná té naposledy zpracované (té kterou zadal uživatel v nastavení apli-

kace). Pokud aktuálně načtená faktura neodpovídá té hledané, uloží si aplikace její číslo pro pozdější zpracování a pokračuje v načtení další faktury na stránce.

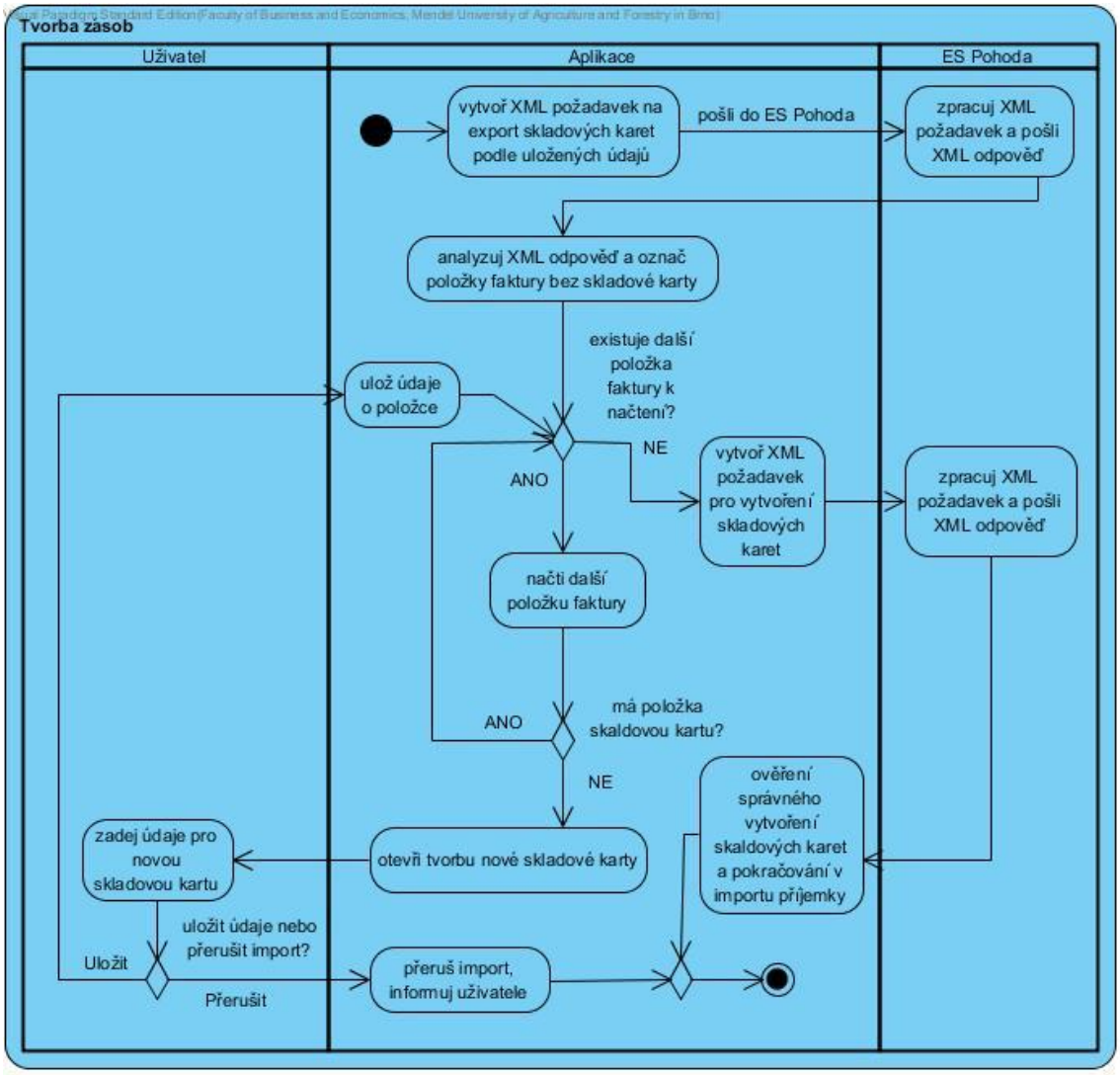


Obr. 10 Diagram aktivit pro hledání dokladů na webu dodavatele

Jestliže aktuálně načtená faktura odpovídá té hledané, musí si aplikace ověřit, jestli byla před touto fakturou načtena už jiná. Jestliže ne, je tato faktura nejnovější a neexistuje žádná další faktura ke zpracování. Pokud existuje nějaká dříve načtená faktura, je to právě ta, která má být aktuálně zpracována. Pokud aplikace při načtení další faktury narazí na konec stránky, ověří si, jestli už předtím prošla dostatečný počet stránek. Pokud ne, přejde na další stránku a opakuje hledání. Pokud již dostatečný počet stránek prošla, ukončí hledání a oznámí neúspěch.

Na tomto diagramu lze dobře vidět, jak je tato činnost plně automatizována, jelikož všechny prováděné akce jsou akcemi aplikace.

Poslední diagram aktivit, který jsem vypracoval, vyjadřuje aktivitu, při které se vytváří nové skladové karty v ES Pohoda. V této aktivitě již vystupují tři „plavecké dráhy“, jsou to uživatel, aplikace a ES Pohoda.



Obr. 11 Diagram aktivit pro vytvoření nové skladové karty

Jakmile aplikace získá data z dokladů stažených na webu dodavatele, musí ověřit existenci skladových karet, pro všechny položky faktury. Za tímto účelem vytvoří aplikace XML požadavek na export skladových karet z ES Pohoda, podle údajů ze stažených dokladů. Tento požadavek je poslán do Pohody, která ho zpracuje a pošle aplikaci odpověď. Aplikace prochází všechny položky faktury a podle přijaté XML odpovědi určí, zdali daná položka má či nemá skladovou kartu. Pokud skladová karta neexistuje, otevře aplikace okno pro tvorbu nové skladové karty a požádá uživatele o zadání potřebných údajů. Uživatel může v takovéto situaci import přerušit, čímž zruší celý import příjemky a aplikace se vrátí na svoji hlavní obra-

zovku, nebo může vytvářenou skladovou kartu uložit, načež aplikace přejde k další položce faktury a provede opět stejný cyklus.

Jakmile aplikace úspěšně získá od uživatele data, pro vytvoření všech chybějících skladových karet, vytvoří na základě nich XML požadavek pro tvorbu nových skaldových karet, který následně pošle do ES Pohoda. Pohoda ho zpracuje a vrátí aplikaci odpověď. Aplikace pro jistotu ještě XML odpověď analyzuje a ověří správné vytvoření skaldových karet. Jestliže je vše v pořádku, pokračuje aplikace v importu příjemky.

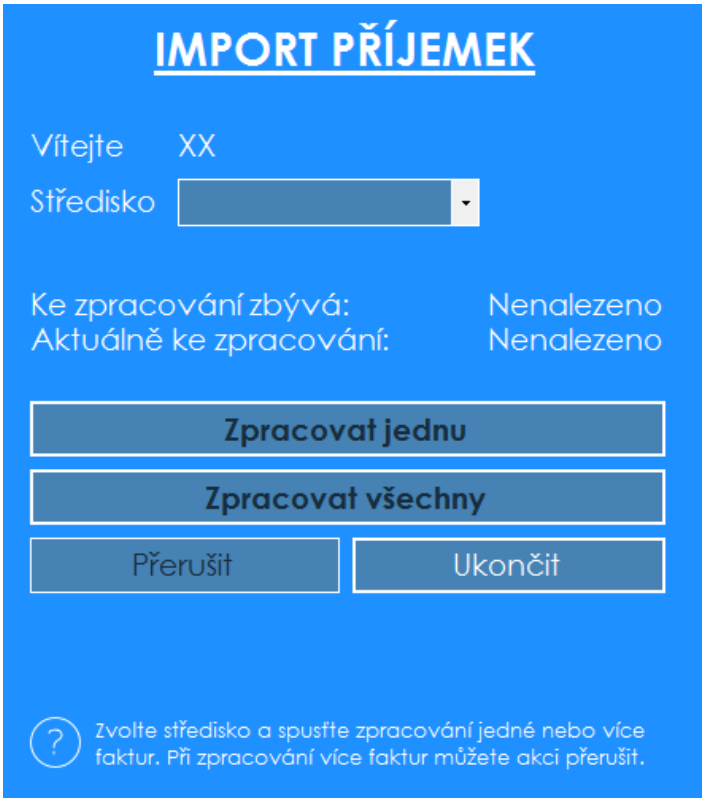
6.1.4 Návrh vzhledu

Kromě návrhu struktury a fungování aplikace jako celku, jsem se v práci věnoval rovněž návrhu uživatelského rozhraní a vzhledu. Během návrhu a implementace funkcionality aplikace jsem se návrhu vzhledu příliš nevěnoval, nechal jsem ho až na závěr vývoje. Zadavatel nepředložil žádné konkrétní požadavky na vzhled aplikace, proto byl návrh vzhledu v mojí režii.

Po zkušenostech s ES Pohoda, jsem se rozhodl vytvořit něco razantně odlišného. Pohoda používá poměrně malé písmo, malá tlačítka a její vzhled působí zastarale. Ve svojí aplikaci jsem chtěl mít pravý opak. Proto jsem se soustředil na jednoduchý design s přehledným uspořádáním. Používám velká tlačítka spolu s velkým, snadno čitelným písmem.

Prvotní návrh vzhledu, rozmístění tlačítek, barvy aplikace atd., jsem kreslil pouze na papír. Tyto náčrtky nejsou v takovém stavu, abych je mohl umístit do této práce. Vývojové prostředí Microsoft Visual Studio má v sobě zakomponován editor formulářů, na něž vývojář vkládá jednotlivé prvky, se kterými dále pracuje při psaní kódu. Tento nástroj mi umožnil navrhovat uživatelské rozhraní přímo v tomto IDE, mohl jsem jej libovolně upravovat a nemusel jsem se příliš zabývat složitým návrhem uživatelského rozhraní, například v nějakém grafickém editoru.

Pro ilustraci přikládám obrázek 9, na kterém můžete vidět konečný návrh vzhledu aplikace. Z tohoto obrázku si můžete udělat představu o rozvržení aplikace, volbě barev a písma, i o celkovém pojetí vzhledu aplikace.



Obr. 12 Vzhled aplikace

Jak můžete vidět na obrázku 9, v aplikaci jsem se rozhodl použít firemní barvy, což jsou modrá a bílá. Použil jsem bezpatkový font Century Gothic, které vychází z fontu sans-serif. Tento font, spolu s barvami, působí moderním dojmem a je snadno čitelný.

6.2 Vývojové prostředí

Na základě současného stavu problematiky ve firmě, analýzy požadavků a navrhnutého řešení jsem provedl výběr vhodných technologií pro tvorbu aplikace. Rozhodl jsem se, že aplikaci implementuji v programovacím jazyce Visual Basic .NET a jako vývojové prostředí tudíž nejlépe poslouží Microsoft Visual Studio.

6.2.1 Microsoft Visual Studio

Rozhodl jsem se aplikaci implementovat v jazyce Visual Basic .NET, k čemuž využiji vývojové prostředí Visual Studio Community 2015 od firmy Microsoft. Toto vývojové prostředí (zkráceně IDE[23]) lze využít k tvorbě konzolových aplikací, aplikací s grafickým rozhraním, webových stránek, webových aplikací či webových služeb. Jak již bylo řečeno, prostředí vyvíjí firma Microsoft, a proto není divu, že produkty

[23] Integrated Development Environment – integrované vývojové prostředí

v něm vytvořené jsou podporovány především na platformách vyvíjených touto společností.

6.2.2 Visual Basic .NET

Pro vývoj aplikace jsem si zvolil objektově orientovaný jazyk Visual Basic .NET. Jedná se o novou generaci jazyka Visual Basic, upravenou pro platformu .NET Framework. S tímto jazykem jsem měl již nějaké zkušenosti z dřívějška. Navíc, tento jazyk spolu s vývojovým prostředím Visual Studio umožňuje snadný a rychlý vývoj, včetně jednoduché tvorby uživatelského rozhraní, což dokonale odpovídá mým požadavkům.

6.3 XML komunikace s ES Pohoda

Kromě klasického ovládání přes uživatelské rozhraní nabízí ES Pohoda možnost komunikovat s okolím aplikace pomocí značkovacího jazyka XML. Díky tomuto rozhraní může Pohoda přijímat a odesílat data do a z různých aplikací, včetně internetových obchodů a externích softwarových modulů, pro zpracování dat souvisejících s databázemi programu Pohoda. Jelikož jde o poměrně rozsáhlou problematiku, firma Stormware vytvořila webovou stránku[24], která se tomuto tématu věnuje.

Tento způsob komunikace je v současné době preferován, například před exportem a importem tabulek i přímým přístupem do databáze, protože zajišťuje lepší bezpečnost při provádění operací. Při XML komunikaci dochází ke kontrole a validaci dat tak, jako by s nimi bylo nakládáno přímo v prostředí ES Pohoda. Při importu dokladů pomocí XML, jsou kontrolovány a případně dopočítány všechny nezbytné údaje. V případě chyby je import přerušen a uživatel upozorněn na nutnost přepracování XML souboru. (XML komunikace, 2016)

XML komunikace je na rozdíl od jiných forem importu a exportu stále ve vývoji a její možnosti, výkonnost a použitelnost tak neustále rostou. Výhody této formy komunikace jsou její otevřenost, nezávislost, dostupnost a v případě použití transformačních XSL souborů také plná přenositelnost mezi libovolnými systémy. XML komunikaci lze používat pomocí dávkového zpracování, které komunikaci zcela automatizuje a umožňuje ji tak implementovat do externích aplikací, vyvíjených obchodními partnery společnosti Stormware, případně jinými společnostmi. XML komunikaci je možné rovněž používat přímo z prostředí ES Pohoda, v takovém případě však tato komunikace zcela nevyužívá svůj potenciál. (XML komunikace, 2016)

6.3.1 Mechanismus XML komunikace

Obecný mechanismus XML komunikace s ES Pohoda lze popsat pomocí několika kroků.

[24] https://www.stormware.cz/xml/

1. Spuštění programu Pohoda z příkazové řádky
2. Načtení INI soubor s parametry importu
3. Načtení XML souboru nebo celého adresáře s XML soubory do ES Pohoda
4. Vrácení XML souboru z ES Pohoda

Ve výčtu byl několikrát zmíněn XML soubor. Tento pojem nemá ovšem pokaždé stejný význam. V kroku 3, kdy se mluví o načtení nebo poslání XML souboru do ES Pohoda, jde o XML požadavek, který má za úkol něco provést. XML požadavek může sloužit pro import (tvorba nových skladových karet) nebo export (export vydaných faktur z ES Pohoda). Ve 4. kroku tento pojem představuje XML odpověď, kterou ES Pohoda vrací po zpracování XML požadavku.

ES Pohoda může pomocí XML komunikace najednou zpracovat buďto jeden samotný XML požadavek, nebo celý adresář, obsahující více XML požadavků. Která možnost se provede, záleží na parametrech INI souboru.

Mechanismus XML komunikace začíná tím, že nějaký externí proces spustí ES Pohoda s parametry pro XML import/export. V mojí aplikaci se provede spuštění ES Pohoda pomocí příkazové řádky. Obecný formát příkazu posílaného do příkazové řádky vypadá následovně.

```
Pohoda.exe /XML "Uzivatel" "Heslo" "Cesta k INI souboru"
```

Příkaz se skládá z cesty k ES Pohoda, parametru „/XML“, který indikuje XML komunikaci, uživatelského jména a hesla, s právy pro XML komunikaci a cesty k INI souboru, jež zprostředkuje XML komunikaci.

INI soubor obsahuje parametry, které musí ES Pohoda znát, aby mohl zahájit import/export. ES Pohoda může zpracovat buď jeden XML požadavek, nebo celý adresář, obsahující několik XML požadavků. To, která možnost se provede, záleží na parametrech INI souboru. INI soubor může obsahovat následující parametry.

- input_dir – cesta k adresáři, ve kterém jsou umístěny XML požadavky do ES Pohoda. Používá se, pokud chceme importovat více XML souborů.
- response_dir – adresář, do kterého se budou umisťovat XML odpovědi z ES Pohoda. Pokud není zadán, vytvoří se adresář Response, ve vstupním adresáři
- input_xml – cesta k XML požadavku. Používá se, pokud chceme importovat pouze jeden XML soubor.
- response_xml – cesta k XML odpovědi. Používá se, pokud importujeme pouze jeden XML požadavek a výstup chceme do konkrétního souboru.
- database – název databáze účetní jednotky, se kterou chceme komunikovat.
- check_duplicity – pokud je nastavena hodnota „1“, kontroluje se, zda byl zadaný doklad již importován.
- format_output – pokud je nastavena hodnota „1“, provede se přeformátování výsledného XML do strukturovanější podoby.

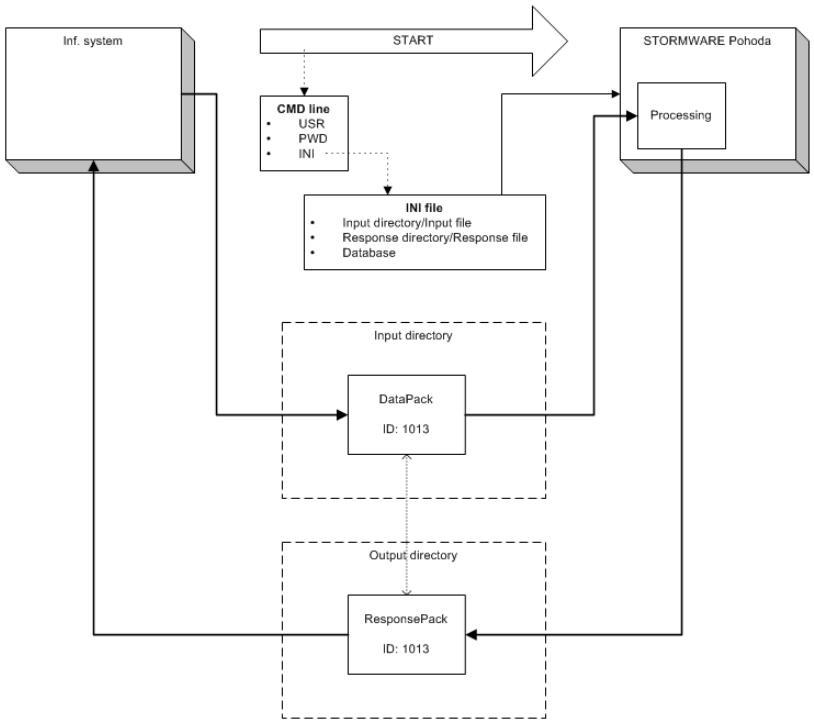
INI soubor, pro zpracování jednoho XML požadavku může mít následující strukturu.

```
[XML]
input_xml=Vstupni_xml_soubor
response_xml=Vystupni_xml_soubor
database=Nazev_database_ucetni_jednotky
check_duplicity=1
format_output=0
```

Při zpracovávání jednoho XML požadavku (tento způsob jsem používal ve svojí aplikaci) jsou povinné atributy input_xml a database. Ostatní atributy se již vyplňovat nemusí.

XML soubory s požadavkem mohou sloužit pro import či export do a z ES Pohoda. XML požadavky pro import jsou většinou obsáhlejší, neboť obsahují údaje, které se do Pohody importují. Požadavky pro export jsou pak o poznání kratší. Přehled dokladů pro import a export lze nalézt na webu společnosti Stormware.

Jakmile ES Pohoda XML požadavek zpracuje, vytvoří podle parametrů INI souboru XML odpověď. Z tohoto souboru lze potom zjistit informace o provedeném importu či exportu. Moje aplikace dokáže tyto soubory analyzovat a adekvátně na ně reagovat. Pokud se například nezdaří tvorba nových skladových karet, pošle Pohoda tuto informaci rovněž v XML odpovědi, moje aplikace ji analyzuje a vyhodnotí, že import příjemky nemůže dále pokračovat.



Obr. 13 XML komunikace s ES Pohoda

Na obrázku 13 můžete vidět mechanismus XML komunikace, který shrnuje vše, co bylo napsáno výše.

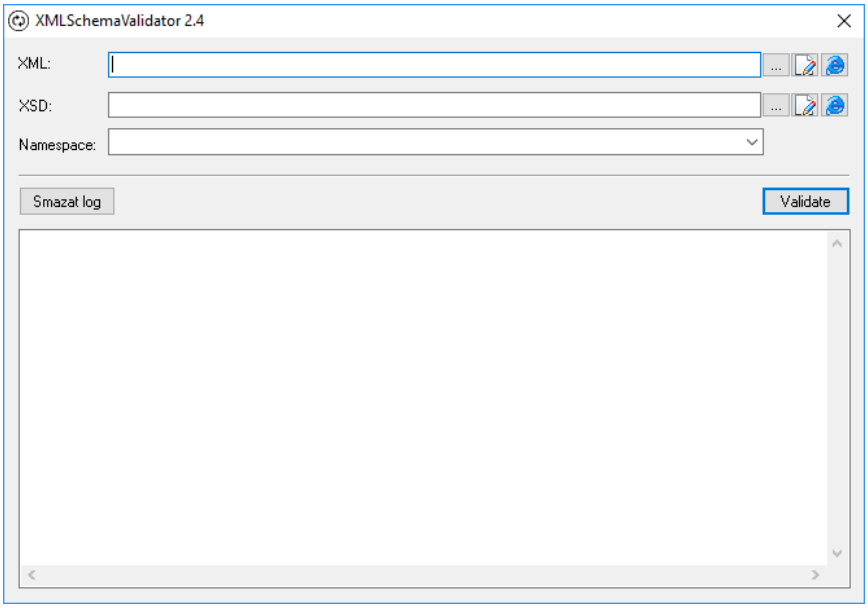
6.3.2 Šablony a XML Validator

Formáty XSD šablon[25] a ukázkové příklady XML souborů pro import[26] a export[27] dat do a z databází programu Pohoda, jsou veřejně přístupné na webu společnosti Stormware. XML soubory lze před jejich importem velmi snadno zkontrolovat pomocí validačního programu Stormware XML Validator.

[25] https://www.stormware.cz/xml/seznamschemat.aspx

[26] https://www.stormware.cz/xml/dokladyimport.aspx

[27] https://www.stormware.cz/xml/dokladyexport.aspx



Obr. 14 Stormware XML Validator

XML Validator má jednoduché rozhraní. V prvním řádku je potřeba vybrat vstupní XML soubor, který bude validován. Ve druhém řádku je potřeba vybrat XSD schéma, podle kterého bude provedena validace zadaného XML souboru. Pro validaci XML dokumentu dle struktury programu Pohoda je nutné definovat namespace. Tento řádek aplikace doplní automaticky po vybrání XSD šablony.

Jakmile jsou všechny řádky vyplněny, stačí stisknout tlačítko „Validate“ a aplikace začne s validací XML souboru. V případě, že XML soubor neodpovídá XSD šabloně, validátor na to uživatele upozorní.

6.4 Implementace

Jakmile jsem měl základní návrh struktury aplikace, začal jsem s jejím vývojem. Kapitoly jsem opět rozdělil tak, aby korespondovaly s jednotlivými funkčními částmi aplikace. U každé části přiblížím její funkcionalitu, případně nějakou část implementace, která je zajímavá.

6.4.1 Nalezení poslední nezpracované faktury

Hned v úvodu implementace jsem narazil na několik menších problémů. Potřeboval jsem projít web dodavatele a nalézt na něm poslední nezpracovanou fakturu. Překvapením pro mě bylo, když jsem zjistil, že webová stránka AT Computers je napsaná v ASP.NET. Jde o open-source framework, pro tvorbu webových aplikací, jejichž funkcionalita běží na straně serveru. Jakmile jsem otevřel zdrojový kód webu, viděl jsem následující kód.

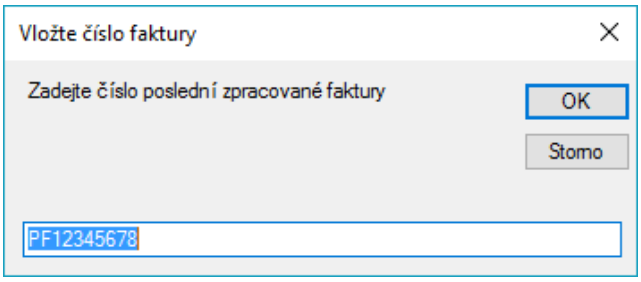
```
<html>
 <head>
```

```
  <title>ATC BusinessLink</title>
  <link rel="alternate" type="application/rss+xml" title="RSS
    www.atcomp.cz" href="http://www.atcomp.cz/rss.aspx" />
  <meta http-equiv='content-type' content='text/html; charset=utf-8'/>
  <link rel="icon" type="image/png" href="/img/abl-favicon.png"/>
  <link rel="apple-touch-icon" href="/img/apple-touch-icon.png" />
 </head>

 <frameset rows="174,*" frameborder="0" framespacing="0" id="hlavni">
  <frame src="header.aspx" name="vrch" id="vrch" frameborder="0"
   scrolling="no" marginwidth="0" marginheight="0" framespacing="0"/>
  <frame src="obsah.aspx" name="obsah" id="obsah" frameborder="0"
   scrolling="auto" marginwidth="0" marginheight="0" framespacing="0"
   />
 </frameset>
</html>
```

Jak si můžete všimnout, z tohoto kódu se příliš údajů vyčíst nedá. Stránka se skládá z hlavičky a dvou rámů. První reprezentuje záhlaví webu, druhý se plní obsahem. Chvíli mi trvalo, než jsem pochopil, jak tento web funguje a jak se dostat k požadovanému obsahu. Musel jsem nejprve najít a uložit rám, který se dynamicky plní obsahem. Ten už obsahoval běžné HTML tagy, které jsme očekával.

Nyní, když jsem již věděl, jak se dostat k obsahu webu, musel jsem aplikaci naučit, jak najít poslední nezpracovanou fakturu. Předtím však ještě krátká odbočka k tomu, jak aplikace získává číslo poslední zpracované faktury (k tomu, aby mohla najít poslední nezpracovanou, musí vědět, která se zpracovala jako poslední). Během počátečního vývoje jsem zadával číslo poslední zpracované faktury přímo do kódu a podle potřeb jsem ho měnil. Jakmile jsem však tuto část aplikace konzultoval se zadavatelem, uvědomil jsme si, že bude nutné, aby aplikace umožňovala nastavení čísla poslední zpracované faktury, podle potřeb uživatele. Do aplikace jsem přidal funkcionalitu, která při prvním spuštění toto číslo od uživatele vyžádá. Navíc, v nastavení aplikace je možnost toto číslo kdykoliv změnit.



Obr. 15 Nastavení čísla poslední zpracované faktury

Zadávání čísla poslední zpracované faktury jsem vyřešil pomocí dialogu. Toto nastavení můžete vidět na obrázku 15. Aplikace při zadávání neověřuje žádný konkrétní formát tohoto čísla, jelikož se může průběhu času měnit. Ověřuje se pouze, že je něco zadáno. Vždy musí být tudíž nějaké „číslo“ nastaveno.

Ještě předtím, než mohla aplikace začít na webu dodavatele hledat číslo poslední nezpracované faktury, musel jsem vyřešit, jak v kódu obecně pracovat s webovou stránkou a jak se dostat do částí webu, kde je nutné přihlášení. Pro práci s webovým obsahem jsem musel použít instanci třídy WebBrowser, která reprezentuje webový prohlížeč a umožňuje aplikaci procházet webový obsah. Problém byl ovšem v tom, jak se dostat do částí webu, kde je vyžadováno přihlášení. Věděl jsem, že bude nutné pomocí URL nějakým způsobem předat parametry, resp. přihlašovací údaje. Musel jsem tedy nastudovat problematiku předávání parametrů pomocí URL adresy v ASP.NET. V následujícím kódu můžete vidět výslednou URL adresu, pomocí které mohla třída WebBrowser pracovat s webem, jako by byla přihlášena.

```
http://www.atcomp.cz/presmerovani.aspx?txtName= +
My.Settings.AtCompName + &txtPassword= + My.Settings.AtCompPsw +
&goto=/faktury.aspx
```

Ihned za běžnou URL adresou webové stránky, která končí v tomto případě .aspx, jsem přidal ? (otazník), následovaný identifikátorem HTML tagu (txtName), znamínkem „rovná se“ a hodnotou, kterou jsem chtěl předat (údaj uložený v nastavení aplikace). Takovýmto způsobem jsem stránce předal přihlašovací jméno a heslo, nutné pro vstup do zákaznické sekce webu.

Za předaným heslem můžete vidět ještě klíčové slovo „goto“ a předanou hodnotu „/faktury.aspx“. Jakmile se třída WebBrowser na webu přihlásí, přejde do přehledu vydaných faktur. Mezi jednotlivými parametry je symbol „&“, jež slouží pro předávání více parametrů.

Nyní již zpět k získávání čísla poslední nezpracované faktury. Faktury na webu dodavatele jsou pro lepší přehlednost rozděleny na více stránek. Proto jsem aplikaci musel naučit, že nenajde-li fakturu na první stránce přehledu, musí přejít na další. V aplikaci je nastavena citlivost prohledávání, která stanovuje, kolik stránek přehledu se má prohledat, než bude oznámen neúspěch při hledání. Reakcí na neúspěšné hledání je otevření dialogu pro zadání čísla poslední zpracované faktury.

Hledání tedy spočívá v procházení HTML elementů a ověřování shodnosti čísla faktury s číslem poslední zpracované faktury. Jakmile je takovéto číslo nalezeno, uloží se číslo faktury, které jí předcházelo, tedy číslo poslední nezpracované faktury. Pokud není faktura na stránce nalezena, přejde aplikace na stránku další a hledání se opakuje. Hledání končí nalezením správného čísla faktury nebo prohledáním určitého počtu stránek. Tento proces je přehledně vyjádřen v diagramu aktivit na obrázku 10.

6.4.2 Nalezení dodacích listů svázaných s fakturou

V předchozím kroku aplikace hledá na základě čísla poslední zpracované faktury, číslo poslední nezpracované faktury. Jakmile je toto číslo nalezeno, aplikace si ho uloží. Kromě něj si musí rovněž uložit interní číslo faktury, což je jednoznačný identifikátor faktury na webu dodavatele, sloužící k provázání faktury s dodacími

listy. Jestliže je tedy hledané číslo faktury nalezeno a uloženo, spustí se hledání čísel dodacích listů, svázaných s touto fakturou. Pro hledání těchto čísel, používá aplikace následující kód.

```
'Podle interního čísla faktury najde svázané dodací listy
Private Sub findDeliveryNotesId(invoiceInternalNum As String)
Dim url As String = "http://www.atcomp.cz/presmerovani.aspx?txtName="
+ My.Settings.AtCompName + "&txtPassword=" + My.Settings.AtCompPsw +
"&goto=/historie_dokumentu.aspx$$cis=" & invoiceInternalNum &"$typ=FA"

Main.WebBrowser1.Navigate(url)
WaitForPageLoad()

'Před přidáním nových dodacích listů je stávající seznam vymazán
deliveryNotesIds.Clear()

Dim currDeliveryNotesWindow As HtmlWindow =
Min.WebBrowser1.Document.Window

'Projde všechny svázané doklady a uloží ty s označením HD, tedy dodací
listy
For Each frame As HtmlWindow In currDeliveryNotesWindow.Frames
  If frame.Name = "obsah" Then
    Dim elementCollection As HtmlElementCollection =
      frae.Document.GetElementsByTagName("Td")

    For Each element As HtmlElement In elementCollection
      If element.InnerText IsNot Nothing Then
        If element.InnerText.Contains("HD") Then
          deliveryNotesIds.Add(element.InnerText.ToString)
        End If
      End If
    Next
  End If
Next
End Sub
```

Třídě WebBrowser se předají potřebné parametry a pomocí parametru „goto=/historie_dokumentu.aspx“ je přesměrována na stránku se svázanými doklady faktury. Kromě přihlašovacích údajů se rovněž předá typ dokladu a interní číslo faktury, na základě kterého jsou procházeny svázané doklady správné faktury.

Dále jsou pak pomocí cyklu prohledány HTML elementy a ukládána čísla všech svázaných dokladů s označením „HD“, což je označení pro dodací listy. Menší nástrahou je to, že faktura může mít klidně více dodacích listů, ale stejně tak nemusí mít ani jeden. Proto je důležité postupovat při dohledávání dodacích dokladů opravdu důkladně.

```
#Region "Page Loading Functions"
 Private Sub WaitForPageLoad()
  AddHandler Main.WebBrowser1.DocumentCompleted, New WebBrowserDocu
   mentCompletedEventHandler(AddressOf PageWaiter)
```

```
  While Not pageready
   Application.DoEvents()
  End While
  pageready = False
 End Sub

 Private Sub PageWaiter(ByVal sender As Object, ByVal e As WebBrow
  serDocumentCompletedEventArgs)
  If Main.WebBrowser1.ReadyState = WebBrowserReadyState.Complete Then
   pageready = True
   RemoveHandler Main.WebBrowser1.DocumentCompleted, New WebBrowserDo
    cumentCompletedEventHandler(AddressOf PageWaiter)
  End If
 End Sub
#End Region
```

Důležitou součástí této části aplikace je rovněž volání metody WaitForPageLoad(). Tato metoda se při prohledávání webu stará o to, že je načten celý obsah stránky a nedojde tedy k přehlédnutí části obsahu, jež se ještě nestihl načíst. Tato metoda je důležitá, protože načítání obsahu může trvat různě dlouho, v závislosti na rychlosti internetového připojení.

6.4.3 Stažení nalezených dokladů

Jakmile má aplikace uloženo číslo poslední nezpracované faktury a k ní svázaných dodacích listů, musí tyto doklady stáhnout. Stahování neprobíhá přímo ze stránek dodavatele, ale z webu[28], který dodavatel rovněž spravuje. Z tohoto webu jdou doklady stáhnout ve formátu XML, což je velká výhoda při jejich prohledávání. Zpracování stažených dokladů se však bude věnovat další kapitola.

Ještě předtím, než budou doklady staženy, musí aplikace zajistit vytvoření pracovního adresáře, včetně všech jeho podadresářů. Aplikace toto zajišťuje před každým stahováním dokladů z webu dodavatele, ale i před každou komunikací s ES Pohoda. Pracovní adresář aplikace musí obsahovat následující podadresáře.

- Documents – slouží pro ukládání stažených dokladů z webu dodavatele, pro každou fakturu je zde vytvořena složka s číslem faktury.
- History – všechny zpracované doklady se přesunou do složky History.
- Ini – do této složky se ukládají INI soubory, zprostředkovávající komunikaci s ES Pohoda.
- Request – aplikace do této složky generuje XML požadavky pro ES Pohoda.
- Response – ES Pohoda ukládá XML odpovědi právě do této složky.

Pro zajištění existence všech potřebných adresářů má aplikace třídu WorkingDirManager. Tato třída pomocí svých metod zajistí, že nebytné adresáře budou dostupné. Hlavní metoda třídy WorkingDirManager vypadá následovně.

```
'Metoda vytvoří pracovní adresář a všechny vnořené adresáře
```

[28] http://ws.atcomp.cz/dokumenty.asmx

```
Public Sub ensureWorkingDirs()
 If Not Directory.Exists(Application.StartupPath() + "\AIPWorkingDir")
  Then
  MkDir(Application.StartupPath() + "\AIPWorkingDir")
 End If

 ensureDirIni()
 ensureDirRequest()
 ensureDirResponse()
 ensureDirHistory()
 ensureDirDocuments()
End Sub
```

Tento kód ověří existenci všech potřebných adresářů a pokud neexistují, vytvoří je. Hlavní metoda ensureWorkingDirs() obsahuje volání metod ensureDirIni(), ensureDirRequest() a dalších, které zajistí existenci všech podadresářů hlavního pracovního adresáře.

Při stahování dokladů jsem se potýkal s jediným problémem, odladit správně jazykovou sadu stažených dokladů. Při prvních pokusech se mi doklady stahovaly s jinou jazykovou sadou, která neobsahovala českou diakritiku. Problém se mi podařilo jednoduše odladit, stačilo nastavit třídě WebClient, kterou jsem používal pro stahování, kódování UTF-8, které českou znakovou sadu podporuje.

Aplikace musí stahovat 2 typy dokladů, faktury a dodací listy. Implementace stahování těchto 2 typů dokladů se mírně liší. Za prvé, faktura může mít klidně více dodacích listů, stahování je tedy opakováno v cyklu, podle počtu uložených čísel dodacích listů. Dalším rozdílem je URL adresa pro stažení dokladu. Parametr předaný v URL adrese označuje, jestli se bude stahovat faktura nebo dodací list. Kromě toho se pak v URL adrese předají již klasicky přihlašovací údaje, které opravňují ke stažení daného dokladů

Následující metoda slouží pro stažení dodacích listů, které jsou s fakturou svázány.

```
'Metoda pro stažení dodacích listů
Private Sub dwlDeliveryNotes()
 Dim counter As Integer = 1
 For Each num As String In deliveryNotesIds
  Dim urlDeliveryNote As String =
   "http://ws.atcomp.cz/dokumenty.asmx/DodaciList?Uzivatel=" &
   My.Settings.AtCompName & "&Heslo=" & My.Settings.AtCompPsw &
   "&Cislo=" & num
  Dim xmlDeliveryNote As String = wc.DownloadString(urlDeliveryNote)
  File.WriteAllText(dwlPath & "\DodaciList" & counter.ToString &
   ".xml", xmlDeliveryNote)
  counter = counter + 1
 Next
End Sub
```

Metoda pro stažení dodacích listů prochází dynamické pole deliveryNotesIds, ve kterém jsou uložena čísla dodacích listů, a stahuje jednotlivé dodací listy do pracovního adresáře.

6.4.4 Nalezení údajů ve stažených dokladech

V kódu aplikace jsem vytvořil datové struktury, které odpovídají údajům na fakturách a dodacích listech. Údaje je potřeba extrahovat ze stažených dokladů a do těchto datových struktur je uložit. Nyní přijde vhod, že bylo možné doklady stáhnout ve formátu XML. Jazyk VB.NET totiž obsahuje třídu XmlReader, která je přímo určena pro práci s XML soubory. Tato třída obsahuje metody pro hledání uzlů a elementů, čtení hodnot atributů, ale i tvorbu nových XML souborů.

```
Using xr As XmlReader = XmlReader.Create(filePath)
 While xr.Read()
  If xr.NodeType = XmlNodeType.Element Then
   If xr.Name = "faktura" Then
    inv.CisloFaktury = xr.GetAttribute("cislo")
    inv.Text = "Faktura č. " & inv.CisloFaktury
    'nastaví se podle zvoleného střediska v aplikaci
    inv.Stredisko = ""
    'ukládá se aktuální datum
    inv.Datum = Date.Now.ToString("yyyy-MM-dd")
   ElseIf xr.Name = "zbozi" Then
    Dim invItem As New PolozkaPrijemky
    invItem.Name = xr.GetAttribute("popis")
    invItem.Quantity = xr.GetAttribute("pocet")
    invItem.Unit = xr.GetAttribute("jednotka")
    ...

    If invItem.InternalType = "1" Then
     invItem.UnitPrice = xr.GetAttribute("kusova_cena")
    Else
     invItem.UnitPrice = xr.GetAttribute("castka")
    End If

    inv.Polozky.Add(invItem)
   End If
  End If
 End While
End Using
```

Podle cesty, předané v parametru metody, se prohledá konkrétní faktura ve formátu XML, uložená v pracovním adresáři. Do datových struktur aplikace jsou uloženy všechny potřebné údaje o faktuře a jejich položkách. K faktuře se přiřadí aktuální datum, aby bylo jasné, kdy byla zpracována. Středisko je k faktuře přiřazeno až podle nastavení v aplikaci.

Jakmile jsou uloženy všechny položky faktury, prohledají se ještě stažené dodací listy a k položkám se doplní některé údaje potřebné pro import příjemky, jež lze získat pouze z dodacích listů.

6.4.5 Ověření existence skladových karet

Aplikace má již uloženy všechny potřebné údaje ze stažených dokladů. Před importem příjemky je však nutné ověřit ještě jednu důležitou věc. Zdali všechny zásoby,

figurující na faktuře, mají vytvořenou skladovou kartu. Ověření existence skladových karet probíhá následovně.

1. Vytvoření XML požadavku pro export skladových karet
2. Vytvoření INI souboru
3. Poslání XML požadavku do ES Pohoda, ta vrátí XML odpověď
4. Ověření existence skladových karet, pomocí XML odpovědi

Ve vytvořeném XML požadavku figurují kódy zásob, které aplikace získala ze stažených dokladů. INI soubor zprostředkovává XML komunikaci s ES Pohoda. INI soubor je Pohodě předán pomocí příkazového řádku. Jakmile ES Pohoda vrátí XML odpověď, aplikace ji projde, vyhodnotí a upraví jednotlivé položky faktury. Kód, starající se o přečtení XML odpovědi a nastavení položek faktury, vypadá následovně.

```
'vzhledá v Pohodě skladové zásoby podle EANu
Public Sub generateRequestStockItemsByEan(invoice As Prijemka)
 'nastavení formátu a cesty k XML souboru
 Dim pathRequestStockItemsById As String = Application.StartupPath() +
  "\AIPWorkingDir\Request\requestStockItemsByEan.xml"
 Dim wReqStockItemsById As New
  Xml.XmlTextWriter(pathRequestStockItemsById, Nothing)
  wReqStockItemsById.Formatting = Xml.Formatting.Indented
  wReqStockItemsById.WriteStartDocument()

 'hlavička XML souboru
 wReqStockItemsById.WriteStartElement("dat", "dataPack",
  "http://www.stormware.cz/schema/version_2/data.xsd")
 wReqStockItemsById.WriteAttributeString("xmlns", "stk", Nothing,
  "http://www.stormware.cz/schema/version_2/stock.xsd")
 wReqStockItemsById.WriteAttributeString("xmlns", "ftr", Nothing,
  "http://www.stormware.cz/schema/version_2/filter.xsd")
 wReqStockItemsById.WriteAttributeString("xmlns", "lStk", Nothing,
  "http://www.stormware.cz/schema/version_2/list_stock.xsd")
 wReqStockItemsById.WriteAttributeString("xmlns", "tzp", Nothing,
  "http://www.stormware.cz/schema/version_2/type.xsd")
 wReqStockItemsById.WriteAttributeString("id", "ID")
 wReqStockItemsById.WriteAttributeString("ico",
  My.Settings.JednotkaIco)
 wReqStockItemsById.WriteAttributeString("application",
  My.Settings.JednotkaStwph)
 wReqStockItemsById.WriteAttributeString("version", "2.0")
 wReqStockItemsById.WriteAttributeString("note", "Request - Stock
  items by ID")
 wReqStockItemsById.WriteStartElement("dat", "dataPackItem",
  "http://www.stormware.cz/schema/version_2/data.xsd")
 wReqStockItemsById.WriteAttributeString("id", "ID")
 wReqStockItemsById.WriteAttributeString("version", "2.0")
 wReqStockItemsById.WriteStartElement("lStk", "listStockRequest",
  "http://www.stormware.cz/schema/version_2/list_stock.xsd")
 wReqStockItemsById.WriteAttributeString("version", "2.0")
 wReqStockItemsById.WriteAttributeString("stockVersion", "2.0")
```

```
 'přidání všech položek faktury
 For Each item As PolozkaPrijemky In invoice.Polozky
  wReqStockItemsById.WriteStartElement("lStk", "requestStock",
   "http://www.stormware.cz/schema/version_2/list_stock.xsd")
  wReqStockItemsById.WriteStartElement("ftr", "filter",
   "http://www.stormware.cz/schema/version_2/filter.xsd")
  wReqStockItemsById.WriteStartElement("ftr", "EAN",
   "http://www.stormware.cz/schema/version_2/filter.xsd")
  wReqStockItemsById.WriteValue(item.EAN)
  wReqStockItemsById.WriteEndElement()
  wReqStockItemsById.WriteEndElement()
  wReqStockItemsById.WriteEndElement()
 Next

 wReqStockItemsById.WriteEndElement()
 wReqStockItemsById.WriteEndElement()
 wReqStockItemsById.WriteEndElement()
 wReqStockItemsById.WriteEndDocument()
 wReqStockItemsById.Close()
End Sub
```

Jakmile je XML požadavek, vygenerovaný tímto kódem, poslán do ES Pohoda, vrátí Pohoda XML odpověď. V odpovědi jsou informace o zásobách, jejichž kód byl zaslaném XML požadavku. Pokud informace o některé zásobě chybí, znamená to, že tato zásoba nemá v ES Pohoda skladovou kartu. Informace o tom, jestli skladová karta existuje či ne, je uložena ke každé položce faktury.

V konečných fázích vývoje jsem narazil na jeden, poměrně zásadní problém. Některé položky faktury mohou mít již vytvořenou skladovou kartu, ovšem s odlišnou cenou, než je na stažené faktuře. V takovýchto situacích se nastaví cena skladové karty podle ceny na stažené faktuře. V jiných situacích opět existuje skladová karta zásoby, ale je k ní přiřazena jiná svázaná zásoba než na faktuře. V takových případech bude uživatel o této skutečnosti informován, a záleží pouze na jeho rozhodnutí, jestli skladovou kartu opraví.

6.4.6 Vytvoření chybějících skladových karet

V předchozím kroku aplikace ověřila, které položky faktury již mají skladovou kartu a kterým skladová karta stále chybí. Následně aplikace vygeneruje XML požadavek, sloužící pro vytvoření chybějících skladových karet. Kód aplikace, pro generování XML požadavku pro import chybějících skladových karet, najdete v příloze B. Pro komunikaci s ES Pohoda musí aplikace rovněž vytvořit INI soubor. Povel ke zpracování INI souboru je do ES Pohoda poslán pomocí příkazové řádky. Kód pro spuštění příkazové řádky vypadá takto.

```
Public Sub runIniFile(iniFileName As String, logName As String, logPsw
As String)
 Dim IniFilePath As String = Application.StartupPath() +
  "\AIPWorkingDir\Ini\" & iniFileName & ".ini"
 Process.Start(My.Settings.PohodaPath, "/XML """ + logName + """ """ +
```

```
  logPsw + """ """ + IniFilePath + """")
End Sub
```

Jde o jednoduchou syntaxi, kde metoda Start() třídy Process, pošle instrukci do příkazové řádky. Jako parametry jsou předány cesta k ES Pohoda, cesta k INI souboru a přihlašovací údaje uživatele s právy na XML komunikaci.

Při vývoji aplikace jsem se snažil, aby byla co možná nejvíce automatizovaná, v některých situacích to však nebylo zcela možné. V případě, že některá zásoba nemá vytvořenou skladovou kartu, požádá aplikace uživatele o její vytvoření, pomocí následujícího okna.

Obr. 16 Okno aplikace pro vytvoření nové skladové karty

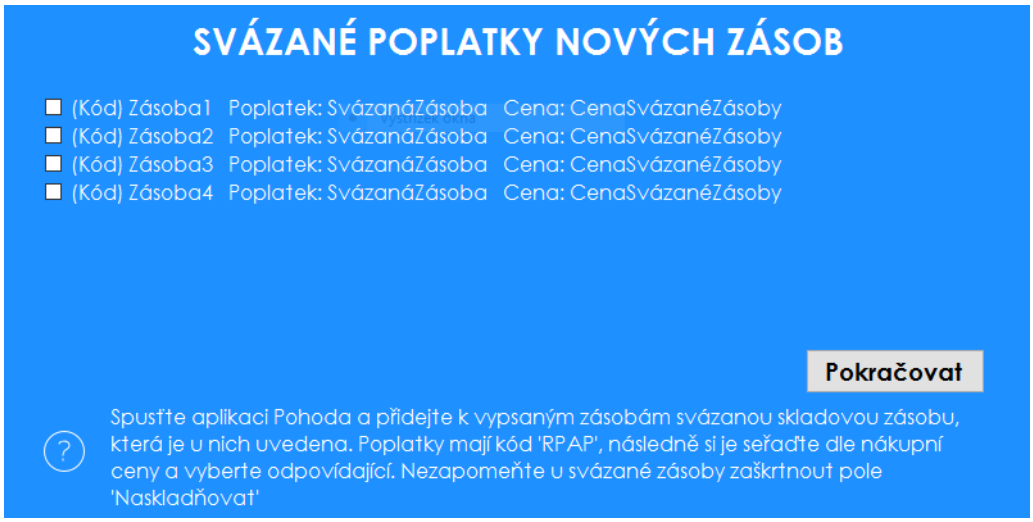
Aplikace automaticky vyplní pole údaji ze stažené faktury, chybějící údaje musí ovšem uživatel doplnit sám. Většinou je potřeba doplnit cenovou skupinu a skladové členění zásoby. Členění skladů získává aplikace při svém spuštění přímo z účetní jednotky. Všechny vkládané údaje jsou povinné, a proto bez jejich vyplnění aplikace nemůže pokračovat. Na chybějící hodnoty aplikace uživatele upozorní zčervenáním příslušného pole, jak můžete vidět na obrázku 16 u pole Kód.

Při vytváření skladových karet pomocí XML komunikace jsem narazil na jeden vážný nedostatek. Některé zásoby mohou mít k sobě svázanou jinou zásobu, např. nějaký poplatek. Při obsluze ES Pohoda běžnou cestou, je možné svázat jednu zásobu s druhou, potom se při práci s první zásobou automaticky doplní i ta druhá. Komunikace pomocí XML bohužel tuto možnost postrádá.

Tento problém jsem nejprve zkoušel vyřešit sám, pomocí dostupných zdrojů, ale bohužel jsem na žádné řešení nepřišel. Následně jsem kontaktoval technickou podporu společnosti Stormware. Servisní technik si vzal na řešení nějaký čas, a nakonec přišel s odpovědí, že tato funkcionalita zkrátka není podporovaná a doporučil mi podat požadavek na její zapracování. Osobně mě velice překvapilo,

že tato funkcionalita v XML komunikaci chybí, přitom se nejedná o nic neobvyklého, svázané položky jsou běžné u spousty zásob. Nicméně, poslal jsem požadavek na zapracování funkcionality, a čekal jsem. Mezitím jsem do aplikace zapracoval dočasné řešení. V případě, že byla vytvořena skladová karta zásoby, která má podle faktury svázanou zásobu, aplikace o této skutečnosti uživatele informuje vygenerováním textového souboru se seznamem zásob a jejich svázaných položek. Je pak pouze na uživateli, aby tyto svázané zásoby k vytvořeným skladovým kartám připojil.

O více jak 2 měsíce a nespočet aktualizaci ES Pohoda později, mnou vyžádaná funkcionalita nebyla stále zapracována. Ponechal jsem proto v aplikaci ono dočasné řešení a informoval jsem o této skutečnosti zákazníka. Po konzultaci se zaměstnankyní, která má import příjemek na starosti, jsem se rozhodl zapracovat funkcionalitu, která zobrazí zásoby a jejich svázané položky přímo v aplikaci.



Obr. 17 Přehled svázaných položek zásob

Tato nově přidaná funkcionalita nejenže ušetří uživatele zdlouhavého otevírání vygenerovaného textového souboru, ale navíc je opatřena zaškrtávacími políčky, takže má uživatel přehled, kterým zásobám již svázané položky přidal a kterým ne. Aplikace vyžaduje, aby před pokračováním byla zaškrtnuta všechna políčka, což přinutí uživatele opravdu si jednotlivé řádky projít a zkontrolovat.

Jakmile Pohoda dokončí požadavek na vytvoření skladových karet, vrátí aplikaci XML odpověď. Pokud během vytváření skladových karet nastal nějaký problém, aplikace to zjistí a informuje o tom uživatele.

6.4.7 Vytvoření příjemky ze získaných údajů

Jakmile existují skladové karty pro všechny položky faktury, může aplikace začít s importem samotné příjemky. Z uložených údajů musí vygenerovat XML požadavek pro import příjemky, podle předepsané šablony. Kvůli generování tohoto XML požadavku, ale i všech předchozích, jsem musel důkladně nastudovat problemati-

ku generování XML dokumentů v jazyce VB.NET. Nakonec se mi podařilo pochopit, jakým způsobem docílit požadovaného formátu a dokumenty se začaly generovat tak, jak měly.

V práci bylo již nastíněno, jakým způsobem probíhá XML komunikace s ES Pohoda. Pro každou XML komunikaci s ES Pohoda je nutné mít samotný XML soubor, který Pohodě „řekne“, co od ní chceme. Tento XML soubor musí přesně dodržovat definovanou strukturu. Kromě tohoto souboru je potřeba také INI soubor, který samotnou komunikaci zprostředkuje. Při spouštění příkazu v příkazové řádce Windows se nevolá přímo XML soubor, ale právě tento INI soubor, který obsahuje všechny důležité informace pro komunikaci s ES Pohoda.

INI soubor může zpracovat buď jeden XML požadavek, nebo všechny XML požadavky umístěné v definovaném adresáři. První povinná informace v INI souboru je cesta buď k jednomu konkrétnímu XML požadavku nebo k celému adresáři, obsahujícímu více XML požadavků. Druhým povinným atributem je název databáze ES Pohoda, které bude XML požadavek zaslán. Ostatní atributy, jsou nepovinné. Aplikace přidává do INI souborů ještě cestu, do které se mají ukládat XML soubory s odpovědí z ES Pohoda. Pokud není tato cesta definovaná, budou se soubory automaticky ukládat do stejného adresáře, jako byl XML soubor s požadavkem.

```
'Univerzální generátor INI souborů
Public Sub generateIniFile(name As String)
 Dim filePath As String = Application.StartupPath() +
  "\AIPWorkingDir\Ini\ini" + name + ".ini"
 Dim requestFilePath As String = """ + Application.StartupPath() +
  "\AIPWorkingDir\Request\request" + name + ".xml"""
 Dim responseFilePath As String = """ + Application.StartupPath() +
  "\AIPWorkingDir\Response\response" + name + ".xml"""

 Using sw As New System.IO.StreamWriter(filePath, False)
  sw.WriteLine("[XML]")
  sw.WriteLine("input_xml=" + requestFilePath)
  sw.WriteLine("response_xml=" + responseFilePath)
  sw.WriteLine("database=" + My.Settings.JednotkaStwph)
 End Using
End Sub
```

Metoda si nejprve nachystá do proměnných cesty pro INI soubor a soubory XML požadavku a odpovědi. Následně je pomocí třídy StreamWriter vytvořen INI soubor, který má požadovanou strukturu. Metoda generuje INI soubor podle předaného parametru, což umožňuje její použití u různých INI souborů. Kód pro samotné generování XML souboru, pro import příjemky je poměrně dlouhý, proto jsem ho umístil do přílohy A.

Nyní se ale už dostávám k samotnému importu příjemky do ES Pohoda. Aplikace již tedy získala potřebné údaje z dokladů stažených z webu dodavatele, vytvořila chybějící skladové karty a nyní díky tomu může vytvořit novou příjemku.

```
Private Sub createReceipt()
```

```
 fileGenerator.generateRequestImportReceipt(mLocalReceipt, mAtCompId,
  mCentre)
 fileGenerator.generateIniFile("ImportReceipt")

 If suppProc.fileFound(Application.StartupPath() +
  "\AIPWorkingDir\Request\requestImportReceipt.xml", 5) And
  suppProc.fileFound(Application.StartupPath() +
   "\AIPWorkingDir\Ini\iniImportReceipt.ini", 3) Then
  Dim importReceiptIniFilePath As String = Application.StartupPath() +
   "\AIPWorkingDir\Ini\iniImportReceipt.ini"
  Process.Start(My.Settings.PohodaPath, "/XML """ +
   My.Settings.PohodaName +
   """ """ + My.Settings.PohodaPsw + """ """+ importReceiptIniFilePath
   + """")

  My.Settings.LastProcessedInvoice = invoiceNum
  My.Settings.Save()
 Else
  MessageBox.Show("CHYBA: Chybí soubory INI a XML Request pro tvorbu
   nové příjemky!")
 End If
End Sub
```

Tento kód vytvoří XML požadavek a INI soubor pro komunikaci s ES Pohoda. Následně si ověří, jestli se tyto soubory skutečně vytvořily a spustí INI soubor v příkazové řádce. Na závěr si uloží číslo poslední zpracované faktury, aby od ní mohl při dalším spuštění pokračovat.

6.5 Testování a nasazení aplikace

Testování je široce obsáhlá disciplína, na kterou se dá pohlížet z různých úhlů pohledu. Jde o velice důležitou etapu vývoje softwaru, jelikož ověřuje, zdali bylo splněno to, co bylo zadáno. Cílem testování je vyhledat chyby a zajistit jejich nápravu.

6.5.1 Testování v průběhu vývoje

Testování aplikace započalo již během jejího vývoje. Před připojením každé nové části funkcionality, k již fungujícímu celku, jsem ji vždy testoval nejprve samostatně. Jelikož jednotlivé části aplikace nebyly sami o sobě příliš komplexní, prováděl jsem toto testování pouhým ověřením požadované funkcionality.

Po přidání nové funkcionality byla pak aplikace testována rovněž jako celek. Bylo nutné ověřit, zdali nově přidaná funkcionalita plní v kontextu celé aplikace přesně to, co se od ní očekávalo a jestli správně komunikuje s ostatními komponentami.

V této fázi vývoje probíhalo testování pouze v testovacích podmínkách, kde nebyla ohrožena žádná důležitá data. Byl mi přidělen přístup na firemní testovací server, který jsem mohl používat dle vlastních potřeb. V tomto prostředí jsem měl k dispozici vše nezbytné, včetně plně funkčního ES Pohoda. Kromě toho, že nešlo o reálná data, bylo prostředí téměř identické s tím skutečným.

6.5.2 Testování kompletní funkcionality

Jakmile byl dokončen vývoj základní funkcionality aplikace, ověřoval jsem její správnou funkcionalitu jako celku, tedy jestli spolu jednotlivé části aplikace správně komunikují a vše dohromady pracuje, jak má. V této fázi vývoje vše probíhalo stále na testovacím serveru, jelikož jsem si nemohl dovolit riskovat narušení ostré účetní jednotky.

Dalším krokem bylo, že jsem si domluvil schůzku se zaměstnankyní, která se o import příjemek stará. Představil jsem jí aplikaci, vysvětlil, jak funguje a nechal jí nějaký čas na vyzkoušení. Řekl jsem jí, že mě zajímá několik věcí:

- situace, které mohou při importu nastat
- funkcionalita, která ji v aplikaci chybí
- představa o vzhledu aplikace

O tom, jaké situace mohou nastat a co je potřeba v kódu ošetřit jsem měl představu již před vývojem aplikace. Předtím jsem tyto informace získával především od majitele firmy, který má o této problematice rovněž dobrý přehled. Důvodem, proč jsem tyto informace znovu zjišťoval bylo především to, že jsem si chtěl před testováním aplikace osvěžit stávající a získat nové znalosti o tom, co všechno může nastat. Ani informace získané ode dvou osob nemusí být kompletní, ale určitě je lepší se zeptat alespoň dvou osob než jedné, která může některé věci opominout. Všechny získané znalosti jsem do aplikace zakomponoval.

Jelikož byla tato aplikace vytvářena čistě z praktických důvodů, nebyly na ni kladeny příliš velké nároky a nečekala se od ní příliš velká přidaná hodnota nad potřebnou základní funkcionalitu. Jednoduše řečeno, důležité bylo to, aby splnila, co měla, tedy import příjemek do ES Pohoda. Zaměstnankyně starající se o import příjemek, neměla příliš představu o tom, co je možné a co ne. Nakonec se mi přece jen svěřila s několika požadavky, které by bylo dobré do aplikace implementovat.

- ukládání přihlašovacího jména
- možnost nastavit středisko, provádějící import
- možnost změnit účetní jednotku
- změnit způsob výpisu svázaných zásob

Ukládání přihlašovacího jména byla maličkost, která jistě zpříjemní přihlašování každému, kdo s ním přichází do styku denně.

Požadavek na změnu střediska, provádějícího import je rovněž opodstatněný. K aplikaci může být přihlášen jeden uživatel, ale import příjemky může být prováděn klidně pod jiným střediskem, které reprezentuje jiného uživatele.

Třetí požadavek má rovněž dobrý důvod. Možnost měnit účetní jednotku je důležitá v případě přechodu na jinou účetní jednotku. Taková situace může nastat i ve stejné firmě, ale především v případě, kdy by měla být aplikace nasazena v jiné firmě. Původní smělé plány o takovéto expanzi mluvili, reálně je však tato situace stále poměrně daleko.

Poslední požadavek souvisí s omezenými možnostmi XML komunikace s ES Pohoda. Při vytváření skladových karet není možné pomocí XML komunikace ke skladové kartě přidat svázanou zásobu. Byl zadán požadavek na přidání této funkcionality na firmu Stormware a bylo nasazeno dočasné řešení. V případě, že měla být ke skaldové kartě přidána svázaná zásoba, uživatel aplikace byl o tom informován seznamem svázaných zásob v textovém souboru. Tato forma prezentace byla poněkud nepraktická a společně jsme se domluvili na tom, že by bylo vhodné tento přehled zakomponovat přímo do aplikace.

Požadavky na vzhled nebyly příliš odlišné od těch původních. Hlavní důraz má být kladem především na jednoduché a přehledné ovládání aplikace.

6.5.3 Nasazení finální verze

Jakmile jsem do aplikace zakomponoval získané požadavky, prováděl jsem její testování stále ještě v izolovaném prostředí na testovacím serveru. Potřeboval jsem aplikaci odladit pro všechny možné situace, o kterých jsem se během vývoje dozvěděl.

Jakmile jsem si myslel, že jsem odladil všechny známé nedostatky, potřeboval jsem aplikaci vyzkoušet v reálném prostředí. Opět jsem seznámil zodpovědnou zaměstnankyni s aktuálním stavem aplikace a zasvětil ji do jejího používání. Import příjemek není aktivita, která by trvala celý den, a tak jsem si vyhradil čas, abych mohl zaměstnankyni s importem pomoci a společně s ní opravit případné chyby.

Během testování na ostré účetní jednotce probíhalo vše již relativně bez problémů. Bylo potřeba odladit několik maličkostí, nešlo však o závažné chyby, které by výrazným způsobem narušovaly běh aplikace.

6.5.4 Podpora aplikace po nasazení

Podpora z mojí strany neskončila ani po nasazení aplikace do ostrého provozu. Jelikož ve firmě pracuji, přicházím s jejími zaměstnanci neustále do styku. Díky tomu mohu konzultovat aktuální problémy aplikace a obratem je řešit.

Přestože ve firmě pracuji a mohu tak odpovídat na případné dotazy osobně, rozhodl jsem se sepsat jednoduchý manuál, který provede uživatele procesem a naučí ho s touto aplikací pracovat. Aktuálně na tomto manuálu stále pracuji. Výhodou vytvoření manuálu je to, že nebudu muset v práci s aplikací zaškolovat další zaměstnance. Důležité informace budou obsaženy v manuálu, případně si je budou moci zaměstnanci předat mezi sebou. Jako dokumentace aplikace poslouží tato diplomová práce, ve které jsou obsaženy i všechny technické detaily týkající se vývoje aplikace.

7 Diskuze

V této kapitole kriticky zhodnotím vlastní řešení, porovnám jej s existujícími řešeními, která jsem v práci analyzoval a uvedu směr, kterým by se vývoj mého řešení mohl ubírat v budoucnu.

7.1 Porovnání vlastního řešení s konkurencí

V teoretické části práce bylo představeno několik komerčních řešení. Při jejich analýze bylo dosaženo závěru, že aplikace, která by zcela odpovídala zadání, vlastně neexistuje. Tento závěr dává smysl, jelikož zadání této práce je natolik specifické, že kdyby takovéto řešení již existovalo, firma by o něm nejspíše věděla. Z tohoto důvodu, bylo nutné vytvořit řešení nové. Jelikož se nepodařilo nalézt přímou konkurenci vyvíjené aplikace, byla v práci analyzována taková existující řešení, která se vyvíjené aplikaci alespoň přibližují, nebo s ní mají něco společné.

Řešení plusSystem od společnosti IT Future, jako jediné z analyzovaných řešení, neodpovídá žádné části funkcionality vyvíjené aplikace. I přesto, že jde o komplexní balík aplikací, žádná z nich se nevěnuje importu příjemek do ES Pohoda. Důvodů, proč jsem toto řešení analyzoval, je několik. Jde o komplexní řešení, které funguje na platformě Microsoft Windows a pracuje převážně s ekonomickým systémem Pohoda. Snaží se uživateli usnadnit činnosti, týkající se ekonomické aktivity podniku. Navíc, jde o jediné z analyzovaných řešení, které jsem si mohl skutečně vyzkoušet. Jelikož je funkcionalita aplikací plusSystem a mojí aplikace poněkud odlišná, není možné z tohoto úhlu pohledu aplikace porovnat. Po stránce vizuální se produkty plusSystem drží vzhledu, definovaného aplikací Pohoda. Jde o lehce zastaralý vzhled, který se víc jak na příjemné uživatelské rozhraní soustředí na funkčnost a praktičnost. Jak již bylo napsáno dříve, tomuto vzhledu jsem se ve svojí aplikaci snažil vyhnout. Rovněž mi šlo o co možná nejlepší funkčnost zvoleného vzhledu, navíc jsem se však soustředil také na moderní, jednoduchý a přehledný design.

Zbylá tři řešení, tedy Infomatic, Qinve a FlexiInvoice, již mají s vyvíjenou aplikací společnou alespoň část své funkcionality. Jejich společným znakem je to, že umožňují zpracování faktur od jakéhokoliv dodavatele. Tyto aplikace umožňují vkládání faktur pomocí jejich naskenování a následného vložení souboru ve formátu PDF či JPG. Na jednu stranu, získávání faktur není u těchto aplikací omezeno pouze na jednoho dodavatele, na stranu druhou, tento způsob získávání faktur může být časově náročný. Moje aplikace se soustředí na zpracování faktur pouze od jednoho dodavatele, toto zpracování ovšem probíhá zcela bez nutnosti zásahu uživatele, což uživateli ušetří čas. Omezení pouze na jednoho dodavatele, jenž má moje aplikace, se může zdát na první pohled jako velký nedostatek. Jelikož však bylo toto omezení již součástí zadání, nelze v tomto případě snad ani o omezení mluvit.

Řešení Infomatic poskytuje uživateli komplexní služby, od digitalizace dokumentů, až po procesní řízení. Aplikace Qinve potom poskytuje pouze možnost digi-

talizace dokumentů. Vyvíjená aplikace bude tvořit jakýsi kompromis mezi těmito dvěma řešeními. Bude uživateli poskytovat širší funkcionalitu, než je pouze digitalizace dokumentů, na druhou stranu však nepůjde o tak mohutné řešení jako je Infomatic.

Aplikace Infomatic a Qinve umožňují uživateli import digitalizovaných dokladů do různých ekonomických softwarů. Například u aplikace Qinve jsou to Helios, Pohoda, K2, Abra a další. Po této stránce je moje aplikace omezena, jelikož umožňuje import dokladů pouze do ES Pohoda. Stejně jako u omezení dodavatele, jde o funkcionalitu, která byla již součástí zadání. Rovněž se tedy nedá příliš mluvit o nějakém vážném omezení.

Všechna analyzovaná řešení, až na FlexiInvoice, fungují jako aplikace běžící přímo na pracovní stanici. FlexiInvoice, jako jediné, je vytvořeno jako webová aplikace. Toto zpracování přináší na jednu stranu výhody, v podobě snadného nasazení a používání. Na stranu druhou, nutnost internetového připojení může být v některých případech svazující. Moje řešení je zpracováno jako klasická aplikace, běžící na pracovní stanici. Ke spuštění aplikace stačí pouze spouštěcí .exe soubor. Jelikož však aplikace při svém fungování dohledává nezpracované faktury na webu dodavatele, je k jejímu správnému používání internetové připojení vyžadováno.

7.2 Zhodnocení vlastního řešení

Vytvořená aplikace splňuje základní funkcionalitu, kterou zadavatel očekával. Dokáže si sama obstarat nezpracované doklady z webu dodavatele a provést proces, který bylo předtím nutné provádět zcela manuálně, téměř zcela bez zásahu člověka. Plně automatizováno není pouze vytváření nových skladových karet, kde aplikace vyžaduje zásah uživatele. V této fázi aplikace potřebuje zadání údajů, které nemůže sama jednoznačně zjistit.

Při získávání požadavků zadavatel jasně stanovil, že výsledné řešení má fungovat pro dodavatele AT Computers a pro ES Pohoda. Tohoto zadání jsem se při implementaci držel a nesoustředil jsem se na to, aby měla aplikace co možná nejširší použití. I když aplikace podporuje pouze jednoho dodavatele a jeden ekonomický systém, snažil jsem se toto omezení využít jako její výhodu. Díky tomu, že aplikace zpracovává doklady pouze od jednoho dodavatele, dovoluje jí to se více soustředit na automatizaci získávání dokladů z webu dodavatele. Z použití jediného ekonomického softwaru se mi žádnou výraznou výhodu získat nepodařilo. Musím ale říci, že je aplikace pro tento ekonomický software alespoň dobře optimalizována.

Nedostatkem aplikace je tedy její omezení na jednoho dodavatele a jeden ekonomický systém. Dalším omezením je nutnost zásahu uživatele při vytváření skladových karet. Toto omezení by bylo vhodné co nejvíce omezit. Hlavní výhodou aplikace je pak její zcela automatizované získávání dokladů od dodavatele. Co se týká bezpečnosti, aplikace používá přihlašovací údaje z ES Pohoda. Pro účely aplikace tedy nebylo nutné zajišťovat tvorbu nových uživatelských účtů. Díky uživatelským účtům je možná jednoznačná identifikace uživatelů, používajících apli-

kaci. Spolehlivost aplikace je poměrně dobrá. Během vývoje i po dokončení aplikace probíhalo důkladné testování, které odhalilo většinu závažných chyb. V aplikaci by bylo ještě vhodné doladit systém logování jejich aktivit, což by usnadnilo práci s aplikací, při hledání a řešení chybových stavů.

7.3 Možnosti vylepšení

Jelikož aplikace nebyla zpracována pouze pro účely diplomové práce, ale její používání bude dlouhodobější, bude nejspíše v budoucnu nutné provést změny, které posunou její funkcionalitu dále.

Aplikace byla vyvíjena pro platformu Microsoft Windows a pro použití s ES Pohoda. Existuje velké množství firem, minimálně v okruhu firmy Assec Computers, které využívají pro chod firmy totožné prostředky. Proto má tato aplikace velký potenciál pro rozšíření i mimo firmu Assec Computers. Překážkou pro případnou expanzi je pak omezení aplikace pouze na jednoho konkrétního dodavatele. Pokud by se měla aplikace v budoucnu v tomuto směru dále rozvíjet, bylo by potřeba zapracovat na automatizovaném získávání dokladů i od jiných dodavatelů, případně na alternativním způsobu zadávání dokladů do aplikace.

V budoucnu by bylo vhodné do procesu tvorby nových skladových karet implementovat nějakou formu umělé inteligence, která by dokázala alespoň odhadnout údaje, které musí nyní vkládat uživatel ručně, na základě importovaných dat a předchozích voleb uživatele. Toto by uživateli ještě více usnadnilo práci s aplikací.

Zajímavou myšlenkou je rovněž uchovávání informací o zpracovaných datech. Z nasbíraných informací by pak mohla aplikace vytvářet různé přehledy, které by uživateli pomohly v rozhodování. Jelikož by šlo o velký objem dat, bylo by nutné aplikaci připojit k nějaké databázi.

8 Závěr

Cílem diplomové práce bylo navrhnout a implementovat aplikaci, jež bude ve firmě Assec Computers automatizovat import příjemek do ekonomického systému Pohoda. Automatizace měla pomoci ke snížení časové náročností a chybovosti při importu.

V teoretické části práce jsem nejprve představil ekonomický systém Pohoda a firmu, pro kterou bylo řešení zpracováváno. Rovněž jsem se věnoval automatizaci obecně, její historii, přínosům pro společnost a trendům do budoucna. Podstatnou částí byla také analýza současného stavu problematiky, ať už ve firmě, tak mimo ni.

V praktické části jsem nejprve analyzoval zákazníkovi požadavky na nové řešení. Následně jsem navrhnul a implementoval aplikaci, která v současné době ve firmě plní svoji funkci. Cíl, který byl stanoven na začátku práce, jsem tedy splnil. I přesto, že není plně automatizována (při vytváření skaldových karet je nutná asistence uživatele), urychluje aplikace zpracování příjemek ve firmě. O snížení chybovosti je stále ještě předčasné mluvit, jelikož aplikace neběží dostatečně dlouhou dobu.

9 Literatura

ARLOW, JIM A ILA 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.

ASSEC Computers [online]. 2012 [cit. 2016-09-29]. Dostupné z: http://www.assec.cz/

Definition of Rapid Application Development. The Economic Times [online]. 2016 [cit. 2016-12-08]. Dostupné z: http://economictimes.indiatimes.com/definition/rapid-application-development

DORFMAN, M. A RICHARD H. THAYER. *Software engineering.* Los Alamitos, Calif.: IEEE Computer Society Press, c1997. ISBN 0818676094.

FlexiInvoice [online]. 2016 [cit. 2016-10-25]. Dostupné z: http://flexiinvoice.cz/

GOTTESDIENER, ELLEN. *Rad Realities: Beyond The Hype To How Rad Really Works [online].* [cit. 2016-12-08]. Dostupné z: https://www.ebgconsulting.com/Pubs/Articles/RAD_Realities_Beyond_the_Hype_Gottesdiener.pdf

IEEE Standard Glossary of Software Engineering Terminology. IEEE Xplore: Digital Library [online]. 2016 [cit. 2016-10-27]. Dostupné z: http://ieeexplore.ieee.org/document/159342/?reload=true

Infomatic [online]. 2016 [cit. 2016-10-25]. Dostupné z: http://www.infomatic.cz/

LACKO, BRANISLAV. *Automatizace a automatizační technika.* Praha: Computer Press, 2000. Všechny cesty k informacím. ISBN 80-7226-246-7.

LEFFINGWELL, DEAN. A DON. WIDRIG. *Managing software requirements: a unified approach.* Reading, MA: Addison-Wesley, c2000. ISBN 0201615932.

NOF, SHIMON Y. *Springer handbook of automation.* New York: Springer, c2009. ISBN 3540788301.

PlusSystem [online]. 2016 [cit. 2016-10-25]. Dostupné z: http://plussystem.eu/aplikace/plusrouter

Qinve [online]. 2016 [cit. 2016-10-25]. Dostupné z: http://www.qinve.com/cs/

Rapid Application Development. Medix [online]. 2016 [cit. 2016-12-08]. Dostupné z: https://www.mendix.com/rapid-application-development/

SAWYER, PETE A GERALD KOTONYA. *Software Requirements.* Computing Department, Lancaster University, UK, 2001.

SDLC - RAD Model. Tutorials Point [online]. 2016 [cit. 2016-12-08]. Dostupné z: https://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm

Stormware [online]. 2016 [cit. 2016-10-20]. Dostupné z: http://www.stormware.cz/

Systems Engineering Fundamentals. Defense Acquisition University Press, 2001.

Rapid Application Development Model. WaveMaker [online]. 2016 [cit. 2016-12-28]. Dostupné z: http://www.wavemaker.com/rapid-application-development-model/

WIEGERS, KARL EUGENE. *Požadavky na software.* Vyd. 1. Brno: Computer Press, c2008. ISBN 978-80-251-1877-1.

XML komunikace. Stormware [online]. 2016 [cit. 2016-12-04]. Dostupné z: http://www.stormware.cz/xml/

10Seznam obrázků

11Seznam tabulek

Přílohy

A Generování souboru pro import příjemky

```
'Generuje XML Request soubory pro import příjemek
Public Sub generateRequestImportReceipt(prijemka As Prijemka, atCompId As
 String, centre As String)
 'cesta pro uložení souboru
 Dim pathRequestImportReceipt As String = Application.StartupPath() +
  "\AIPWorkingDir\Request\requestImportReceipt.xml"

 'vytvoření XML a jeho nastavení a úvodní tag dokumentu
 Dim wImportPrijemky As New
  Xml.XmlTextWriter(pathRequestImportReceipt, Nothing)
 wImportPrijemky.Formatting = Xml.Formatting.Indented
 wImportPrijemky.WriteStartDocument()

 'hlavička souboru
 wImportPrijemky.WriteStartElement("dat", "dataPack",
  "http://www.stormware.cz/schema/version_2/data.xsd")
 wImportPrijemky.WriteAttributeString("xmlns", "pri", Nothing,
  "http://www.stormware.cz/schema/version_2/prijemka.xsd")
 wImportPrijemky.WriteAttributeString("xmlns", "typ", Nothing,
  "http://www.stormware.cz/schema/version_2/type.xsd")
 wImportPrijemky.WriteAttributeString("id", "ID")
 wImportPrijemky.WriteAttributeString("ico", My.Settings.JednotkaIco)
 wImportPrijemky.WriteAttributeString("application",
  My.Settings.JednotkaStwph)
 wImportPrijemky.WriteAttributeString("version", "2.0")
 wImportPrijemky.WriteAttributeString("note", "Request - Receipt im
  port")

 wImportPrijemky.WriteStartElement("dat", "dataPackItem",
  "http://www.stormware.cz/schema/version_2/data.xsd")
 wImportPrijemky.WriteAttributeString("id", "ID")
 wImportPrijemky.WriteAttributeString("version", "2.0")

 wImportPrijemky.WriteStartElement("pri", "prijemka",
  "http://www.stormware.cz/schema/version_2/prijemka.xsd")
 wImportPrijemky.WriteAttributeString("version", "2.0")

 'datum příjemky, vyplněno aktuální datum při zpracování
 wImportPrijemky.WriteStartElement("pri", "prijemkaHeader",
  "http://www.stormware.cz/schema/version_2/prijemka.xsd")
 wImportPrijemky.WriteStartElement("pri", "date",
  "http://www.stormware.cz/schema/version_2/prijemka.xsd")
 wImportPrijemky.WriteValue(prijemka.Datum)
 wImportPrijemky.WriteEndElement()

 'popis příjemky, bude ukládat číslo faktury z AT Comp
 wImportPrijemky.WriteStartElement("pri", "text",
  "http://www.stormware.cz/schema/version_2/prijemka.xsd")
 wImportPrijemky.WriteValue(prijemka.Text)
 wImportPrijemky.WriteEndElement()

 wImportPrijemky.WriteStartElement("pri", "partnerIdentity",
  "http://www.stormware.cz/schema/version_2/prijemka.xsd")
 wImportPrijemky.WriteStartElement("typ", "id",
  "http://www.stormware.cz/schema/version_2/type.xsd")
 wImportPrijemky.WriteValue(atCompId)
 wImportPrijemky.WriteEndElement()
 wImportPrijemky.WriteEndElement()
```

```
wImportPrijemky.WriteStartElement("pri", "centre",
 "http://www.stormware.cz/schema/version_2/prijemka.xsd")
wImportPrijemky.WriteStartElement("typ", "ids",
 "http://www.stormware.cz/schema/version_2/type.xsd")
wImportPrijemky.WriteValue(centre)
wImportPrijemky.WriteEndElement()
wImportPrijemky.WriteEndElement()
wImportPrijemky.WriteEndElement()

wImportPrijemky.WriteStartElement("pri", "prijemkaDetail",
 "http://www.stormware.cz/schema/version_2/prijemka.xsd")
'každou položku v prom. seznamPoložekSEanem
For Each polozka As PolozkaPrijemky In prijemka.Polozky
 If polozka.InternalType = "2" Then
  wImportPrijemky.WriteStartElement("pri", "prijemkaItem",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")

  'množství
  wImportPrijemky.WriteStartElement("pri", "quantity",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")
  wImportPrijemky.WriteValue(polozka.Quantity)
  wImportPrijemky.WriteEndElement()

  'měrná jednotka
  wImportPrijemky.WriteStartElement("pri", "unit",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")
  wImportPrijemky.WriteValue(polozka.Unit)
  wImportPrijemky.WriteEndElement()

  'sazba DPH, http://www.stormware.cz/xml/proVyvojare/sazbaDPH.aspx
  wImportPrijemky.WriteStartElement("pri", "rateVAT",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")
  If polozka.RateVAT = "0" Then
   wImportPrijemky.WriteValue("none")
  ElseIf polozka.RateVAT = "15" Then
   wImportPrijemky.WriteValue("low")
  Else
   wImportPrijemky.WriteValue("high")
  End If
  wImportPrijemky.WriteEndElement()

  'vazba na skladovou kartu, hledá podle pole EAN v zásobách
  wImportPrijemky.WriteStartElement("pri", "stockItem",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")
  wImportPrijemky.WriteStartElement("typ", "stockItem",
   "http://www.stormware.cz/schema/version_2/type.xsd")
  wImportPrijemky.WriteStartElement("typ", "EAN",
   "http://www.stormware.cz/schema/version_2/type.xsd")
 wImportPrijemky.WriteValue(polozka.EAN)
 wImportPrijemky.WriteEndElement()
 wImportPrijemky.WriteEndElement()
 wImportPrijemky.WriteEndElement()

 'jednotková cena položky
 wImportPrijemky.WriteStartElement("pri", "homeCurrency",
  "http://www.stormware.cz/schema/version_2/prijemka.xsd")
 wImportPrijemky.WriteStartElement("typ", "price",
  "http://www.stormware.cz/schema/version_2/type.xsd")
 wImportPrijemky.WriteValue(polozka.UnitPrice)
 wImportPrijemky.WriteEndElement()
```

```
  wImportPrijemky.WriteEndElement()

  wImportPrijemky.WriteEndElement()
 ElseIf polozka.InternalType = "3" Then
  wImportPrijemky.WriteStartElement("pri", "prijemkaAccessoryChargesI
   tem", "http://www.stormware.cz/schema/version_2/prijemka.xsd")

  'množství
  wImportPrijemky.WriteStartElement("pri", "quantity",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")
  wImportPrijemky.WriteValue(polozka.Quantity)
  wImportPrijemky.WriteEndElement()

  'sazba DPH, http://www.stormware.cz/xml/proVyvojare/sazbaDPH.aspx
  wImportPrijemky.WriteStartElement("pri", "rateVAT",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")
  If polozka.RateVAT = "0%" Then
   wImportPrijemky.WriteValue("none")
  ElseIf polozka.RateVAT = "15%" Then
   wImportPrijemky.WriteValue("low")
  Else
   wImportPrijemky.WriteValue("high")
  End If
  wImportPrijemky.WriteEndElement()

  'jednotková cena položky
  wImportPrijemky.WriteStartElement("pri", "homeCurrency",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")
  wImportPrijemky.WriteStartElement("typ", "unitPrice",
   "http://www.stormware.cz/schema/version_2/type.xsd")
  wImportPrijemky.WriteValue(polozka.UnitPrice)
  wImportPrijemky.WriteEndElement()
  wImportPrijemky.WriteEndElement()

  'note
  wImportPrijemky.WriteStartElement("pri", "note",
   "http://www.stormware.cz/schema/version_2/prijemka.xsd")
  wImportPrijemky.WriteValue(polozka.Name)
  wImportPrijemky.WriteEndElement()

  wImportPrijemky.WriteEndElement()
 End If
Next
 wImportPrijemky.WriteEndElement()

 'patička příjemky
 wImportPrijemky.WriteStartElement("pri", "prijemkaSummary",
  "http://www.stormware.cz/schema/version_2/prijemka.xsd")
 wImportPrijemky.WriteEndElement()
 wImportPrijemky.WriteEndElement()
 wImportPrijemky.WriteEndElement()
 wImportPrijemky.WriteEndElement()
 wImportPrijemky.WriteEndDocument()
 wImportPrijemky.Close()
End Sub
```

B Generování XML požadavku pro import skladových karet

```
Public Sub generateRequestImportStockItems(prijemka As Prijemka)
 Dim stockItemImportNo As Integer = 1
 Dim pathRequestImportStockItems As String = Application.StartupPath() +
  "\AIPWorkingDir\Request\requestImportStockItems.xml"

 'nastavení XML souboru a začáteční tag
 Dim wImportStockItems As New Xml.XmlTextWriter(pathRequestImportStockItems,
  Nothing)
 wImportStockItems.Formatting = Xml.Formatting.Indented
 wImportStockItems.WriteStartDocument()

 'hlavička XML souboru
 wImportStockItems.WriteStartElement("dat", "dataPack",
  "http://www.stormware.cz/schema/version_2/data.xsd")
 wImportStockItems.WriteAttributeString("xmlns", "stk", Nothing,
  "http://www.stormware.cz/schema/version_2/stock.xsd")
 wImportStockItems.WriteAttributeString("xmlns", "typ", Nothing,
  "http://www.stormware.cz/schema/version_2/type.xsd")
 wImportStockItems.WriteAttributeString("id", "ID")
 wImportStockItems.WriteAttributeString("ico", My.Settings.JednotkaIco)
 wImportStockItems.WriteAttributeString("application",
  My.Settings.JednotkaStwph)
 wImportStockItems.WriteAttributeString("ppp", "2.0")
 wImportStockItems.WriteAttributeString("note", "Import skladových zásob")

 'projde všechny položky v parametru prijemka, nastavi jmeno položky v XML
  souboru
 For Each polozka As PolozkaPrijemky In prijemka.Polozky
  Dim pp As String = "ZAS" + stockItemImportNo.ToString

  'budou se vytvářet skladové karty pouze pro zásoby, které ji dosud nemají
  If polozka.InternalCardExists = False Then
   If polozka.InternalType = "2" Then
    'ID položky v XML souboru
    wImportStockItems.WriteStartElement("dat", "dataPackItem",
     "http://www.stormware.cz/schema/version_2/data.xsd")
    wImportStockItems.WriteAttributeString("id", stockItemImportName)
    wImportStockItems.WriteAttributeString("version", "2.0")
    stockItemImportNo += 1

    wImportStockItems.WriteStartElement("stk", "stock",
     "http://www.stormware.cz/schema/version_2/stock.xsd")
    wImportStockItems.WriteAttributeString("version", "2.0")
    wImportStockItems.WriteStartElement("stk", "stockHeader",
     "http://www.stormware.cz/schema/version_2/stock.xsd")

    'typ zasoby
    wImportStockItems.WriteStartElement("stk", "stockType",
     "http://www.stormware.cz/schema/version_2/stock.xsd")
    wImportStockItems.WriteValue("card")
    wImportStockItems.WriteEndElement()

    'kod zasoby
    wImportStockItems.WriteStartElement("stk", "code",
     "http://www.stormware.cz/schema/version_2/stock.xsd")
```

```
wImportStockItems.WriteValue(polozka.Code)
wImportStockItems.WriteEndElement()

'EAN
wImportStockItems.WriteStartElement("stk", "EAN",
 "http://www.stormware.cz/schema/version_2/stock.xsd")
wImportStockItems.WriteValue(polozka.EAN)
wImportStockItems.WriteEndElement()

'nákupní DPH
wImportStockItems.WriteStartElement("stk", "purchasingRateVAT",
 "http://www.stormware.cz/schema/version_2/stock.xsd")
If polozka.RateVAT = "0%" Then
 wImportStockItems.WriteValue("none")
ElseIf polozka.RateVAT = "15%" Then
 wImportStockItems.WriteValue("low")
Else
 wImportStockItems.WriteValue("high")
End If
wImportStockItems.WriteEndElement()

'prodejní DPH
wImportStockItems.WriteStartElement("stk", "sellingRateVAT",
 "http://www.stormware.cz/schema/version_2/stock.xsd")
If polozka.RateVAT = "0%" Then
 wImportStockItems.WriteValue("none")
ElseIf polozka.RateVAT = "15%" Then
 wImportStockItems.WriteValue("low")
Else
 wImportStockItems.WriteValue("high")
End If
wImportStockItems.WriteEndElement()

'isSales
wImportStockItems.WriteStartElement("stk", "isSales",
 "http://www.stormware.cz/schema/version_2/stock.xsd")
wImportStockItems.WriteValue("true")
wImportStockItems.WriteEndElement()

'isInternet
wImportStockItems.WriteStartElement("stk", "isInternet",
 "http://www.stormware.cz/schema/version_2/stock.xsd")
wImportStockItems.WriteValue("true")
wImportStockItems.WriteEndElement()

'jméno
wImportStockItems.WriteStartElement("stk", "name",
 "http://www.stormware.cz/schema/version_2/stock.xsd")
wImportStockItems.WriteValue(polozka.Name)
wImportStockItems.WriteEndElement()

'jednotka
wImportStockItems.WriteStartElement("stk", "unit",
 "http://www.stormware.cz/schema/version_2/stock.xsd")
If polozka.Unit = "balení" Then
 wImportStockItems.WriteValue("bal")
ElseIf polozka.Unit = "centimetry" Then
 wImportStockItems.WriteValue("cm")
ElseIf polozka.Unit = "gramy" Then
 wImportStockItems.WriteValue("g")
ElseIf polozka.Unit = "hodiny" Then
```

```
     wImportStockItems.WriteValue("hod")
    ElseIf polozka.Unit = "kilogramy" Then
     wImportStockItems.WriteValue("kg")
    ElseIf polozka.Unit = "kilometry" Then
     wImportStockItems.WriteValue("km")
    ElseIf polozka.Unit = "kusy" Then
     wImportStockItems.WriteValue("ks")
    ElseIf polozka.Unit = "litry" Then
     wImportStockItems.WriteValue("l")
    ElseIf polozka.Unit = "metry" Then
     wImportStockItems.WriteValue("m")
    ElseIf polozka.Unit = "minuty" Then
     wImportStockItems.WriteValue("min")
    ElseIf polozka.Unit = "palety" Then
     wImportStockItems.WriteValue("pal")
    ElseIf polozka.Unit = "páry" Then
     wImportStockItems.WriteValue("PAR")
    ElseIf polozka.Unit = "metrické centy" Then
     wImportStockItems.WriteValue("q")
    ElseIf polozka.Unit = "sady" Then
     wImportStockItems.WriteValue("SAD")
    ElseIf polozka.Unit = "tuny" Then
     wImportStockItems.WriteValue("t")
    End If
    wImportStockItems.WriteEndElement()

    'sklad
    wImportStockItems.WriteStartElement("stk", "storage",
     "http://www.stormware.cz/schema/version_2/stock.xsd")
    wImportStockItems.WriteStartElement("typ", "ids",
     "http://www.stormware.cz/schema/version_2/type.xsd")
    wImportStockItems.WriteValue(polozka.Storage)
    wImportStockItems.WriteEndElement()
    wImportStockItems.WriteEndElement()

    'typ ceny
    wImportStockItems.WriteStartElement("stk", "typePrice",
     "http://www.stormware.cz/schema/version_2/stock.xsd")
    wImportStockItems.WriteStartElement("typ", "ids",
     "http://www.stormware.cz/schema/version_2/type.xsd")
    wImportStockItems.WriteValue(polozka.PriceType)
    wImportStockItems.WriteEndElement()
    wImportStockItems.WriteEndElement()

    'nákupní cena
    wImportStockItems.WriteStartElement("stk", "purchasingPrice",
     "http://www.stormware.cz/schema/version_2/stock.xsd")
    wImportStockItems.WriteValue(polozka.UnitPrice)
    wImportStockItems.WriteEndElement()
    wImportStockItems.WriteEndElement()
    wImportStockItems.WriteEndElement()
    wImportStockItems.WriteEndElement()
   End If
  End If
 Next

 'zakončení XML souboru
 wImportStockItems.WriteEndElement()
 wImportStockItems.WriteEndDocument()
 wImportStockItems.Close()
End Sub
```