FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Rozšíření skladového systému Atlantis
Student:	Jan Cvrček
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je rozšířit skladový systém, pracovně pojmenovaný Atlantis, původně vyvinutý v rámci diplomových prací Ing. Oldřicha Malce a Ing. Pavla Kováře.

Postupujte v těchto krocích:
Analyzujte současný stav skladového systému Atlantis.
Na základě analýzy navrhněte nejpodstatnější rozšíření - zaměřte se hlavně na proces objednávek z externích systémů. Návrh realizujte jak softwarový, tak uživatelský.
Návrh projděte a řádně konzultujte se svým vedoucím.
Na základě návrhu implementujte výsledné rozšíření.
Rozšíření řádně otestujte, minimálně na jednom reálném uživateli skladu.
Shrňte dosažené výsledky, navrhněte možná budoucí vylepšení dle získaných zkušeností.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 19. února 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Rozšíření skladového systému Atlantis

Jan Cvrček

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

13. května 2021

Poděkování

V první řadě bych chtěl poděkovat vedoucímu mé práce, panu Ing. Jiřímu Hunkovi, za příležitost pracovat na projektu Atlantis, ale hlavně za jeho podporu, odezvu a komunikaci v průběhu práce. Stejně tak bych chtěl moc poděkovat Ing. Oldřichu Malcovi, který mi vždy při vývoji poskytl potřebné informace a pomoc, když jsem zrovna potřeboval s čímkoliv poradit. Obdobně bych chtěl ocenit veškerou podporu a pomoc, které se mi dostalo od Ing. Marka Erbena, který byl mým vedoucím v obou předmětech BI-SP, díky čemuž jsem měl možnost se v předstihu seznámit s technologiemi, které jsem později na tomto projektu uplatňoval. V neposlední řadě patří velký dík všem členům týmu, kteří vyvíjí skladový systém Atlantis, od nichž se mi dostalo velké ochoty vždy, když jsem potřeboval.

Na závěr bych chtěl poděkovat mé rodině, která mě podporovala po celou dobu mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 13. května 2021 . . . . . .

Odkaz na tuto práci

Cvrček, Jan. *Rozšíření skladového systému Atlantis*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato bakalářská práce je zaměřena na analýzu, návrh a vylepšení již existujícího skladového systému s pracovním názvem Atlantis, který byl v minulosti vyvinout Ing. Oldřichem Malcem a Ing. Pavlem Kovářem v rámci jejich diplomových prací. Systém je v současné době již plně funkční a nasazen v několika menších skladech. V této práci se věnuji hlavně optimalizaci a usnadnění existujících procesů a navrhování, posléze i vytváření, nových, zákazníky žádaných rozšíření, které ulehčují práci jak pracovníkům v stávajících menších skladech, tak novým potenciálním uživatelům s většími sklady. Tyto návrhy jsou většinou orientované na koncepty spojené objednávkami, které pocházejí ze systémů třetích stran. Výsledkem práce je celá řada podnětů a návrhů pro vylepšení stávající podoby systému, ale v první řadě realizace rozšíření, které značně automatizuje a zjednodušuje proces zpracovávání objednávek.

Klíčová slova sklad, skladový systém, rozšíření, Vue.js, PHP, Symfony, zpracování objednávek

Abstract

This bachelor thesis is focused on the analysis, design and improvement of the existing warehouse system with the working name Atlantis, which was developed in the past by Ing. Oldřich Malec and Ing. Pavel Kovář as part of their diploma work. The system is now fully functional and deployed in several smaller warehouses. In this thesis I am mainly focused on optimization and facilitation of existing processes and designing, and then creating, new, customer-requested extensions that facilitate the work of both workers in existing smaller warehouses and new potential users with larger warehouses. These designs are mostly oriented towards concepts linked to orders originating from third-party systems. The work has resulted in a number of suggestions and proposals for improving the current system design, but primarily the implementation of an extension that significantly automates and simplifies the order processing process.

Keywords warehouse, warehouse system, extension, Vue.js, PHP, Symfony, order processing

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Při vývoji libovolného software je vždy velice náročné docílit takového výsledku, který by maximálně vyhovoval a ulehčoval práci co nejširšímu spektru zákazníků, jež budou daný software denně využívat. K úspěšnému výsledku vede často velmi dlouhá cesta složená z iterativních analýz, návrhů a implementace nových součástí, u kterých na samém počátku projektu ještě nemuselo být vůbec jasné, že budou jednoho dne potřebné. Tento projekt není v tomto aspektu nijak odlišný a já se zapojím do jeho dalšího vylepšování a rozšiřování na základně analýzou zjištěných nedostatků, které budou v tuto chvíli pro současné i potenciální uživatele tohoto systému nejvíce prioritní.

Moje motivace pro vybrání si právě tohoto projektu, který se zaměřuje na úpravu již existujícího systému, pramení z dosavadní kladné zkušenosti s vývojem drobných webových aplikací na míru různým zákazníkům pohybujících se v rozličných oblastech.

Hlavním cílem této práce je navrhnutí a případná realizace takových změn a rozšíření, které by současný skladový systém upravily tak, aby jeho funkcionality vyhovovaly a ulehčovaly maximum práce co nejvíce zákazníkům, nehledě na to, že jejich sklady jsou odlišné (velikostí i typem fungovaní). Dle zadání se budu snažit dosáhnout co největšího množství reálných výsledků zejména v oblasti procesů spojenými se zpracováváním objednávek.

V textu se nejprve zaměřím na současný stav aplikace, kde za pomoci uživatelů a zadavatele identifikuji nedostatky, které jsou v tuto chvíli a výhledově v blízké budoucnosti nejpalčivější. Na základě poznatků z této analýzy pak vytvořím návrh pro úpravy a rozšíření aplikace, které by měly dané nedostatky odstranit. Po konzultaci a schválení všech návrhů ze strany zadavatele se pustím do realizace nejvíce prioritních úprav. V poslední fázi pak mnou implementované rozšíření a úpravy v aplikaci otestuji, a to jak programově, tak uživatelsky za pomoci nějakého uživatele, který zná danou problematiku a skladovou aplikaci rutinně používá. V poslední části shrnu výsledky, kterých bylo v rámci této práce dosaženo, a navrhnu případné další vylepšení a

postupy, kterými by mohlo být vhodné se v návaznosti na tuto práci zabývat.

Tato práce navazuje na předchozí diplomové práce Ing. Oldřicha Malce a Ing. Pavla Kováře, v rámci kterých byl celý skladový systém Atlantis původně vyvinut. V práci se vyskytuje rovněž návaznost na bakalářskou práci Bc. Denisa Talára, který již částečně analyzoval a vypracoval řadu návrhů pro některé z problematik, kterými se zde budu zabývat.

Kapitola 1

Analýza

V této kapitole se zaměřím na analýzu současného stavu systému. Nejprve popíši základní koncepty a procesy, které se odehrávají v rámci rutinního používání aplikace a na základě takto získaných poznatků identifikuji potenciální nedostatky, pro které by se mohlo vyplatit upravit stávající, nebo dokonce navrhnout nějaké úplně nové řešení. Dle zadání se zaměřím mimo jiné zejména na to, jakým způsobem by bylo možné v systému zpracovávat poptávky přicházející ze systémů třetích stran (jako jsou e-shopy a podobně).

Při analýze současného stavu budu ve velkém množství čerpat informace ze stavu samotného systému v době psaní tohoto textu. Dále pro účely popsání některých procesů (co se při jaké operaci z hlediska uživatele děje v aplikaci a co se u toho fyzicky odehrává ve skladu) budu čerpat informace i z diplomových prací Ing. Oldřicha Malce a Ing. Pavla kováře, byť se aplikace v současné podobě v některých částech, z hlediska uživatelského rozhraní, trochu liší od tehdejší původní podoby. [1], [2]

Během identifikace jednotlivých dílčích nedostatků budu brát v potaz i některé již objevené podněty z práce Bc. Denise Talára, který se v ní zaměřoval na optimalizaci nejčastějších procesů v tomto systému, kde se věnoval zejména skladům napojeným na menší e-shopy. [3] Při analýze tedy přihlédnu zprvu i k jím identifikovaným problémům a zhodnotím, zda jsou pro současné spektrum zákazníků stále prioritní, případně pak dle toho jejich formulace patřičně upravím.

1.1 Současný stav systému

1.1.1 Proces vyskladnění produktů

Při analýze systému jsem se nejprve zaměřil na proces úzce související s požadavkem na přidání poptávek do systému, na který se mám prioritně v této práci zaměřit – jedná se o proces vyskladnění zboží.

1.1.1.1 Vyskladňění jako typ uživatelské úlohy

V aplikaci jsou veškeré větší úkony, které jsou stěžejní pro fungování skladu, realizované ve formě takzvaných úloh. Úloha je nějaký úkon, který může splnit v aplikaci pouze příslušná uživatelská role (vedoucí nebo skladník). Může být přiřazena v jednu chvíli pouze jednomu uživateli (nikdo jiný na ní tudíž nemůže v ten samý okamžik pracovat). Po dokončení je většina úloh schvalována uživatelem s rolí vedoucí – úloha je buď označena jako uzavřená, nebo je vrácena zpět k přepracování uživateli, který ji odevzdal, případně vrácena zpět bez přiřazení na konkrétního uživatele.

Proces vyskladnění je typ právě jedné takovéto úlohy, kde figuruje jako vykonávající aktér role skladník. Role vedoucí se u tohoto typu úloh stará o vytvoření úlohy a její finální schválení.

1.1.1.2 Průběh úlohy vyskladnění

Jako první krok je v tomto procesu vždy úvodní vytvoření úlohy vyskladnění vedoucím. Ten při vytváření úlohy pokaždé ve formuláři specifikuje skladové položky, které se mají vyskladnit a jejich množství. Další povinné informace, které musí před vznikem úlohy poskytnout jsou:

- Sklad a podsklad, kterého se úloha týká
- Priorita vytvářené úlohy

Pokud je navíc v konfiguraci skladu nastaveno, že veškeré vyskladnění jsou typu doručení (courier), tak musí vedoucí navíc specifikovat ještě umístění, na které má skladník zboží vyskladnit (místo, ze kterého bude zboží předáno dopravci).

V dalším kroku si skladník na sebe přiřadí tuto volnou úlohu a začne na ní pracovat. V závislosti na typu úlohy vyskladnění ji příslušně zpracuje. V případě, že se jedná o osobní převzetí, skladník vyskladňuje veškeré předměty pouze k sobě do inventáře. Po přesunutí všech předmětů a dokončení je úloha rovnou systémem schválena a není vyžadována kontrola od vedoucího. Jedná-li se v opačném případě o typ úlohy doručení, skladník přesouvá veškeré položky zprvu do svého inventáře, stejně jako v prvním případě, dále však musí všechny předměty položit na cílové umístění, které bylo již při vytváření úlohy určeno vedoucím. Po odevzdání úlohy může vedoucí vykonanou práci zkontrolovat a dle tohoto úlohu schválit, nebo zamítnout a vrátit k přepracování.

1.1.2 Využívání vyskladnění pro simulaci poptávek

Z důvodu absence konceptu poptávek jako takových se v současné době v některých skladech využívají úlohy vyskladnění pro simulaci poptávek. Tento způsob je nyní využíván zejména u menších internetových obchodů.

Vzhledem k tomu, že je aplikace realizována jako dvě striktně oddělené části (backendová a frontendová část), není žádný problém zavolat jakýkoliv backendový endpoint z jiného místa, než je jen samotná frontendová část aplikace. Právě tohoto je využíváno u vytváření úloh vyskladnění. Typickým scénářem je například, že zákazník dokončil svoji objednávku v systému internetového obchodu, který vezme údaje zákazníka a na skladovém systému zavolá endpoint pro jeho vytvoření – v našem systému se na základě těchto údajů založí nový odběratel. Identifikátor tohoto odběratele je pak použit při dalším požadavku na náš systém – vytvoření úlohy vyskladnění, adresované v předchozím kroku vytvořenému odběrateli s identifikátory a množstvím příslušných produktů, které byly součástí zákazníkovy objednávky.

1.1.2.1 Odpovědnost za nedostupné produkty

Od této chvíle se však v tomto procesu začne projevovat skutečnost, že pro takovéto zacházení nebyl původně vůbec navrhnut. Například, pokud nejsou všechny produkty dostupné, je tato situace řešená tím způsobem, že se k úloze vyskladnění při vytváření přidá poznámka, která značí, že je objednávka nekompletní a skladník by ji neměl začít zpracovávat. Poté se systém internetového obchodu pokouší cyklicky úlohu vyskladnění aktualizovat o chybějící předměty (je na něm, aby si držel informace o tom, co v jím založeném vyskladnění chybí).

1.1.2.2 Ztráta informace o původu produktů a konvence priorit

Další ne zcela optimální aspekt spočívá v tom, že není žádoucí, aby se každá objednávka ve skladu uspokojovala zvlášť v rámci jedné úlohy vyskladnění. Z tohoto důvodu je v systému realizováno provizorní slučování těchto úloh. Typicky to bývá tak, že je vyskladnění, které vzniklo výše popsaným postupem z objednávky, nejprve označeno jako nízkoprioritní. Skladníci jsou uvědoměni, že na úlohách s takovouto prioritou nemají pracovat. Pokud dojde na straně obchodu k vystavení faktury pro objednávku, priorita příslušné úlohy je změněna na standardní a její odběratel zaměněn za generického odběratele. Protože systém v současném stavu úlohy slučuje dle stejného odběratele, jsou pak tyto úlohy s generickým odběratelem spojeny do jedné úlohy vyskladnění, která obsahuje produkty všech úloh, ze kterých vznikla. Při slučování úloh jsou vždy navíc spojeny jejich popisky, takže po ztracení informace o odběrateli je dostupná alespoň informace o tom, z kterých objednávek konečné vyskladnění vzniklo (dílčí vyskladnění mají vždy v popisku číslo objednávky). Nicméně není zde možné v rámci skladového systému dohledat, do které objednávky jaký z produktů přímo patří.

1.1.2.3 Souhrn analýzy tohoto konceptu

Je tedy zřejmé, že je proces vyskladnění pro výše uvedené úkony ne zcela optimální. První z problémů je, že se v něm spoléhá na konvence mezi skladníky a může omylem dojít k narušení celého procesu – například k vyskladnění nekompletní nebo nezaplacené objednávky. Další z nevyhovujících skutečností je fakt, že se v tomto procesu ztrácí řada důležitých informací, jako například, komu jsou které produkty adresovány (pokud personál skladu nemá zároveň přístup i do systému, skrz který byly objednávky původně vytvořeny). Jako jeden z posledních nevyhovujících aspektů bych hodnotil to, že odpovědnost za nekompletní a neuzavřená vyskladnění v tuto chvíli nese systém třetí strany, jenž by podle mého názoru neměl být jediným subjektem, který má informace o stavu a kompletnosti či nekompletnosti objednávky, potažmo vyskladnění. Bylo by zde vhodné trochu více abstrahovat tento proces pro třetí strany (jako je již zmiňovaný internetový obchod), aby měl samotný skladový systém přehled o tom, co za produkty (i za předpokladu, že nejsou momentálně dostupné) konkrétní objednávka obsahuje a po jejich pozdějším naskladnění systém tyto produkty automaticky přiřadil do příslušných objednávek, které je již poptávají, bez čekání na další příkaz od třetí strany.

1.1.3 Externí objednávky

Mimo využití úloh vyskladnění pro simulaci poptávek, je nyní v současném systému také možné použití externích objednávek, které v současné chvíli využívá jen jeden systém třetí strany – systém Hamster. Vzhledem k tomu, že externí objednávka má některé funkcionality, které by později mohly být využity při návrhu a realizaci konceptu poptávek, nebo by dokonce mohly být nějakým způsobem přepracovány tak, aby námi kladené požadavky splňovaly a stal se z nich tak plnohodnotný nástroj pro zpracovávání poptávek, rozhodl jsem se tuto část systému rovněž zanalyzovat.

Externí objednávka je, stejně jako vyskladnění, typem úlohy. Jedná se o takový vzor pro budoucího vyskladnění, které z něj později vznikne – obsahuje drtivou většinu údajů z těch, které jsou potřeba pro vytvoření nového vyskladnění. Mezi hlavní rozdíly patří například to, že produkty zahrnuté v externí objednávce nemusí být všechny v tu dobu skladem – je zde trochu abstraktnější vazba na poptávaný produkt, místo vazby na konkrétního kus (na konkrétní šarži nebo sériové číslo), jako je tomu právě v úloze vyskladnění.

1.1.3.1 Nemožnost částečného uspokojení objednávky

Abstrakce produktů, které náleží do objednávky je na jednu stranu přínosná, vzhledem k tomu, že lze ze systému třetí strany založit objednávku i přes to, že některé produkty nejsou skladem, na druhou stranu zde mohou vzniknout nepříjemnosti spojené s částečným uspokojením, protože z objednávky, která poptává nedostupné produkty, nebude možné vytvořit žádné vyskladnění.

1.1.3.2 Zamezení nechtěné manipulaci s objednávkou

Další vlastností, která je však čistě pozitivní, je skutečnost, že není možné s objednávkou v tomto případě nijak nechtěně manipulovat, na rozdíl od objednávky ve formě již založeného vyskladnění, kde nízká priorita obvykle symbolizovala zákaz manipulace s vyskladněním – externí objednávka již nejde zaměnit za vyskladnění, které je určené ke zpracování, ba dokonce uživatelská role skladník nemá k tomuto typu úlohy vůbec přístup.

1.1.3.3 Absence optimálního rozdělení objednávek do vyskladnění

V neposlední řadě však nelze opomenout negativní skutečnost v podobě toho, že pro každou externí objednávku vzniká separátní úloha vyskladnění a tyto úlohy se nijak neslučují, protože jejich odběratel zůstává stejný, jako je uveden v objednávce – to znemožňuje vyskladnění produktů z více různých objednávek současně a skladník tak musí neoptimálně chodit po skladě a vyskladňovat produkty z každé objednávky zvlášť, i kdyby byly v objednávkách poptány produkty po jednom kusu. Na druhou stranou může být výhodou této stinné stránky alespoň skutečnost, že je zde u objednávek zachována dohledatelnost a není zde problém s dohledáním původu jednotlivých položek ve vyskladnění (kdo je jejich odběratelem).

1.1.3.4 Shrnutí aktuálního stavu externích poptávek

Externí objednávky tedy v současné podobě částečně již řeší některé nedostatky objevené u procesu simulace poptávek pomocí úloh vyskladnění. Řešení však stále není zcela optimální – narážíme zde na problémy s nemožným částečným uspokojováním objednávek a zároveň s jejich neoptimálním rozdělováním do separátních úloh vyskladnění. Jako základ by však, podle mého názoru, tento dosavadní koncept mohl postačit. Bude zde ovšem potřeba provézt řadu změn, aby se odstranily nebo minimalizovaly výše popsané neduhy.

1.1.4 Evidence expedic k odběrateli

V předchozí sekci, kde jsem se věnoval zmapování současného stavu externích objednávek, jsem již zmínil, že u objednávek adresovaným konkrétním odběratelům, je velice důležité mít možnost dohledat, které objednávky byly odeslány jakému odběrateli. Tato myšlenka mne posunula ještě trochu dále – vybereme-li si konkrétního odběratele evidovaného v současném systému, tak jakým způsobem je nyní vlastně možné zjistit, jaké veškeré produkty byly na jeho jméno v minulosti expedovány a odeslány? A mimo to, jak je možné jednoduše dohledat i celkové expedice objednávek s jejich časy a tím mít dále informace o tom, jaké produkty byly v v jeden čas zákazníkovi spolu odeslány?

Abych odpověděl na výše položené otázky – cesta ke zjištění informace, které produkty a v jakých dávkách byly k zákazníkovi expedovány, je v součas-

ném systému poměrně přímočará, ovšem značně náročná, protože pracovník skladu musí manuálně projít veškeré dokončené úlohy vyskladnění, v jejich detailu zjistit, zda se jednalo o expedici pro hledaného zákazníka a v kladném případě pak přejít do záložky v úloze, kde nalezne seznam vyskladněných produktů, včetně jejich kvantit. Správného výsledku se ovšem pracovník dobere jen pokud byly veškeré úlohy vyskladnění založeny na základě externích objednávek nebo byly manuálně vytvořené uživatelem. Pokud vznikaly tyto úlohy už od počátku jako vyskladnění s nízkou prioritou *(proces popsaný v kapitole 1.1.2)*, tak byla informace o původním odběrateli při slučování s dalšími vyskladněními ztracena, a tak není v silách pracovníka tyto expedice dohledat.

Optimalizace úkonů vedoucích k obdržení těchto informací je tedy žádoucí, protože pracovníkovi ušetří značné množství práce a hlavně zajistí získání kompletních údajů, z nichž nemohou být v současné podobě za určitých podmínek některé obdrženy. Pokud bychom pro všechny budoucí objednávky používali již existující úlohu externí objednávky, kde je vždy odběratel v detailu zachován, neměl by být problém veškeré jeho expedice zjistit (pomocí existujících vazeb v současné podobě databázového modelu).

1.1.5 Evidence odběratelem vráceného zboží

Po evidenci pohybu produktů směrem k zákazníkovi se přirozeně může objevit i myšlenka na zaznamenávání těch produktů, které byly zákazníkem po prvotním převzetí po nějaké době vráceny zpět (například u e-shopových skladů, kde mají zákazníci nárok zboží v omezené časové lhůtě vrátit [5]). Když se nad tímto konceptem zamyslíme, je v podstatě nezbytný pro relevantnost údajů o odběrech zákazníka, popsaných v předchozí kapitole – pokud budeme do zákazníkem odebraných produktů započítávat i ty kusy, které později vrátil, tak nebudeme dostávat pravdivé výsledky.

Při vrácení zboží od zákazníka by mohla být užitečná jedna ze základních, již existujících skladových úloh – naskladnění. Pokud bychom byli schopni tuto úlohu nějakým způsobem asociovat s konkrétním odběratelem, vrácené zboží by pak nemusel být žádný problém dohledat.

1.1.6 Rozdělování objednávek do vyskladnění

Optimální rozdělování objednávek do jednotlivých úloh vyskladnění je věcí, kterou žádná z dosavadních metod využívaných k vytváření objednávek neřeší. V současném stavu není v podstatě možné ani manuální seskupení objednávek do jedné úlohy vyskladnění. Počet objednávek v jednom vyskladnění je vždy určen způsobem, jakým byly objednávky vytvořeny – v případě simulace objednávek pomocí samotných úloh vyskladnění *(proces popsaný v kapitole 1.1.2)*, budou ve finálním vyskladnění veškeré objednávky založené od posledního dokončeného vyskladnění, v případě externích objednávek *(kapitola*

1.1.3) pak bude počet vyskladnění přímo úměrný počtu objednávek v poměru 1:1.

Analýzou a dílčími návrhy pro optimalizaci některých procesů spojených s vyskladňováním produktů ze skladů, které jsou napojeny zejména na menší e-shopové systémy, se ve své práci vydatně zabýval Bc. Denis Talár. [3] Ve své analýze došel k závěru, že pro sklady současných zákazníků je nejvíce vhodný způsob dávkového vyskladňování [6], při kterém se vícero objednávek seskupuje do jednoho společného vyskladnění. Při tomto seskupování je ovšem důležité rozdělit dílčí objednávky tak, aby byl pohyb po skladě a potřebná práce při balení produktů co nejvíce minimalizována. Je tedy výhodné do vyskladnění rozdělovat objednávky, které poptávají stejné produkty [7], nebo alespoň produkty na stejných/blízkých skladových umístěních. Pro sklady s menšími počty objednávek nebo/a nižší rozmanitostí skladovaných produktů, by tedy teoreticky mohlo být pro pracovníky dostačující mít možnost vytvořit si vyskladnění z manuálně vybraných objednávek. Sklady, kde je počet objednávek natolik vysoký, že není možné, aby je pracovníci stíhali rozdělovat do vyskladnění manuálně, případně pak také sklady s velkým množstvím rozdílných produktů, umístěných na různých umístěních, by pak mimo samotnou možnost manuálního seskupení objednávek vyžadovaly i některé automatizované procesy, které by podle předem definovaných strategií z objednávek sestavovaly jednotlivá vyskladnění co nejoptimálněji. Některým těmto strategiím řešícím problém, jak optimálně rozdělovat objednávky, se dále věnoval Denis v pozdějších částech své práce, kde některé možné strategie popsal – zaměřil se nejvíce na strategie orientující se na rozdělení objednávek dle dodavatele, který je bude k zákazníkovi převážet. [3]

Po delší konzultaci se zadavatelem práce [4], kde jsem se dozvěděl informace o tom, že je v blízké budoucnosti v plánu získat zákazníky, jejichž sklady budou značně rozsáhlejší a budou přijímat denně mnohem více objednávek, než sklady stávajících zákazníků, bude tedy podle mého názoru potřeba umožnit nejen základní seskupování objednávek (ať už v jakékoliv podobě) do úloh vyskladnění, ale navíc i nějaký automatizovaný přístup k tvorbě vyskladnění. Při pozdějším návrhu s vedoucím zhodnotíme, zda použít některé již navržené postupy ke shlukování objednávek z Denisovy práce, nebo se pokusíme navrhnout nějaké lepší strategie – v závislosti na typu skladů potenciálních zákazníků.

1.1.7 Další informace spojené s objednávkou

Tato část, více než s objednávkami, souvisí v důsledku spíše s dalším procesem navrženým v práci Bc. Denise Talára – proces balení vyskladněných produktů do jednotlivých zásilek. [3] I tak je ale nutné tuto doménu řešit již při vzniku objednávek – proces balení totiž od jednotlivých objednávek očekává určité vstupní informace.

1.1.7.1 Informace o dopravě objednávky

V návrhu balení jsou detailně popsané části procesu, kdy se na vybraném skladovém umístění balí veškeré produkty do jednotlivých zásilek a poté je pro každou z těchto zásilek vytištěn štítek, na kterém jsou informace určené pro dopravce, jako je hmotnost zásilky a další údaje vztahující se způsobu dopravy. V současném stavu je tisk štítku řešen přímo skrze systémy třetích stran, ze kterých byly objednávky odeslány a případná váha zásilky se na štítek prostě po vytištění dopisuje ručně. Jedním z cílů Denisova návrhu je tak kompletní řešení této záležitosti uvnitř skladu a zbavení se tak závislosti na systému třetí strany v oblasti tisku štítků a zároveň dosažením dalšího zjednodušení pro pracovníky. [3]

Z tohoto vyplývá, že v době balení objednávek bude nutné mít tyto následující informace:

- Do jaké objednávky balené produkty patří *(popsáno v předchozích kapitolách)*
- Váha zabalené zásilky
- Informace spojené s dopravou

Vzhledem k tomu, že prvním bodem jsem se zabýval již výše v analýze a informace v druhém bodě je možné zjistit až ve chvíli zabalení zásilky, zbývá zde k prodiskutování jen poslední bod – informace spojené s dopravou.

V současné době je informace o dopravci skladovému systému poskytována až při finalizaci vyskladnění na určité místo – je zde několik problémů spojených s tímto přístupem. Vzhledem k tomu, že způsob dopravy by se měl volit ke každé jednotlivé objednávce, vzniká zde problém, pokud se vyskladnění skládá z více než jedné objednávky (k čemuž dochází, pokud se objednávky simulují úlohami vyskladnění a pak slučují). Tento přístup je však ne zcela optimální i při využití externích objednávek, které se na jednotlivá vyskladnění mapují v poměru 1:1 – při založení objednávky nemáme žádnou informaci o dopravě a tato informace se musí zjišťovat až při dokončování vyskladnění asociovaného s danou objednávkou, obvykle z nějakého systému třetí strany (e-shop). Z výše uvedeného plyne tedy skutečnost, že je vybírání dopravy v této podobě vhodné hlavně jen k vyskladněním, které vznikly bez jakékoliv iniciující objednávky.

Navíc dalším současně nevyhovujícím aspektem, který mi sdělil zadavatel [4], je, že informace o dopravě obsahuje v současném systému pouze název jednotlivých dopravců, ovšem nejsou zde žádné další přídavné informace o tom, jak má být balíček doručen – dopravci dávají často při doručení na výběr z různých služeb, jako třeba, kam přesně zásilku doručit. Mimo tyto služby asociované s typem doručení dopravci požadují i informace o platbě zásilky *(zaplaceno kartou, na dobírku, ...)* a dále se zde mohou případně i vyskytovat další nepovinné údaje *(například označení: Křehké, 18+, ...)*. [8]

Bude tedy určitě potřeba zajistit možnost uvedení veškerých těchto informací již při vytváření objednávky.

1.2 Souhrn analýzy

V následující kapitole se tedy zaměřím na veškeré mnou identifikované nedostatky a pro všechny z nich vypracuji návrhy. Abych shrnul veškerá témata analýzy, která budou předmětem návrhu, jedná se v první řadě o navrhnutí konceptu poptávky jako takové, kde budu dbát na to, abychom se zbavili veškerých nevyhovujících aspektů popsaných v analýze, ať už se jedná o vytváření objednávek za pomoci úloh vyskladnění, nebo externích objednávek. V návrhu budu chtít u poptávek umožnit jejich částečné uspokojování. Dále se budu z velké části věnovat konceptům, které by umožnily manipulaci s velkým množstvím objednávek a hlavně celkovou automatizaci jejich zpracovávání. V neposlední řadě vypracuji i návrhy, které by měly zajistit, že budou při procesu balení dostupné všechny potřebné informace pro expedované objednávky.

Kapitola 2

Návrh

V následující kapitole se zaměřím na návrh změn, které by měly odrážet zanalyzované části systému z předchozí kapitoly. Adresuji mnou vyslovené nedostatky a místa, kde jsme nalezli prostor pro zlepšení.

U návrhů jednotlivých částí se budu snažit postupovat iterativně, v každé iteraci budu nejprve formulovat myšlenky, které mě vedly ke konkrétnímu návrhu. Pokud to bude potřeba a bude se jednat o návrh, který zahrnuje nějaký složitější a potenciálně více rozvětvený proces, popíšu nejdříve průběh tohoto procesu. V dalším kroku se pak u dané problematiky zaměřím na vypracování návrhu GUI (ve všech případech půjde o wireframe). Z těchto dvou prvotních částí by pak měly automaticky vyplynout potřebné implementační změny, které navrhnu s ohledem na současný technický stav systému. Posledním krokem je prezentování těchto prvotních návrhů zadavateli – budeme vždy konzultovat, co by mělo být v návrhu jinak, případně, co v něm chybí (a tím odstartujeme potenciálně další cyklus iterace) a v neposlední řadě, zda má smysl daný návrh v současné chvíli řešit, nebo dát prostor jiným více perspektivním návrhům a tento návrh odložit k pozdějšímu dokončení a samotné realizaci.

Po poslední iteraci vypracování všech těchto vrstev návrhu, kdy se nám bude konečně výsledek líbit, by pak mělo být možné začít s implementací finálního návrhu.

Z důvodu velkého množství iterací jednotlivých návrhů a s tím spojeným velkým množstvím vypracovaných materiálů, budu v textu vždy ukazovat a popisovat hlavně výsledky až z finální iterace.

V jednom z návrhů budu částečně vycházet z práce Bc. Denise Talára. [3] V textu zhodnotím jeho návrh a některé části zkusím využít, nebo se z nich alespoň inspirovat.

2.1 Návrh konceptu poptávek

*V následujícím textu jsou často používány (a občas zaměňovány) termíny **poptávka** a **objednávka**. Aby nedošlo ke zmatení čtenáře – objednávka je v našem kontextu abstraktní pojem, který zkrátka značí nějaký požadavek na zboží. Poptávkou je na druhou stranu myšleno trochu více konkrétněji, že se jedná o námi navrhovaný vylepšený koncept objednávek, oproti těm dosavadním v systému. Poptávka pak značí i skutečnost, že dotyčný odběratel produkty nějakým způsobem poptává/rezervuje, i když nejsou například v tu dobu na skladě dostupné.*

Při návrhu nového konceptu poptávek do stávajícího systému bylo zřejmé, že bude potřeba eliminovat výše popsané nedostatky, které se vyskytují ať už u provizorního procesu vytváření objednávek pomocí vyskladnění, nebo přímo u externích objednávek.

Když po konzultaci se zadavatelem [4] shrnu naše požadavky, tak by poptávky měly, na rozdíl od stávajících způsobů objednání produktů, splňovat následující:

- Možnost částečného uspokojení poptávky
- Možnost změny produktů v již vytvořené poptávce
- Možnost zrušení objednávky nebo jen některých jejich produktů
- Mít poptávku asociovanou ke konkrétnímu uživateli po celou dobu jejího životního cyklu
- Schopnost při procesu balení produktů dohledat, do které poptávky produkty patří
- Mít potřebné informace pro tisk štítku při procesu balení
- Mít k dispozici historii veškerých změn v poptávce

Další požadavky, které se, spíše než ke konkrétní jedné poptávce, váží k celkovému nakládání a zpracovávání většího množství poptávek:

- Možnost shlukování poptávek do jednotlivých vyskladnění
- Manuální shlukování poptávek
- Automatické shlukování poptávek na základě sofistikovanějších strategií
- Lehce přístupná evidence poptávek konkrétních odběratelů
- Možná asociace vrácení produktů odběratelem k jeho konkrétním poptávkám

Jedná se o shrnutí bodů ze mnou sestaveného detailnějšího katalogu požadavků, který je možné si prohlédnout v příloze.

V prvních částech návrhu tedy přihlédnu více k poptávce, jako jednotné entitě a postupně budu přecházet k návrhům, které se týkají jejich hromadného zacházení.

2.1.1 Průběh životního cyklu poptávky a jejích produktů

Vzhledem k tomu, že námi definované požadavky kladou důraz na možnosti úprav poptávky (přidání, nebo odebrání produktů) a zároveň si dávají za cíl umožnit částečné uspokojování dané poptávky, vzniká zde nespočetné množství situací, na které je zapotřebí při návrhu myslet. Poptávka může mít každý svůj produkt v cela odlišné fázi uspokojení a na každý produkt může přijít v jakékoliv fázi požadavek na jeho vyškrtnutí z objednávky.

Jednotlivé kusy produktů objednávky se mohou nezávisle na sobě nacházet v následujících fázích:

1. Kus se nachází v poptávce a čeká, až bude pro něj vytvořeno vyskladnění *(v poptávce)*
2. Pro kus již bylo vytvořeno vyskladnění, které čeká, až na něm začne skladník pracovat *(ve volném vyskladnění)*
3. Kus se nachází ve vyskladnění, na kterém právě pracuje skladník *(v přiřazeném vyskladnění)*
4. Kus je vyskladněn na umístění, kde čeká na proces balení *(vyskladněn)*
5. Kus je balen do zásilky v procesu balení *(balen)*
6. Kus je zabalen na umístění, kde čeká na odeslání *(zabalen)*
7. Kus byl předán dopravci *(odeslán)*

Vypracoval jsem průběh procesu zpracování jednotlivých kusů produktů z poptávky, kde jsem pro každou z těchto 7 fází, kde se mohou kusy nacházet, znázornil postup, který by se dodržoval při jejich odebrání z objednávky. U první i druhé fáze jsem znázornil mimo jiné i možné přidávání produktů do poptávky – jedná se o jediné fáze, kde nově přidané kusy mohou začít svůj životní cyklus v rámci poptávky. Průběh jsem znázornil pomocí jednoho velkého activity diagramu, jehož některé jeho části vložím přímo do této sekce pro názornost (obrázek 2.1) – v pozdějších fázích nebylo reagování na změny kusů v poptávce jednoduché.

Pro nakreslení zmíněného activity diagramu jsem, stejně jako pro všechny ostatní tyto diagramy, použil software Enterprise Architect [9]. Tento software jsem zvolil, protože jsem s ním byl již seznámen v rámci několika předmětů během studia na FIT ČVUT a měl k němu školní licenci.

Celý průběh zpracování, včetně vytvoření poptávky, je pak popsán navíc ve scénáři, který je možné nalézt v příloze. Dovolím si z něj vyjmout jeden z příkladů, který popisuje, jak by se postupovalo v konkrétní situaci při editaci obsahu poptávky, která má své kusy produktů v různých fázích:

Předpoklad: *Odběratel chce místo* ***12*** *původně poptaných pneumatik jen* ***5****,* ***4*** *pneumatiky byly již odeslány* ***(fáze 7)****,* ***1*** *je poptána a není zatím vyskladňována* ***(fáze 1)****,* ***1*** *je ve volné úloze vyskladnění* ***(fáze 2)****,* ***2*** *jsou vyskladněny na umístění A a čekají na zabalení* ***(fáze 4)****.* ***4*** *pneumatiky jsou zabalené v krabici na umístění B a čekají na odeslání* ***(fáze 7)****.*

Postup: *Systém nejprve sníží počet kusů v produktu poptávky z 12 na 11 (za 1 zatím nevyskladňovaný poptaný kus), poté sníží počet kusů dále o 1 a zruší příslušné vyskladnění produktu ve volné úloze vyskladnění. Dále sníží počet o 2, zruší příslušné vyskladnění produktu a založí naskladnění z místa A s odkazem na tuto změnu produktu poptávky. V posledním kroku zbývá zrušit ještě 3 kusy – systém sníží počet u vyskladnění produktu pro zabalené 4 pneumatiky z 4 na 3 a založí naskladnění z umístění B s odkazem na tuto změnu produktu poptávky.*

Po další konzultaci se zadavatelem [4] jsme došli k názoru, že bude nejlepší umožnit odebrání kusu produktu z poptávky jen tehdy, když se nachází maximálně ve druhé fázi – ve volném vyskladnění. Procesy spojené s ostatními fázemi vyžadují vždy nějakou reálnou akci navíc ze strany pracovníka a jsou náchylné ke vzniku chyb. Navíc pro jejich zpracování není v současnou chvíli kapacita a jejich výskyt není tolik pravděpodobný.

2.1.2 Grafický návrh detailu poptávky

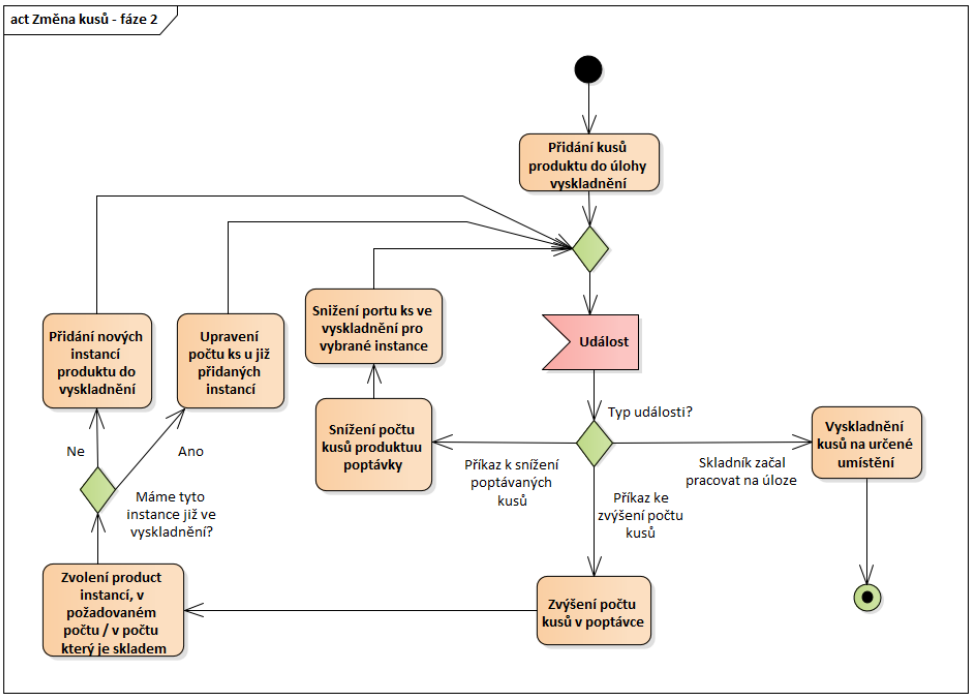
Po formulaci všech požadavků, popsání životního cyklu poptávky a ujasnění si, které veškeré průběhy chceme v dalších návrzích zahrnout, jsem se mohl pustit do vypracování wireframes.

Cílem wireframes je zejména prezentovat rozložení a celkovou strukturu uživatelského rozhraní. Mimo to ukazují však také průběh používání aplikace, její chování a funkcionality, které bude možné z rozhraní využívat. Tyto návrhy jsou pak velice užitečné pro obdržení zpětné vazby od uživatele, protože pomocí nich lze prodiskutovat a případně vrátit k přepracování veškerý obsah frontendové části, ještě před tím, než se vývojář pustí do zdlouhavého vývoje. Toto nám pomůže pochopit představu uživatele a ušetřit si obrovské množství času. [10]

Pro vypracování těchto návrhů jsem použil nástroj Balsamiq [11]. Jedná se o nástroj přímo určený pro tvorbu MI-FI wireframes.

Vzhledem k tomu, že celý návrh zabírá poměrně velké množství obrazovek, vložím přímo do textu jen část z nich – ty, které považuji z návrhu za nejzásadnější. Veškeré kompletní wireframes budou však dostupné k prohlédnutí na přiloženém médiu.

Obrázek 2.1: Změna počtu poptávaných kusů ve druhé fázi (ve vyskladnění)



2.1.2.1 Produkty poptávky

Na obrázku 2.2 můžeme vidět obrazovku, která se uživateli zobrazí, když se rozhodne pro zobrazení detailu konkrétní objednávky. První kartou v tomto detailu je karta s názvem **Produkty poptávky**. Zde budou zobrazeny informace, které jsou pro pracovníky skladu nejvíce důležité – aktuální stav uspokojení produktů v poptávce a možnost vybrat z nich libovolné kusy k dalšímu uspokojení.

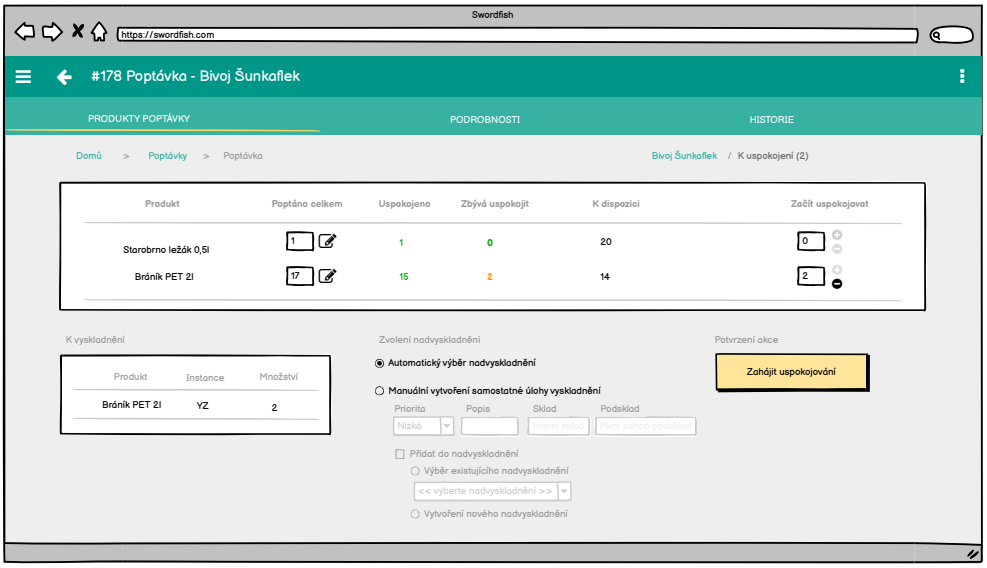
Prvním důležitým prvkem na této obrazovce je tabulka se samotnými produkty, je zde uveden název produktu, počet celkem poptaných kusů, počet dosud uspokojených kusů, počet kusů daného produktu, který je momentálně k dispozici ve skladu a v posledním sloupečku konečně počet kusů k dalšímu uspokojení.

Z návrhu je vidět, že sloupeček s celkem poptanými kusy je editovatelný – umožňuje vedoucímu upravit množství kusů, snížení počtu je však možné jen na hladinu již uspokojených kusů (ve fázi č. 3 a dále), pokud se pracovník pokusí snížit hodnotu pod tuto hranici, aplikace mu formou chybové hlášky oznámí, že takovou změnu nebylo možné provézt.

Podobné omezení logicky platí i pro pole, kde pracovník volí počet kusů, které chce nově uspokojit a přidat do vyskladnění. Není zde možné překročit počet kusů produktů, které jsou momentálně skladem.

Spodní část obrazovky se věnuje souhrnu kusů, které byly vybrány k uspokojení. První tabulka zobrazuje seznam vybraných produktů s jejich množstvím, nachází se zde také sloupeček, který určuje, jaká konkrétní instance poptávaného produktu bude vybrána při přidání do vyskladnění (pouze pro produkty, u kterých se evidují jejich sériová čísla nebo šarže). Prostřední část má sloužit pro vybrání již existujícího vyskladnění nebo založení nového, přímo pro tyto vybrané produkty. Vzhledem k tomu, že se však shlukováním produktů objednávek zabývám v dalších částech návrhu, nechám si detailnější vysvětlení tohoto konceptu na později. Posledním elementem na stránce je pak tlačítko, které potvrdí výběr produktů a zahájí jejich uspokojování.

Obrázek 2.2: Poptávka – Poptané produkty a jejich stav



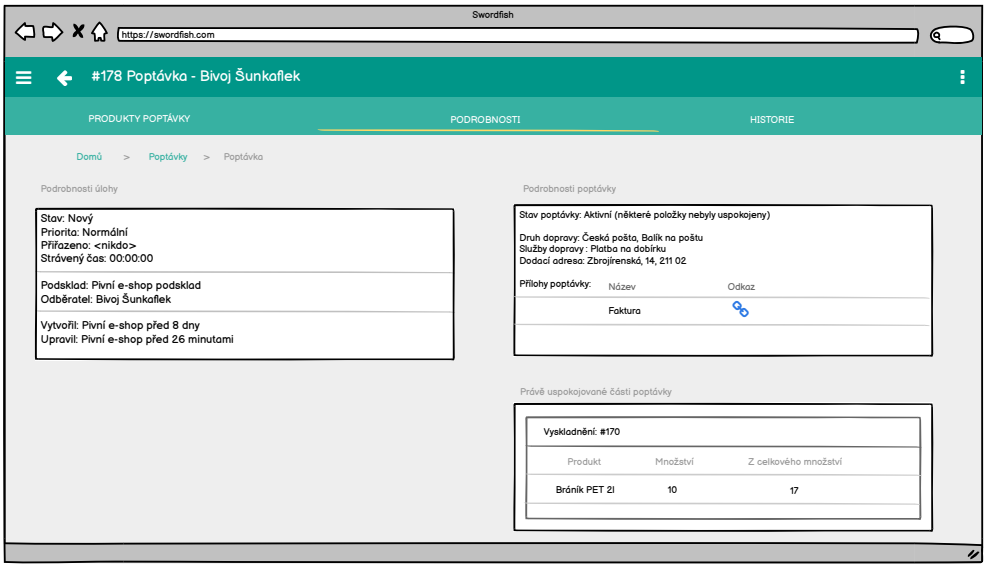
2.1.2.2 Podrobnosti poptávky

Další záložka detailu (2.3) je hlavně obecného informačního charakteru. Nalezneme zde běžné informace, jako u každé úlohy – stav, priorita, strávený čas… Mimo to zde však můžeme naleznout veškeré informace o zvolené dopravě objednávky (těmto informacím se věnuje jedna z následujících sekcí návrhu). Posledním prvkem na této obrazovce je tabulka zobrazující informace o právě uspokojovaných produktech poptávky – produkty, které zatím nebyly odeslány, ale pracuje se na jejich vyskladnění a balení (fáze 3–6).

2.1.2.3 Historie poptávky

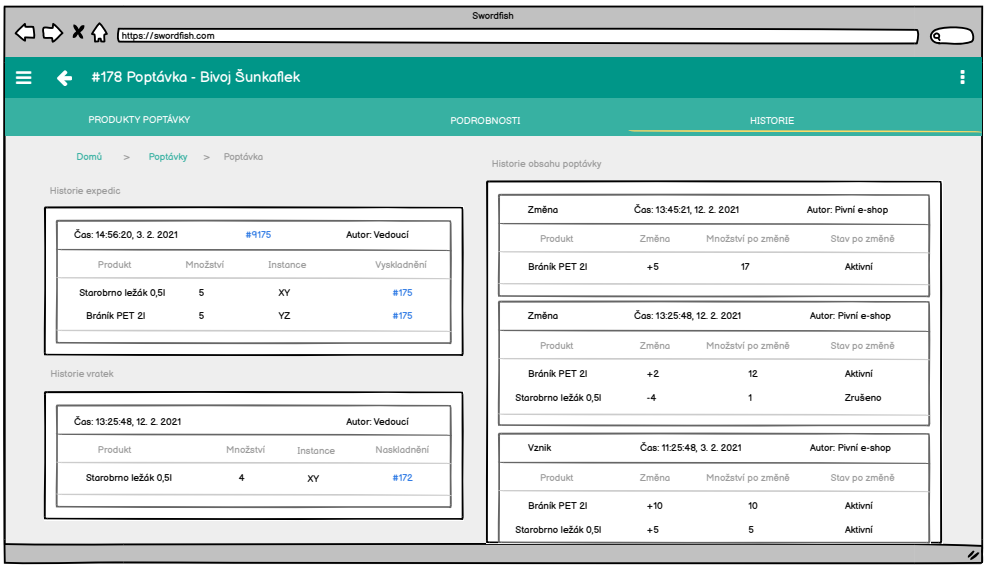
Poslední karta s názvem **Historie** se zaměřuje na veškeré záznamy spojené s historií produktů poptávky. Můžeme zde najít tabulku *(levý horní roh)*, která

Obrázek 2.3: Poptávka – Podrobnosti



nese informace o tom, jak a kdy byly spolu jednotlivé kusy produktů expedovány. Pravá tabulka má pak na starosti veškeré změny množství produktů v poptávce – kdo změnu provedl, čas změny, počet přidaných/odebraných kusů, počet kusů po změně. Poslední tabulka vlevo dole zobrazuje historii vráceného zboží odběratelem (na detailní popis evidence vrácených kusů se zaměřím detailněji později v návrhu).

Obrázek 2.4: Poptávka – Historie



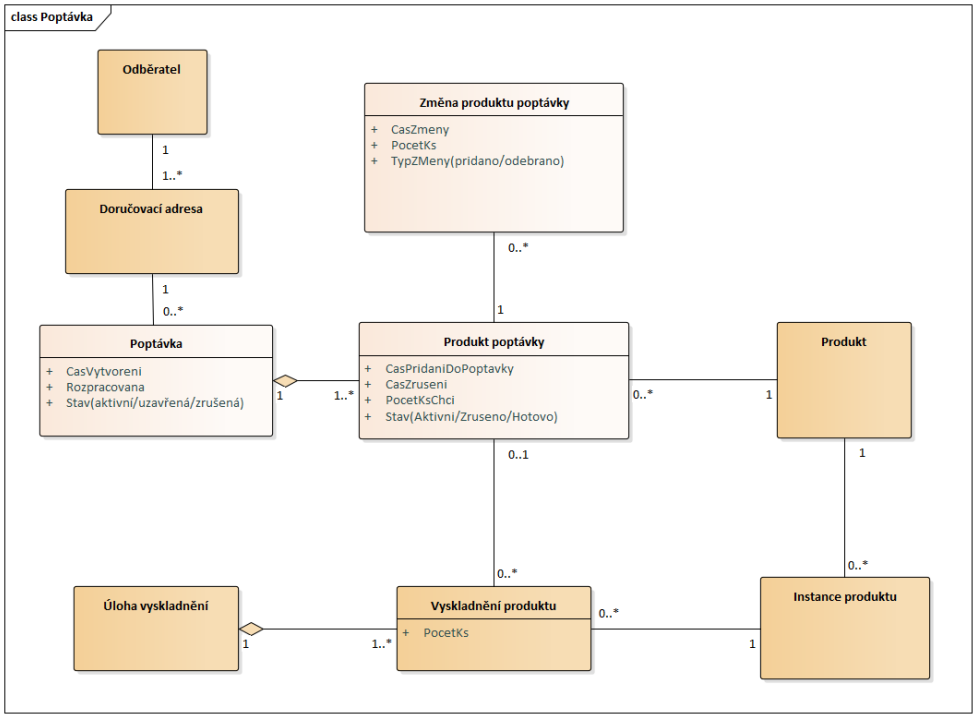
2.1.3 Zakomponování do současné podoby systému

Samotná poptávka by měla do systému v současné podobě přinést hned několik nových objektů, které budou asociovány s některými, již existujícími objekty. Dovolil jsem si vypracovat malý doménový model 2.5, který se zaměřuje čistě na nově přidané objekty související s poptávkami a jejich asociace s ostatními, již existujícími objekty.

Doménový model reprezentuje strukturu konceptů a objektů, které jsou součástí řešené domény. Zobrazuje jejich vzájemné rozdělení a relace mezi nimi. Model je vytvářen za účelem analýzy požadavků – jedná se o byznys úroveň pohledu a model je tedy abstrahován od jakýchkoliv technických a implementačních záležitostí. [12]

Objekty, které se v doméně skladového systému již nachází, budu vždy vyznačovat oranžovou barvou

Obrázek 2.5: Doménový model poptávky



Z modelu 2.5 je patrné, že momentální stav produktů v poptávce bude vždy dopočítáván za pomocí vícera entit – entity **Změna produktu poptávky** a **Vyskladnění produktu**.

Poptávka: Představuje samotnou poptávku s vazbou na odběratele, který objednávku vytvořil skrze jeho adresu.

Produkt poptávky: Typ jednoho produktu v poptávce s vazbou na konkrétní produkt evidovaný ve skladu.

Změna produktu poptávky: Historie změn počtu kusů jednoho typu produktu v poptávce, skutečný počet poptaných kusů je dopočítáván na základě této entity.

Vyskladnění produktu: V doméně již existující entita, patří pod konkrétní vyskladnění a implikuje, že jsou některé kusy daného produktu poptávky uspokojovány / již byly uspokojeny. Má vazbu na konkrétní instanci produktu – sériové číslo, nebo šarži.

2.2 Hromadné manipulace s poptávkami

2.2.1 Problém se shlukováním poptávek do vyskladnění

Moje původní představa byla, že dle návrhu 2.5 bude možné, kromě přidávání jen některých kusů do určitých vyskladnění, rovněž možné zahrnout produkty více poptávek do jednoho vyskladnění *(v návrhu vzniká mezi **Poptávkou** a **Úlohou vyskladnění** pomyslná N:N relace pomocí dvou mezilehlých entit).* Tato myšlenka se ukázala být jako správná, ovšem do chvíle, než jsme se zadavateli narazili na problém s vazbou na odběratele.

Systém v současné podobě spoléhá na to, že jednotlivá vyskladnění mají vždy informaci o tom, kdo je jejich odběratelem. Tuto informaci musí uživatel poskytnout již při vytváření úlohy – zkrátka tedy vyskladnění bez přiřazeného odběratele nemůže existovat. Tím pádem by zde nastávaly situace, kdy by se do vyskladnění s odběratelem A mohly přidávat produkty z poptávek, adresovaným zprvu jen odběrateli A, ale později i B, C, D, E... Vznikla by zde tím pádem datová nekonzistence. Při konzultaci se zadavatelem [4] jsme se shodli na tom, že dosavadní návrh, kdy má vyskladnění pouze jednoho možného odběratele, je neoptimální, avšak místo toho, abychom nějakým brutálním zásahem měnily dosavadní design, pokusili jsme se přijít s nějakým lepším návrhem, který by toto úskalí vyřešil.

2.2.2 Koncept pro shlukování vyskladnění – nadvyskladnění

Nápad, se kterým jsme při objevení výše popsaného problému se zadavateli přišli, by se dal popsat takto – nechme vyskladněním jejich vazbu na odběratele a přijměme skutečnost, že do každého vyskladnění půjdou vložit produkty jen z jediné poptávky, místo toho zkusme umožnit skladníkům efektivně manipulovat pomocí nějakého konstruktu s více těmito vyskladněními (a potažmo objednávkami) současně.

Na základě této naší diskuse jsem navrhl koncept, který by umožnil jednotlivá vyskladnění za určitých podmínek sdružit k sobě, do jedné velké skupiny. Tuto skupinu jsme začali pracovně nazývat: „Nadvyskladnění“.

Jako první jsem definoval podmínky, za kterých by jednotlivá vyskladnění měla být možná do tohoto nadvyskladnění sloučit. Dospěl jsem k názoru, že aby vyskladnění šlo obsluhovat současně v rámci jedné úlohy, musí být dodržena všechna následující omezení:

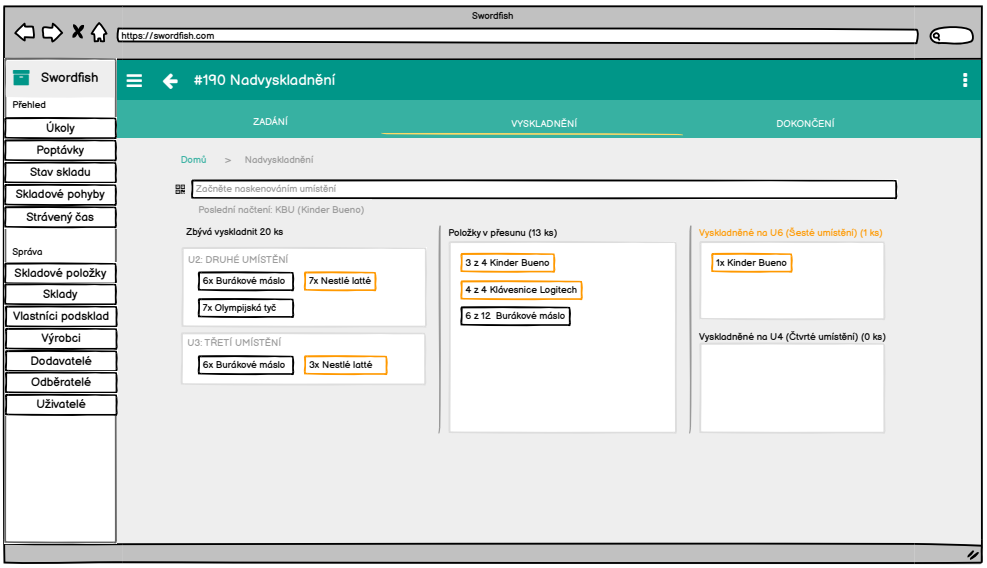
- Všechna vyskladnění musí mít stejnou úroveň priority
- Všechna vyskladnění musí mít stejné cílové umístění
- Všechna vyskladnění musí být ve stejném stavu
- Všechna vyskladnění musí patřit do stejného podskladu

Nadvyskladnění by pak mělo být možné v jakékoliv fázi smazat s tím, že by jeho vyskladnění dál samostatně existovala. Odebírání jednotlivých vyskladnění by mělo být rovněž podporováno v jakékoliv fázi. (jakoukoliv fází je myšleno kdykoliv, mimo probíhajícího zpracovávání skladníkem)

2.2.2.1 Grafický návrh nadvyskladnění

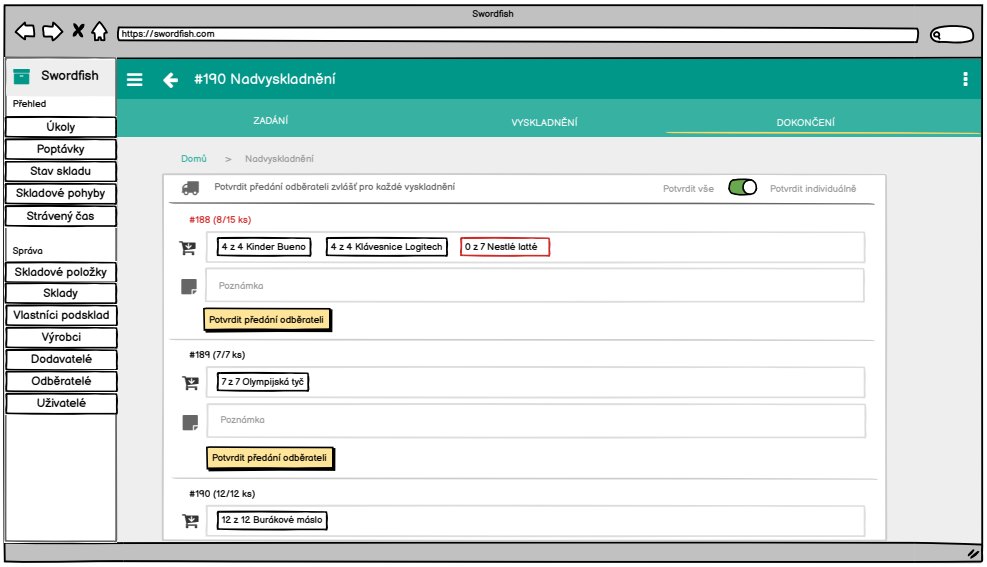
V původním návrhu jsme nejprve po konzultaci s Ing. Oldřichem Malcem [13] nejprve nechtěli zavést omezení na stejné cílové umístění poptávek, a tak jsem se návrh snažil zpracovat tak, aby mohl skladník pohodlně vyskladnit produkty i na vícero cílových umístění. Wireframe je možné vidět na obrázku 2.6. Nicméně jsme se po diskusi s hlavním zadavatelem dohodli, že by to bylo pro skladníky příliš náročné a naopak by jim to přidělávalo velké množství práce a nutného soustředění.

Obrázek 2.6: Vyskladnění na vícero umístění



V závěrečné kartě dokončení vyskladnění jsem znázornil, jak by mohlo vypadat potvrzování jednotlivých vyskladnění v rámci celého nadvyskladnění. U každého zahrnutého vyskladnění by měl být zobrazen přehled vyskladněných produktů, kde by skladník mohl napsat případnou poznámku a potvrdit dokončení každé úlohy zvlášť. Byla by zde rovněž možnost potvrdit všechny úlohy naráz, pokud by šlo pro skladníka vše hladce. Wireframe je možné zhlédnout na obrázku 2.7.

Obrázek 2.7: Dokončení nadvyskladnění



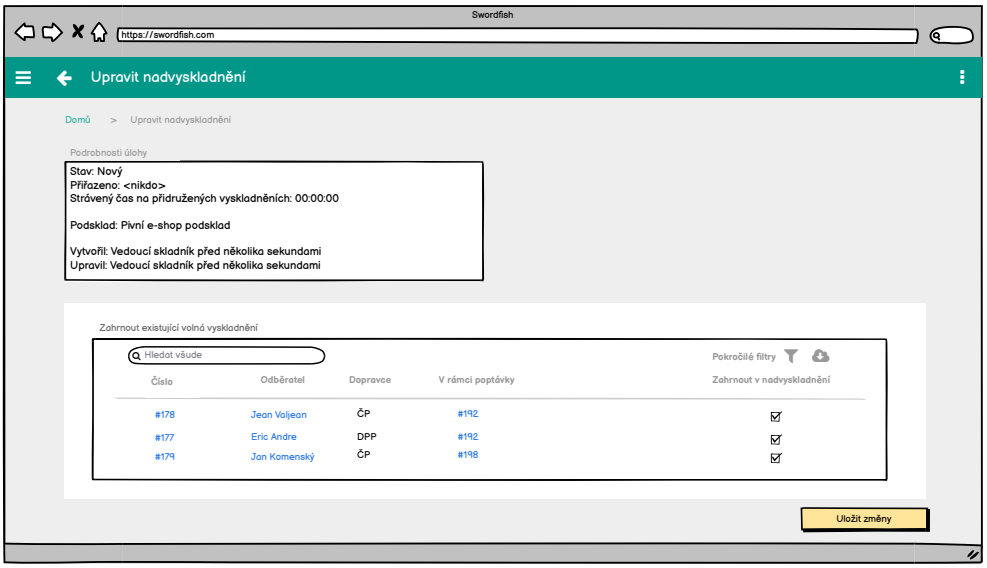
Po dalších konzultacích se zadavatelem jsme však došli k názoru, že bude pro maximální jednoduchost a pohodlnost skladníka nejlepší, když bude úloha nadvyskladnění co nejvíce zaměnitelná s běžnou úlohou vyskladnění – skladník by neměl mít vůbec informaci o tom, do kterého vyskladnění jaké produkty patří, toto by mělo být starostí výhradně vedoucího skladu. Rozhodli jsme se tedy pro roli skladníka využít při zobrazení stejné uživatelské rozhraní, kromě informace, že se jedná o jakousi „skupinu vyskladnění".

Role vedoucí skladu by pak měla mít možnost v kartě nadvyskladnění měnit zahrnuté vyskladnění a být schopná tyto nová nadvyskladnění zakládat. Ukázku s editací obsažených vyskladnění ukazuje obrázek 2.8.

2.2.2.2 Doménový model nadvyskladnění

Naše původní myšlenka zakomponování nadvyskladnění do stávajícího systému byla taková, že bychom uchopili celý koncept více univerzálně a umožnili vzniku seskupení libovolných úloh (ne jen úloh vyskladnění) 2.9. Ovšem vzhledem k tomu, že chceme mít ve velkém množství situací přístup ke specifickým atributům jednotlivých vyskladnění, které jiné úlohy nemají a zároveň

Obrázek 2.8: Úprava nadvyskladnění



je velice nepravděpodobné, že bychom chtěli podobný mechanismus seskupování v blízké budoucnosti aplikovat i pro jakýkoliv jiný typ úloh, rozhodli jsme se, že náš koncept nadvyskladnění bude sdružovat v tuto chvíli jen úlohy vyskladnění. Navíc jsem ve druhém návrhu 2.10 zohlednil i skutečnost, že nadvyskladnění by mělo samo o sobě být úlohou, vzhledem k tomu, že s ním bude skladník manipulovat stejně tak, jako s ostatními úlohami.

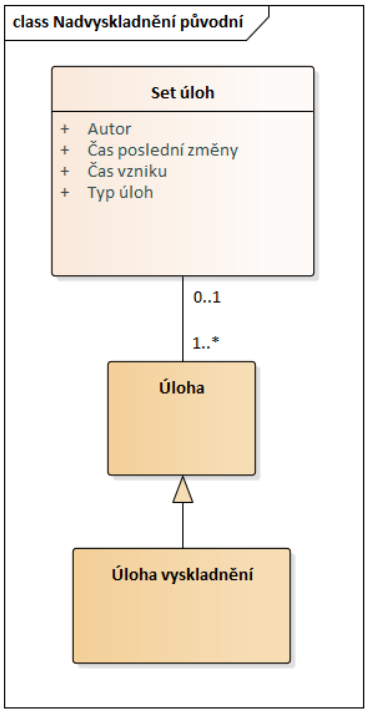
2.2.3 Hromadné uspokojování poptávek

Návrhem pro zpracovávání objednávek a jejich efektivní rozdělování do vyskladnění se zabýval již dříve Denis ve své bakalářské práci. [3] Jeho návrh cílil zejména na menší e-shopové sklady, ve kterých by se hojně využívalo manuálního rozřazování objednávek do jednotlivých vyskladnění, i přesto však ve svém návrhu představil možnou automatizaci, která by umožňovala rozřazování objednávek do vyskladnění dle dopravců, které si odběratelé zvolili.

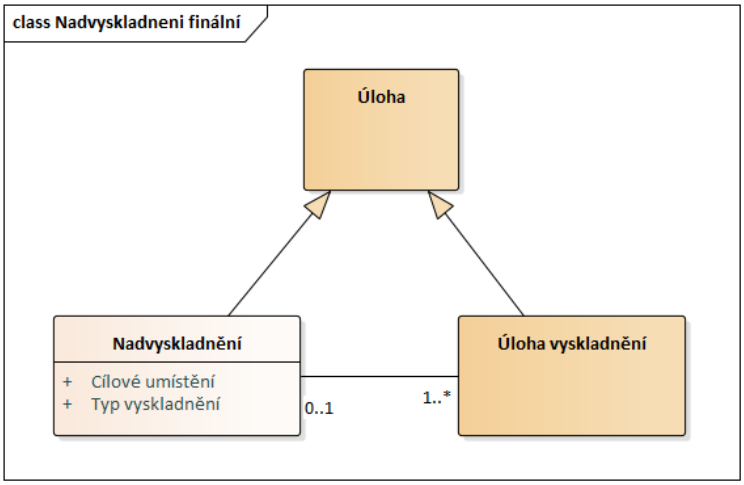
V mém návrhu budu brát v potaz jeho existující zpracování, ale zároveň v něm budu zohledňovat i mnou již navržené koncepty, jako je zejména možnost částečného uspokojení objednávek. Budu v něm také využívat moje předešlé návrhy, které byly potřeba zpracovat již v dřívějších fázích. Navíc, po další diskusi se zadavatelem, se pokusím proces navrhnout více univerzálně, aby byl pohodlný i pro větší sklady, než bylo zamýšleno v původním Denisově návrhu, protože se dle jeho slov spektrum zákazníků začíná rozšiřovat i na majitele velkých skladů s mnohem větším počtem příchozích objednávek.

Ve vypracovaných wireframes jsem se snažil na obrazovkách, které se věnují hromadnému zpracovávání poptávek, přistupovat více izolovaně k obsahu jed-

Obrázek 2.9: Nadvyskladnění – prvotní návrh



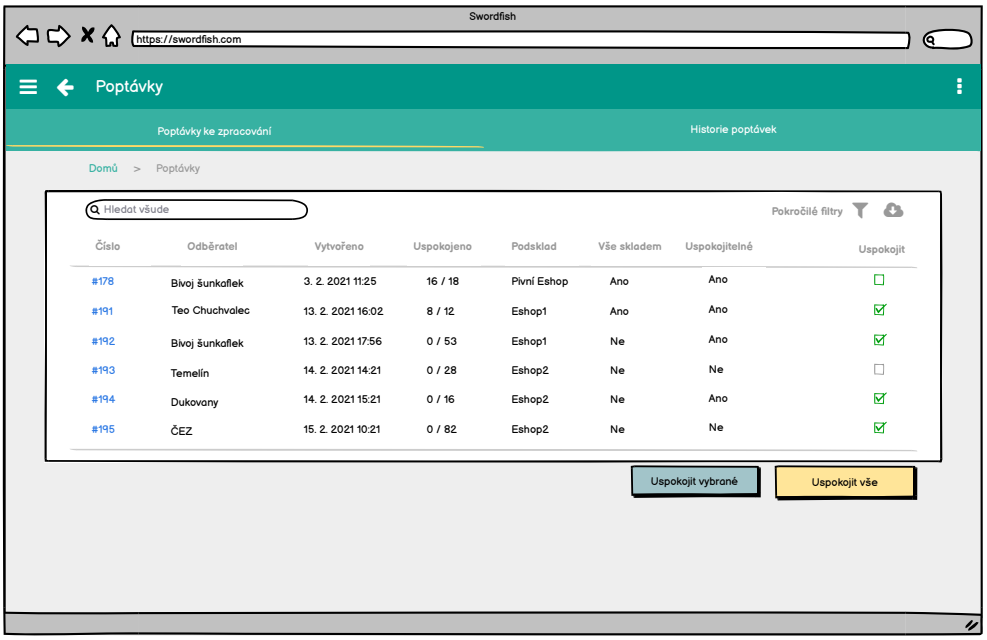
Obrázek 2.10: Nadvyskladnění – finální návrh



notlivých poptávek – cílem bylo, aby mohl uživatel pohodlně zpracovat rychle a efektivně velké množství objednávek, u kterých není žádný problém a na zlomek těch neuspokojitelných se pak mohl zaměřit zvlášť. Na obrázku 2.11 můžeme tedy vidět, že tabulka poptávek, které čekají na vyřízení, obsahuje

jen ty nejvíce prioritní informace potřebné k uspokojení – informace, zda je poptávka celá bez problému uspokojitelná, nebo zda dokonce nejde z objednávky uspokojit vůbec nic. Uživatel může buď sám pomocí checkboxů vybrat objednávky, které chce uspokojit, nebo jednoduše použije tlačítko uspokojit vše a systém vybere jen ty objednávky, u kterých je možné uspokojit alespoň část jejich produktů.

Obrázek 2.11: Zpracování poptávek – výběr poptávek



Po zvolení objednávek, které mají být uspokojeny, by systém zanalyzoval jejich poptané produkty a pokud by došlo k situaci, že by nebylo dostupné dostatečné množství některého produktu a tím pádem nebylo možné uspokojení poptaných produktů ze všech objednávek naráz, systém by zobrazil dvě tabulky. V první z nich by jen uživateli stručně vypsal produkty, které nejsou na skladu vůbec, a tak je není možné uspokojit v žádné z vybraných objednávek, nebo produkty, které nejsou dostupné v poptávaném množství, ale jsou poptány pouze jednou z objednávek, takže je rozdělení kusů triviální – po rozbalení jednotlivých řádků s produkty by systém zobrazil informaci, kterých objednávek se toto přesně týká.

V druhé tabulce by se naopak nacházely konfliktní produkty – ty, kterých není na skladě dostatečné množství, ale „přetahuje“ se o ně vícero objednávek a je na uživateli, aby rozhodl, která objednávka dostane jaké množství kusů. Uživatel může buď manuálně zadat toto množství u jednotlivých objednávek po rozbalení řádku s konfliktním produktem, nebo může pro urychlení práce využít tří automatických postupů:

Přerozdělit od nejstarších: Tento postup v přerozdělování jednoduše dá

přednost těm objednávkám, které byly vytvořeny dříve.

Přerozdělit poměrově: Rozdělení v poměru k poptanému množství produktu v jednotlivých objednávkách.

Přerozdělit rovnoměrně: Rovnoměrné rozdělení, které se pokusí přiřadit všem poptávkám stejné množství kusů.

Výše popsané tabulky lze nalézt na výstřižku z wireframu 2.12.

Obrázek 2.12: Zpracování poptávek – částečné uspokojení

V posledních částech tohoto procesu (2.13) by měl mít uživatel možnost zhlédnout stručný souhrn aktuálního zpracování – informace o tom, jaké poptávky budou uspokojeny, kolik produktů bude z objednávky uspokojeno a celkový počet kusů, který bude nutné pro každou objednávku vyskladnit. Jako poslední pak přichází na řadu zvolení způsobu, jakým se poptávky dostanou do jednotlivých vyskladnění – při tomto by se využívalo již dříve navrženého

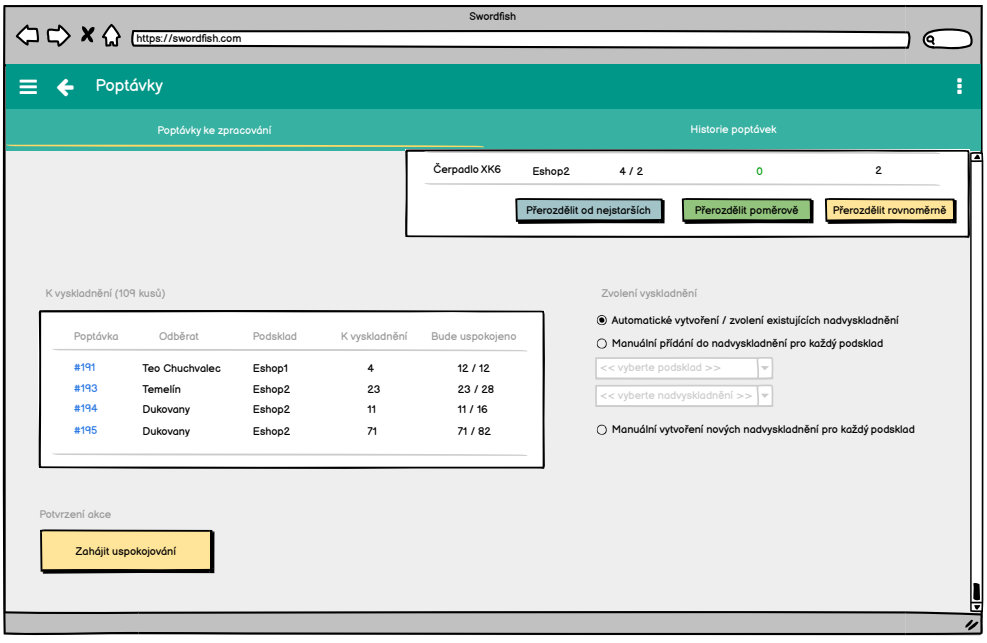
konceptu nadvyskladnění pro shlukování úloh. Uživatel by měl na výběr z následujících možností:

Automatické vytvoření/zvolení nadvyskladnění: Při tomto výběru by systém sám od sebe přerozdělil co nejvýhodněji objednávky do již existujících nadvyskladnění, nebo pro ně vytvořil nové – automatizací tohoto procesu a vymýšlení strategií, na základě kterých by se měly objednávky seskupovat, se budu věnovat v následující podkapitole.

Přídání do nadvyskladnění: Uživatel by měl při zvolení této možnosti sám vybrat pro každý podsklad nadvyskladnění, do kterého budou objednávky asociovány s tímto podskladem přidány.

Vytvoření nových nadvyskladnění: Tato poslední možnost volí nejprimitivnější postup, který pro každý podsklad, pro který jsou objednávky zpracovávány, vytvoří nové nadvyskladnění a objednávky do něj jednoduše přidá.

Obrázek 2.13: Zpracování poptávek – souhrn a rozdělení do vyskladnění



2.2.4 Automatické zpracovávání poptávek

V předešlé sekci jsem při popisování wireframu 2.13 načal jedno velice důležité téma – automatické shlukování objednávek do vyskladnění.

Tento koncept byl uveden již v Denisově práci, kde počítal s tím, že by byl v systému jakýsi automatický management objednávek, který by seskupoval

automaticky poptávky na základě dopravců. V jednom pozdějším prototypu je dokonce vidět, že by jako konfigurační data pro tento automat nemusely tvořit jen údaje ve formě časů příjezdů jednotlivých dopravců (na základě těchto by automat prioritizoval jednotlivá vyskladnění), ale i údaje o tom, kolik je na skladě momentálně pracovníků, aby byl zvolen optimální počet vyskladnění, které mohou být v jeden čas vytvořeny. [3] Tento koncept byl však představen jen velmi abstraktně a bez konkrétních detailů, jak by měl přesně fungovat. Rozhodl jsem se tuto problematiku rozpracovat mnohem podrobněji, vzhledem k tomu, že je pro sklady s velkým množstvím objednávek stěžejní, jako nikdy dříve.

2.2.4.1 Výběr vhodných strategií

Strategií, podle kterých postupovat při sestavování dávkového vyskladnění, je celá řada. Je samozřejmě možné zvolit ten nejjednodušší postup a veškeré příchozí objednávky seskupovat chronologicky, dle data jejich vytvoření do jednotlivých vyskladnění s předem definovanou maximální kapacitou objednávek nebo jednotlivých kusů – tento způsob by byl jistě lepší, než vyskladňování každé objednávky zvlášť, ale rozhodně by nedosahoval kvalit a ušetřené práce, kterou přináší mnohem promyšlenější strategie. [15]

Při výběru vhodných objednávek do společného vyskladnění jde nejvíce o minimalizaci vzdálenosti, kterou skladník musí urazit, aby vyskladnil veškeré objednávky v rámci přiděleného vyskladnění. Při tomto výběru se často berou v potaz nejen vzdálenosti mezi jednotlivými umístěními, ale i například vzdálenost produktů v rámci stejného umístění, velikost a váha produktů, počet umístění, které bude muset skladník navštívit pro uspokojení objednávky a nebo jen třeba samotný počet produktů v objednávce. [15]

Poměrně zásadní problém, který bohužel značně ztěžuje optimalizaci pohybu po skladu, je, že systém v současné chvíli neposkytuje žádná data o tom, jak jsou od sebe jednotlivá úložiště vzdálená, je proto nemožné v tuto chvíli použít algoritmy, které na tyto informace spoléhají. *Byl jsem ovšem informován, že na dalším rozšíření systému v tomto směru již v současné době pracuje jeden z týmů v rámci předmětu BI-SP1, pod vedením Matěje Humlíčka. [14]* Další chybějící informací jsou pak údaje o rozměrech a váze jednotlivých produktů – je tím pádem poměrně obtížné odhadnout, kolik kusů je možné do vyskladnění zahrnout, abychom zamezili tomu, že je skladník nebude schopen pobrat, nebo naopak půjde téměř s prázdnou.

Po zvážení všech možností jsme se rozhodli se zadavatelem nejprve sáhnout po strategii, která bude efektivní i bez znalosti výše uvedených informací. Jedná se o seskupení pouze těch objednávek, které poptávají pouze jeden totožný produkt (nehledě na množství poptaných kusů). [16] Vzhledem k tomu, že se produkty stejného typu shromažďují ve velkém množstvím na stejných umístěních, bude vyžadovaný pohyb skladníka logicky minimalizován, navíc vyskladnění jednotlivých kusů bude extrémně triviální a tedy i rychlou zá-

ležitostí, protože bude mít za úkol brát dokola pouze jeden identický typ produktu.

Naše první strategie je sice ve všech ohledech efektivní, ovšem problém nastává ve chvíli, kdy se na skladě momentálně nenachází tolik stejných jednopoložkových objednávek. Proto jsme se rozhodli vybrat další strategii, která si dává za cíl usnadnit spíše proces balení, který nastává po samotném vyskladnění – tato strategie seskupuje veškeré jednopoložkové objednávky i za cenu toho, že poptávají různé produkty.

Třetí námi zvolená strategie cílí také zejména na zjednodušení procesu balení. Chceme totiž za pomoci ní shlukovat ty objednávky, které jsou poptávaným zbožím naprosto identické.

Naší poslední, takzvanou odpadní strategií, bude ta nejprimitivnější metoda, která shlukuje jednotlivé objednávky na základě času jejich vytvoření – měla by zde být pro ty objednávky, které nebudou vhodné ke zpracovaní žádnou z předchozích strategií.

Abych tedy shrnul námi zvolené strategie, které budeme chtít v první verzi automatického zpracovávání zahrnout:

- Jednoprvkové objednávky se stejným produktem
- Jednoprvkové objednávky s různými produkty
- Identické objednévky
- FIFO – chronologicky, dle času vytvoření

2.2.4.2 Konfigurace automatického zpracovávání

Přestože výběr optimálních strategií je v této doméně nejvíce stěžejní záležitostí, není to stále jediná věc, kterou je zapotřebí navrhnout. Již jsem v minulosti kroužil okolo otázky, jak co nejlépe určit, kolik kusů produktů je v rámci jednoho vyskladnění vhodné zahrnout. Dále jsem připomněl jednu část Denisova dřívějšího návrhu, kde chtěl chytře pracovníkům skladu umožnit vyplnění počtu pracovníků, kteří se ve skladu zrovna nacházejí, aby to posloužilo jako nápověda pro systém spojená s informací, kolik může maximálně v jeden čas vytvořit úloh. [3]

Se zadavatelem jsme zvažovali způsoby, jakými vyřešit problematiku maximální kapacity jednoho vyskladnění. Moje prvotní úvaha byla limitovat je alespoň přibližným počtem kusů produktů – tento nápad sice není dokonalý, ale bez údajů týkajících se velikosti nebo váhy jednotlivých kusů produktů podle mého názoru neexistovalo lepší řešení. Zadavatel mi v tomto názoru oponoval tím, že by bylo v tomto případě přívětivější omezit místo toho vyskladnění maximálním počtem objednávek, které do něj půjdou zahrnout. Argumentoval tím, že bez informace o proporcích produktů stejně nemůžeme nikdy podle počtu kusů odhadnout, jak budou vyskladňované kusy dohromady těžké a rozměrné. Dále zmínil, že výhodou tohoto řešení je hlavně to, že při tomto řešení

není možné, aby se produkty z jedné objednávky „rozdrobily“ v jednu chvíli do více vytvořených vyskladnění, což by jistě způsobilo problémy při balení. V těchto aspektech jsem s ním plně souhlasil a rozhodl se tedy toto omezení určovat pomocí množství objednávek.

Co se týká druhé problematiky, její řešení je o něco přímočařejší. Místo Denisova návrhu se specifikováním počtu pracovníků pro vyskladňování a balení [3] jsem se rozhodl jít o trochu více transparentní cestou a umožnit uživatelům nastavit přímo maximální počet vyskladnění (v našem případě shluk vyskladnění – nadvyskladnění), které mohou být v jednu chvíli připraveny ke zpracování skladníky.

Po dalších konzultacích jsme se dohodli, že bude nejlepší, když rozsahem, pro který mají tyto konfigurovatelné parametry platit, nebude celý systém, ale pouze konkrétní podsklad. Bude tak umožněno mít pro každou část skladu, kde často může pracovat rozdílný počet pracovníků, nastavenou odpovídající konfiguraci.

2.2.4.3 Nastavení a rozšiřitelnost strategií

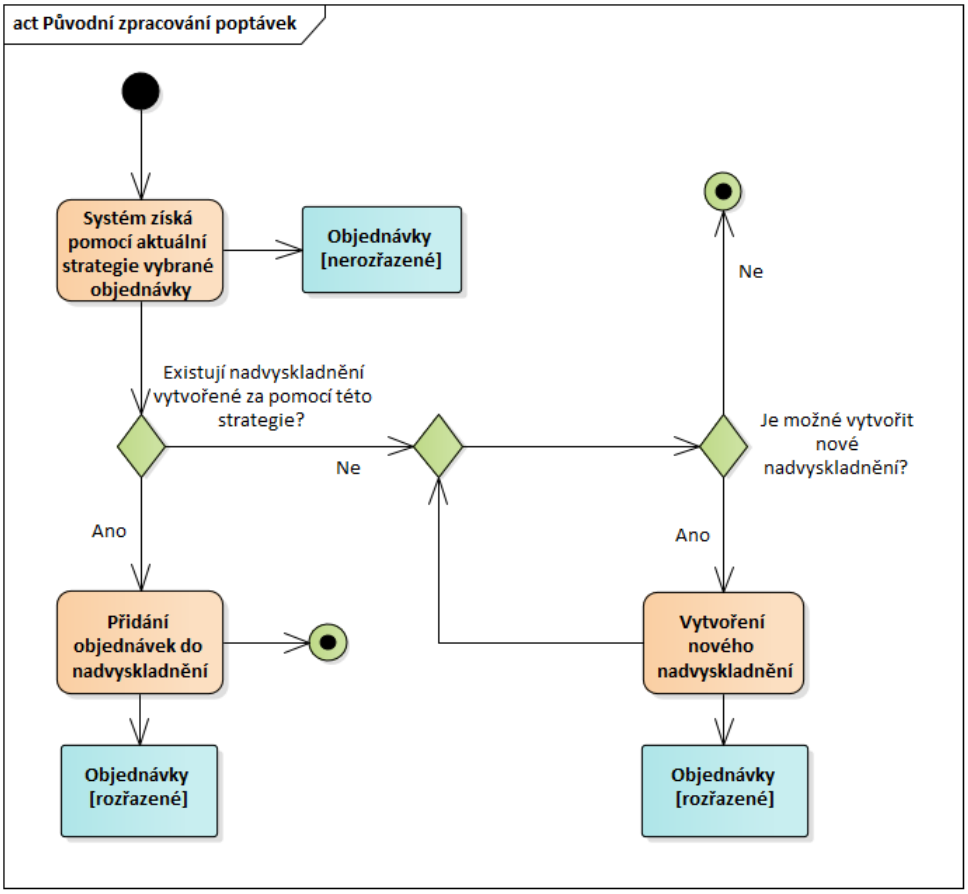
Už od počátku návrhu tohoto konceptu jsme se zadavatelem chtěli umožnit, aby strategie, které bude systém využívat, byly vysoce konfigurovatelné uživatelem a zároveň v budoucnu jednoduše rozšiřitelné. Požadavkem na veškeré další návrhy a pozdější implementaci bude tedy umožnit uživatelům výběr a zvolení pořadí strategií, které budou použity pro automatické zpracovávání objednávek, spolu s ostatními konfiguračními parametry pro konkrétní podsklad. Mimo to pak budeme chtít, aby bylo v budoucnu možné jednoduchým způsobem přidávat do systému nové druhy strategií – až budou například dostupné v systému informace o dimenzích a váhách jednotlivých produktů, informace o vzdálenosti úložišť a nebo se jen objeví potřeba pro vytvoření nové strategie na základě již existujících informací, jako jsou údaje o dopravě.

2.2.4.4 Průběh automatického zpracovávání objednávek

Při návrhu průběhu, jakým by měl systém postupovat při zpracovávání objednávek, jsem došel původně k návrhu 2.14, který vyznačoval, že se systém nejprve pokusí přiřadit veškeré objednávky do některých z již existujících úloh nadvyskladnění a až poté, pokud se některé objednávky nepodaří přiřadit, se pokusí o vytvoření nových nadvyskladnění od nejvíce prioritních strategií k těm nejméně. Bohužel jsem si později uvědomil, že tento prvotní návrh obsahuje jednu chybu – pokud by systém skutečně takto postupoval, bylo by zde riziko, že by objednávky, které by bylo možné rozřadit pomocí lepších strategií, končily místo toho přidané v nadvyskladněních využívajících více „odpadní“ strategie. Bylo proto nezbytné původní návrh předělat – při rozřazování je nutné se pro každou strategii pokusit nejprve najít existující nadvyskladnění a v případě neúspěchu se ihned v dalším kroku pokusit o založení nového, právě

s touto potenciálně výhodnou strategií. Finální návrh si lze prohlédnout na obrázku 2.15.

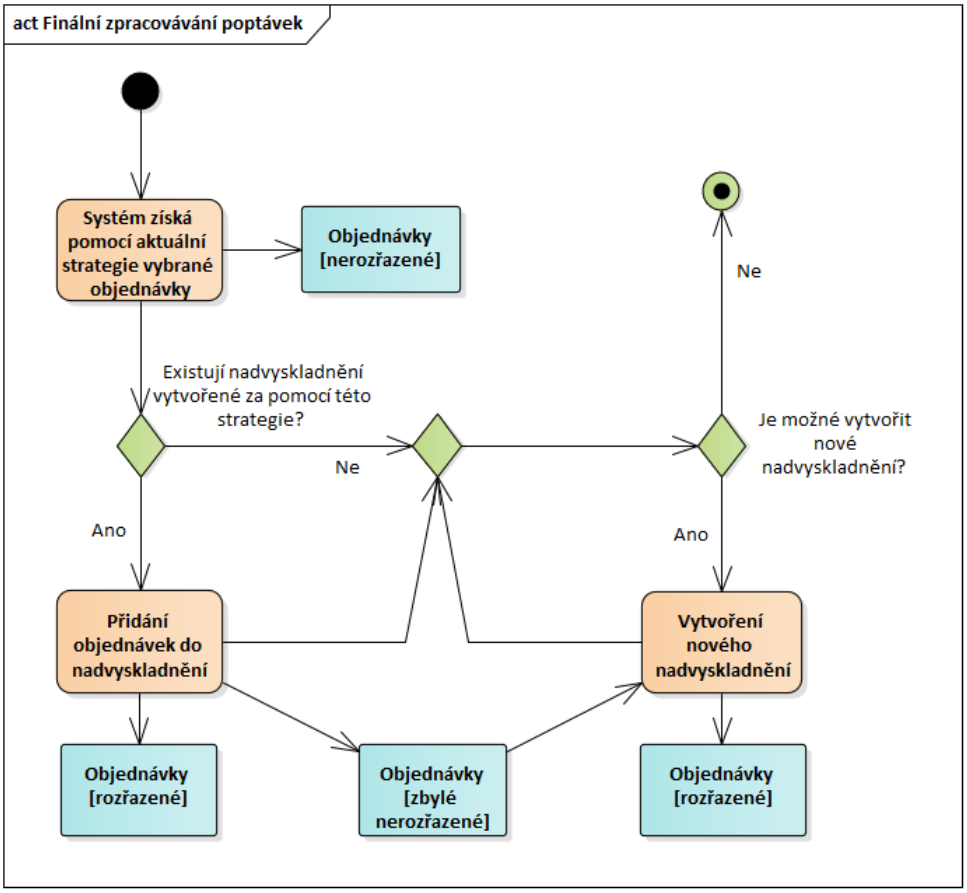
Obrázek 2.14: Zpracování poptávek – původní návrh průběhu



2.2.4.5 Návrh zakomponování na doménové úrovni

Následující doménový model 2.16 ukazuje způsob, kterým by bylo možné do současné podoby přidat konfiguraci automatického zpracovávání, která by byla svázána vždy s konkrétním podskladem. V druhé řadě pak zobrazuje možnou asociaci strategií a podskladu – každý podsklad by mohl využívat více různých strategií, které by se aplikovaly při vytváření zpracovávání poptávek v pořadí jejich priorit a v závislosti na tom, zda jsou aktivní (lze je použít). Každé z vytvořených nadvyskladnění by pak mělo jednak současnou vazbu na podsklad, do kterého patří, a v druhé řadě i nepovinnou vazbu na strategii, za pomocí které bylo vytvořeno (pokud bylo vytvořeno automaticky).

Obrázek 2.15: Zpracování poptávek – finální návrh průběhu



2.2.5 Přiřazování vrácených produktů k poptávkám

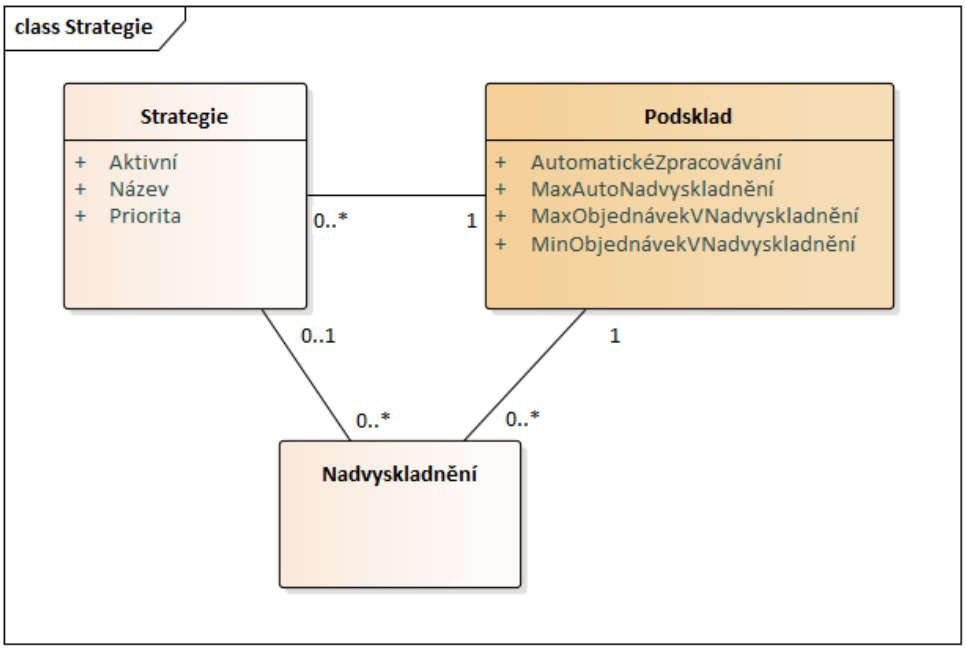
Tento další návrh se věnuje v analýze zjištěnému nedostatku, který bude nyní potřeba vyřešit pro pozdější možnost dopočítání reálných údajů o celkovém odebraném zboží konkrétním odběratelem. Jedná se o asociování vráceného zboží ke konkrétním poptávkám odběratele.

V systému se nachází úloha naskladnění, která slouží k přesunu nově přijatého zboží do skladu a tedy aktualizování stavu skladu o tyto nově naskladněné produkty. Při vykonávání procesu, ze kterého se úloha skládá, skladník načítá čtečkou zprvu vyskladnění, na které chce nové produkty naskladnit a po sléze veškeré produkty, které na toto umístění pokládá.

Je tedy evidentní, že abychom docílili našich požadavků, bude potřeba využít tohoto typu úlohy. Se zadavatelem jsme dospěli k názoru [4], že řešením bude možnost u zakládání této úlohy specifikovat, že se jedné o přijetí vrácených produktů od určitého odběratele.

V mém původním návrhu jsem chtěl umožnit uživateli při zakládání úlohy

Obrázek 2.16: Zpracovávání poptávek – strategie a konfigurace podskladu



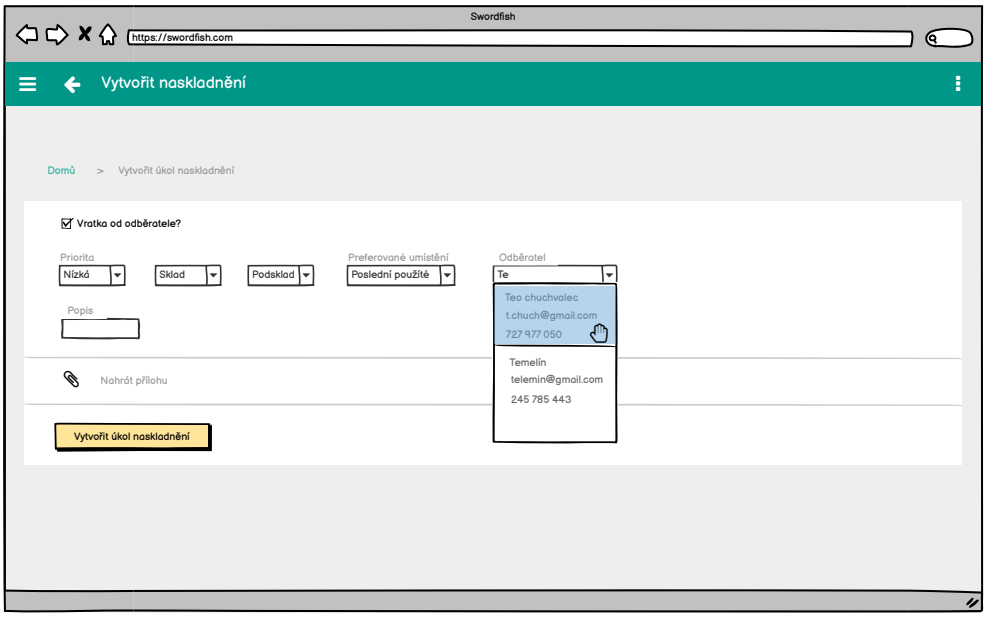
naskladnění pouze specifikovat odběratele, který naskladňované produkty vrací – systém by pak automaticky dohledal jeho příslušné poptávky, kde se vrácené zboží vyskytuje a asocioval je s tímto navrácením. Nastává tu však potenciální problém, kdy by skladník mohl načíst produkt, který odběratel v minulosti vůbec neodebral. Z tohoto důvodu jsme se se zadavatele rozhodli zvolit zjednodušený návrh, kde se místo načítání produktů skladním zvolí při vytváření úlohy přímo produkty, které odběratel v minulosti odebral. Tento návrh je možné zhlédnout na obrazovkách 2.17, 2.18.

Co se strukturálních změn v systému týče, byla by zde dle mého návrhu 2.19 potřeba pouze přidat jednoduchá volitelná vazba mezi úlohu naskladnění a příslušnými změnami u konkrétních poptávek – tyto změny by byly vytvořeny vrácením produktů a díky vazbě na úlohu naskladnění by bylo možné získat informaci, že byly produkty z poptávky odebrány vrácením zboží.

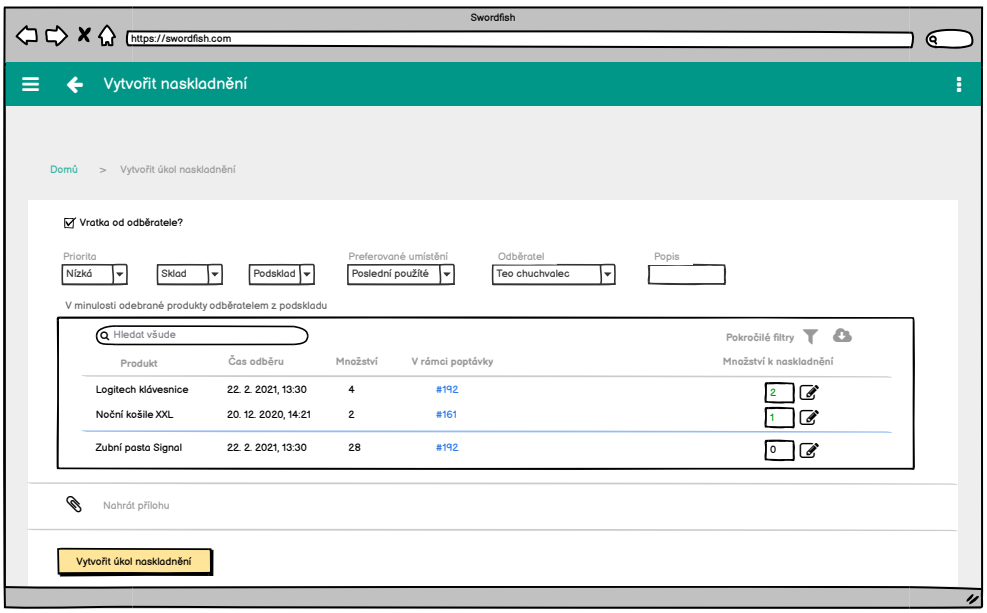
2.2.6 Evidence pohybů odběratele

Při dodržení výše uvedených návrhů by pak neměl být problém v systému zobrazit u konkrétních uživatelů veškeré expedice, které k nim směřovaly, nebo naopak veškeré jimi vrácené zboží a mimo to samozřejmé i jejich původní poptávky. Návrh karty pohybů konkrétního odběratele je ke zhlédnutí na obrázku 2.20 (v návrhu jsem mimo zmíněné věci zpracoval i tabulku, která by mohla evidovat, kolik kusů konkrétního produktu odběratel v minulosti odebral/vrátil).

Obrázek 2.17: Vrácení produktů – výběr odběratele



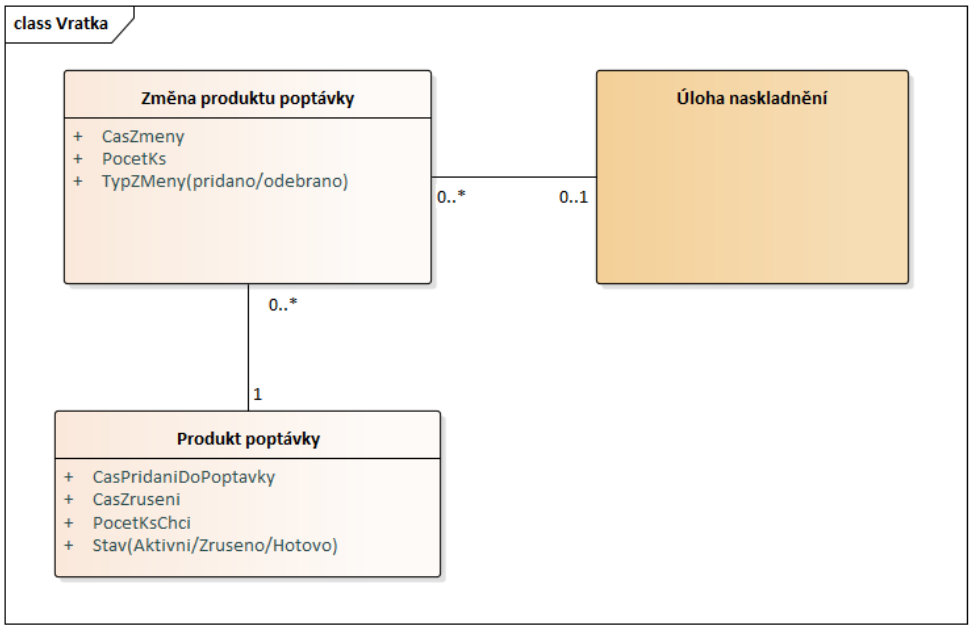
Obrázek 2.18: Vrácení produktů – výběr produktů



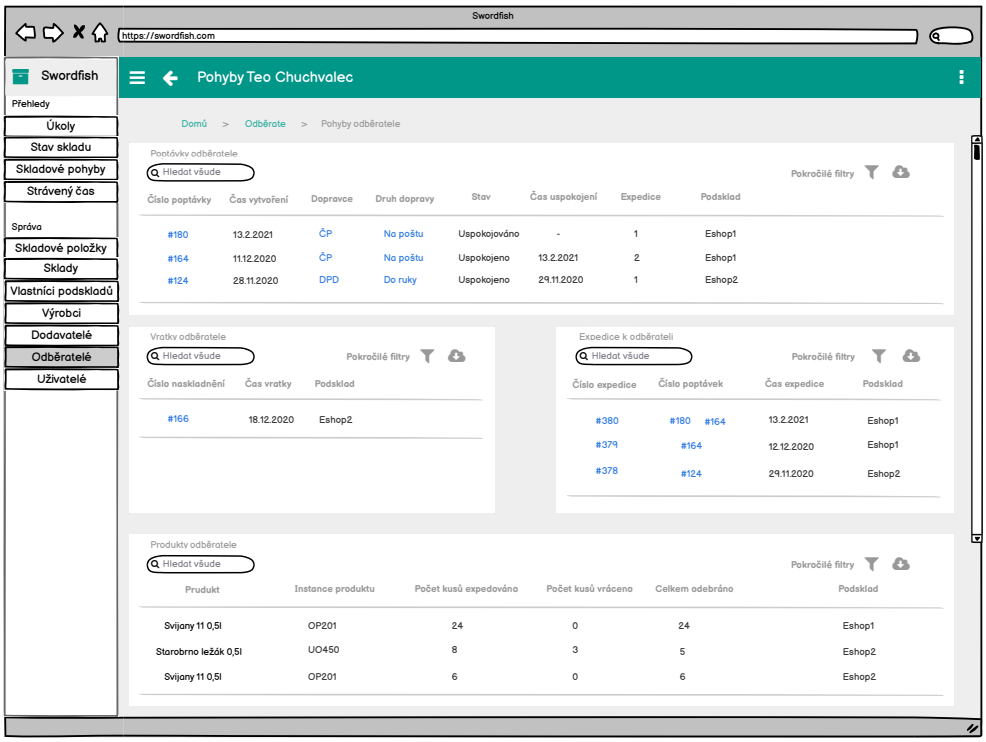
2.3 Údaje spojené s dopravou poptávky

V analýze jsem se v rámci podkapitoly 1.1.7.1 zabýval problematice ukládání informací spojených s dopravou objednávky. Z mých zjištění vyplynulo, že bude pro zachování těchto informací zapotřebí upravit stávající strukturu sys-

Obrázek 2.19: Vrácení produktů – změny v doméně



Obrázek 2.20: Wireframe – detail pohybů odběratele



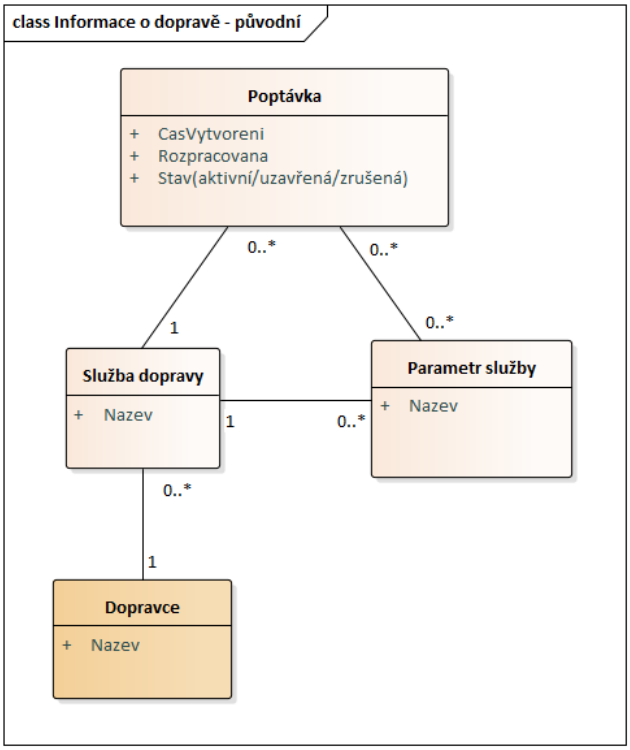
tému. Vzhledem k tomu, že má systém v současném podobě k dispozici pouze název dopravce, který je asociovaný až s konkrétním vyskladněním, bude potřeba na základě dřívější analýzy přidat k dopravcům jejich jednotlivé služby s různými parametry – tyto služby budou namísto vyskladnění nově vázány přímo na poptávku odběratele, protože bude žádoucí, aby systém třetí strany vyplnil tyto údaje již při založení objednávky. Návrh 2.21 popisuje mou původní myšlenku, která se však při implementaci ukázala být nekompletní a muselo dojít k její drobné úpravě.

Další změna, která z tohoto návrhu vyplývá je, že bude s nutné v rámci založení poptávky ze systému třetí strany posílat i tyto informace o:

- dopravci
- jeho službě, kterou si zákazník zvolil
- volitelně parametry, které lze v rámci zvolené služby uplatnit

Výpis veškerých těchto uvedených údajů by měl být externím systémům dostupný skrze API – systém třetí strany by jen s vytvořením objednávky zaslal identifikátory jím zvolených údajů a náš systém by ověřil, že je jím zvolená kombinace validní.

Obrázek 2.21: Doménový model – informace o dopravě

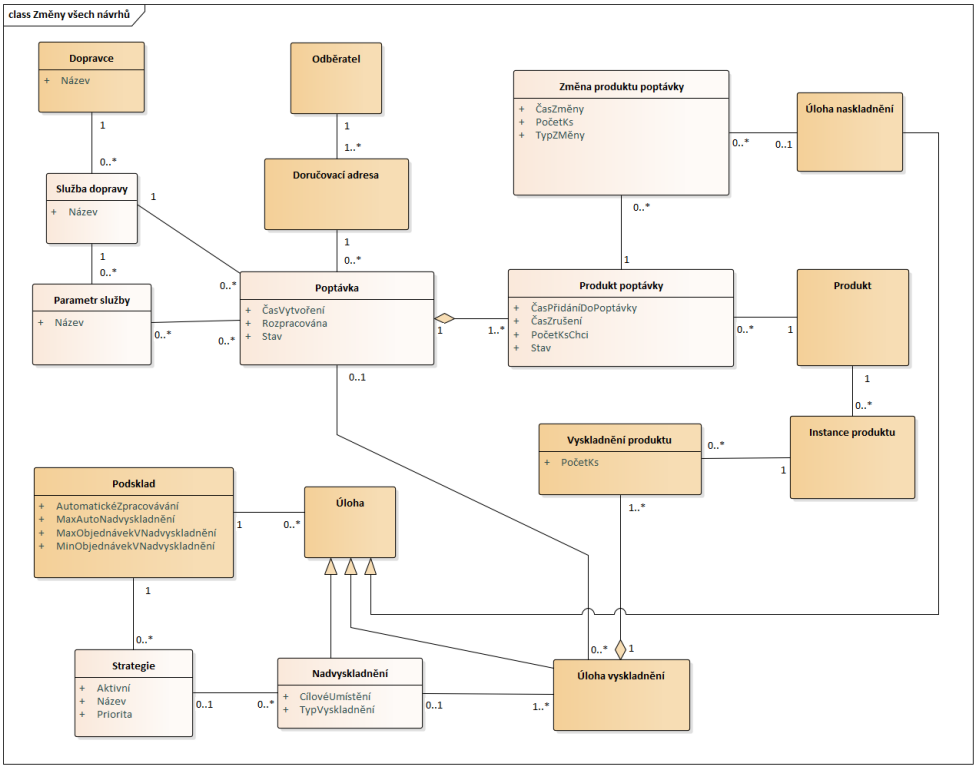


2.4 Souhrn návrhů

Následující doménový model 2.22 shrnuje veškeré strukturální úpravy, které vyplývají ze všech návrhů popsaných v této kapitole. Objekty, které v doméně již existují jsou opět vyznačeny oranžovou barvou.

V následující části práce se detailně zaměřím na některé z návrhů, které byly představeny v rámci této kapitoly a provedu jejich realizaci.

Obrázek 2.22: Doménový model – shrnutí



Kapitola 3

Realizace

V této kapitole se budu věnovat přímo implementaci návrhů, které společně se zadavatelem určíme za v tuto chvíli nejvíce prioritní a potřebné. Vzhledem k tomu, že jsem v rámci předchozí kapitoly pro každou oblast, na kterou jsem se zaměřoval, vypracoval řadu návrhů, měla by být implementace z velké části poměrně přímočará. I tak se však ukáže, že některé věci při implementaci změn nepůjdou přímo tak, jak byly původně zamýšleny v návrhu.

V rámci implementace budu zasahovat do frontendové i backendové části aplikace, které byly napsány v rámci diplomových prací Ing. Oldřicha Malce a Ing. Pavla Kováře. [1], [2] Z tohoto vyplývá, že drtivá většina technologií, které budu v této sekci využívat, je předem daná.

3.1 Volba návrhů pro realizaci

Z kraje kapitoly by bylo vhodné vyjasnit si, které návrhy budou nyní předmětem implementace. Po uvážení všech možností, jsme se zadavatelem dospěli k závěru, že návrhy, které budou nejvíce vhodné pro zákazníky s většími sklady, kteří budou v blízké budoucnosti systém využívat, jsou návrhy týkající se zejména automatického zpracovávání masivního množství objednávek. Jedná se tedy o návrhy popsané v podkapitolách 2.2.2 a 2.2.4. Naopak jsme uznali za vhodné odložit zpracování návrhů, které se věnují převážně jednotlivým objednávkám, nebo návrhy, které popisují různé přehledy a historii – tyto návrhy by byly jistě užitečné, ale v tuto chvíli zda máme zkrátka některé ještě prioritnější.

V poslední řadě jsme se pak dohodli na zpracování návrhu, který se věnuje údajům spojených s dopravou poptávky, který je popsán v kapitole 2.3. Vzhledem k tomu, že bude již brzy implementován do systému proces balení, tak bude vyžadováno, aby veškeré tyto informace pro jednotlivé zásilky byly k dispozici..

3.2 Realizace návrhu pro shlukování objednávek

Vzhledem k tomu, že pokud budeme chtít být schopní automaticky zpracovávat velké množství poptávek tak, jako v návrhu v kapitole 2.2.4, bude nejdříve zapotřebí implementovat návrh věnující se samotnému seskupování objednávek z kapitoly 2.2.2, aby pak systém mohl tento konstrukt při zpracovávání použít. Bylo by sice možné využít pro rozdělování objednávek do vyskladnění současné funkcionality slučování na základě stejného odběratele, ale to zcela určitě vzhledem k předchozí analýze a návrhům nechceme.

Mým cílem bude tedy v této podkapitole implementovat návrh pro seskupování vyskladnění do jednotlivých „nadvyskladnění" – tak, aby bylo více úloh vyskladnění možné seskupit a vyskladnit naráz. Zároveň budeme chtít zachovat informaci o odběrateli a vazbu na objednávku, ze které vyskladnění vzniklo, protože toto vše bude pak potřebné při úloze balení.

3.2.1 Implementace backendové části

Na základě připravených wireframes a doménového modelu se nejprve pustím do implementace backendové části systému, která je vyvíjena v jazyce PHP [17] za použití frameworku Symfony, který je přímo určený pro tvorbu webových aplikací. [18] Tato celá část aplikace pak využívá pro ukládání dat objektově-relační databázový systém PostgreSQL [19]. [19] Komunikace mezi aplikací a databázovým systémem je navíc odstíněna pomocí knihovny Doctrine, která se stará o objektově-relační mapování v jazyce PHP. [20]

Po důkladném seznámení se s výše popsanými technologiemi jsem se nejprve pustil do návrhu REST API endpointů, které by měly mít na starosti obsluhování HTTP požadavků spojených právě s mým konceptem nadyskladnění. Z těchto změn by pak dále měla částečně vyplynout i podoba databázových tabulek (samozřejmě i za pomocí abstraktního doménového návrhu).

3.2.1.1 Návrh REST API endpointů

Při návrhu endpointů pro manipulaci s nadvyskladněními jsem vycházel zprvu z patrných požadavků na funkcionalitu z wireframes 2.6, 2.7 a 2.8, která byla tohoto návrhu UI patrná. Z tohoto mi vyplynulo, že pro hlavní detail úlohy bude zapotřebí vytvořit endpointy pro:

- Vytvoření nadvyskladnění na základě identifikátorů existujících úloh vyskladnění
- Obdržení veškerých informací o konkrétním nadvyskladnění včetně jeho úloh vyskladnění
- Změnu úloh vyskladnění, které jsou součástí nadvyskladnění

Pro následující záložku úlohy, která se věnuje samotnému přesouvání produktů z umístění do skladníkova inventáře a naopak, bude nutné implementovat endpointy po vzoru samotné úlohy vyskladnění, protože výsledná frontendová část bude pro tyto dva typy úloh identická:

- Přesunutí produktu z umístění do inventáře
- Přesunutí z inventáře na umístění

V závěrečné kartě úlohy, kde se řeší zejména odevzdávání a schvalování úlohy, platí to samé, co v předchozím výčtu a bude zapotřebí implementovat identické endpointy jako v případě vyskladnění:

- Dokončení úlohy
- Schválení úlohy
- Odmítnutí úlohy
- Přiřazení úlohy

API celé backendové části aplikace je zdokumentované pomocí nástroje Swagger [21] – i když pro Symfony jistě existují balíčky, které by byly schopny na základě již implementovaných controllerů vygenerovat tuto dokumentaci automaticky, bylo dříve v týmu rozhodnuto, že se tak bude činit ručně, aby měli programátoři nad touto vzniklou dokumentací absolutní kontrolu. Dokumentace je ručně psána za pomocí značkovacího jazyka YAML.

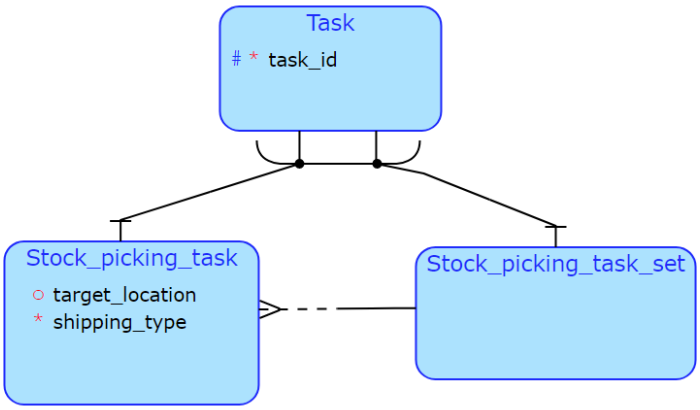
Nástroj textově popsanou podobu endpointů poté zobrazí v líbivé grafické podobě. V případě mnou navržených endpointů pro nadvyskladnění je možné výsledek zhlédnout na obrázku 3.1.

Obrázek 3.1: Podoba endpointů pro koncept nadvyskladnění

3.2.1.2 ER model

Na následujícím obrázku 3.2 je možné zhlédnout ER model [22], který popisuje konečné databázové změny, které vyplynuly z původního doménového modelu a podoby REST API endpointů.

Obrázek 3.2: ER model nadvyskladnění



3.2.1.3 Implementace logiky

Po dokončení rozhraní API a zanesení změn do databáze jsem začal psát kód, který popisoval chování pro zavolání každého z definovaných endpointů. V některých částech bylo toto lehce složitější v tom ohledu, že jsem pracoval na pozadí s více úlohami vyskladnění – například pokud vedoucí přiřadí skladníkovi úlohu nadvyskladnění, je žádoucí, aby se změnil tento stav i u jednotlivých úloh vyskladnění, které pod tuto úlohu patří. Vždy přitom chceme docílit atomického chování – buď se změna provede úspěšně pro všechny figurující entity, nebo se neaplikuje na žádnou z nich. Pro toto žádoucí chování jsem využil vlastností transakcí knihovny Doctrine. [23] Následující ukázka kódu 3.1 lehce demonstruje část této problematiky.

Výpis kódu 3.1: Přiřazení úlohy

```
/**
* Assign stock picking set to storekeeper.
* @param StockPickingSet $taskSet
* @param UserInfo $user
* @return bool
*/
public function assign(StockPickingSet $taskSet, UserInfo $user): bool
{
    if (!$user->hasRoleStorekeeper()) {
        return false;
    }

    if (!$taskSet->canAssign()) {
        return false;
    }
    // Begin transaction
```

```
    $this->entityManager->beginTransaction();

    $taskSet->setAssignedUserId($user->getUserId());
    $taskSet->setState(Task::STATE_IN_PROGRESS);

    foreach ($taskSet->getStockPickings() as $stockPicking) {
        if (!$this->stockPickingService->assign($stockPicking, $user)) {
            $this->entityManager->rollback();
            return false;
        }
    }

    $this->entityManager->persist($taskSet);
    $this->entityManager->flush();
    $this->entityManager->commit();
    return true;
}
```

Další náročnější částí se ukázala být implementace přesunů produktů jednotlivých vyskladnění, včetně jejich momentálního stavu – při každém pohybu bylo nutné procházet úlohy vyskladnění a hledat v nich předměty k přesunutí, někdy bylo nezbytné přesunout produkty v rámci více úloh vyskladnění současně. Co se stavu všech produktů týče, tak bylo nutné stavy identických produktů sloučit před odesláním klientovi (údaje o tom, kolik kusů daného produktu bylo již přesunuto a kolik jich zbývá přesunout). Tato komplikace je dána zejména faktem, že jsem se rozhodl na frontendové části použít pro přesuny produktů stejnou komponentu, jako u běžné úlohy vyskladnění – backendová část se tak musí chovat stejně, jako by se jednalo o běžnou úlohu vyskladnění. Je to pomyslná daň za skutečnost, že bude pak možné u klientské části aplikace pouze nahradit URL endpointů a veškeré přesuny produktů v rámci nadvyskladnění by měly fungovat hladce.

3.2.2 Implementace frontendové části

Frontendová část aplikace, vyvinuta Ing Oldřichem Malcem v rámci jeho diplomové práce [1], je napsaná v jazyce Javascript s použitím moderního frontendového frameworku Vue.js. [24].

Vzhledem k tomu, že Vue.js funguje na principu vykreslování celkového UI za pomoci skládání jednotlivých dílčích komponent, nebyl v tomto případě problém využít pro implementaci úlohy nadvyskladnění pro některé dílčí části komponenty patřící klasické úloze vyskladnění. Tato skutečnost mi usnadnila velké množství práce – na řadě míst, jako je například manipulace s jednotlivými kusy produktů, stačilo pouze vyměnit třídu, která měla na starost komunikaci s API, za třídu, jež namísto endpointů úlohy vyskladnění komunikovala s endpointy našeho nového nadvyskladnění.

Výpis kódu 3.2: Výměna klienta API

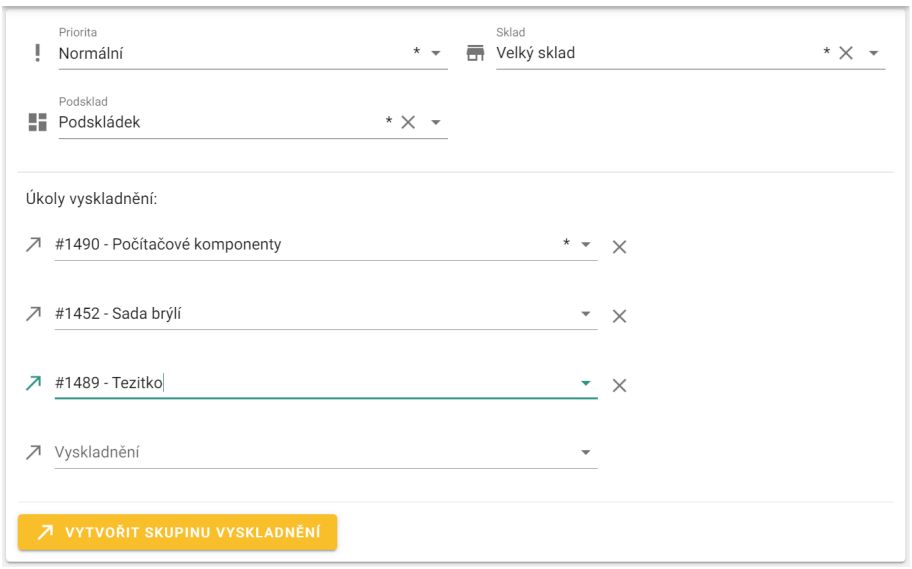
```
API: function () {
    return this.isTaskSet ? TaskStockPickingSetAPI : TaskStockPickingAPI;
}
```

Mimo to bylo však stále nutné implementovat části, které se oproti původní úloze vyskladnění liší. Chceme jistě přidat například obrazovku, kde by si

uživatel mohl tuto úlohu manuálně vytvořit z již existujících vyskladnění. Dále pak v detailu samotné úlohy chceme po vzoru návrhu zachovat informace o tom, z jakých vyskladnění se úloha skládá.

Při implementaci komponenty, která umožní uživateli samotnou úlohu manuálně vytvořit, bylo velice důležité vhodně ošetřit, aby se uživateli nepodařilo vybrat úlohy, které by neměly v rámci jedné úlohy nadvyskladnění koexistovat – například jsou nežádoucí úlohy s rozdílnými cílovými umístěními, rozdílnými prioritami nebo podsklady, do kterých přísluší. Tyto vlastnosti se sice řádně validují v backendové části, i tak by bylo ale nepříjemné umožnit uživateli takovou kombinaci úloh poskládat a pak mu jen zobrazit chybovou hlášku. Výsledek mého snažení je k vidění na obrázku 3.3. *Na obrázku je mimo jiné patrné, že jsem místo pracovního názvu „nadvyskladnění“ vymyslel pro tuto úlohu nový název – „skupina vyskladnění“, a tak se pokusím ve zbylém textu používat spíše toto druhé, dle mého názoru více přirozené, pojmenování.*

Obrázek 3.3: Tvorba skupiny vyskladnění



Co se týče karty s podrobnostmi o samotné úloze, mimo již existující komponentu generického detailu úlohy, jsem do obrazovky přidal seznam úloh vyskladnění zahrnutých ve skupině vyskladnění. Dále zde pak přibyla možnost smazat celou skupinu a tím jednotlivé vyskladnění rozpustit (toto se od prvotního návrhu liší, ale vzhledem k tomu, že nepředpokládám, že by uživatelé chtěli měnit obsah skupiny nějak výrazně často, považuji za v tuto chvíli přímočařejší řešení smazání a pak opětovné vytvoření celé úlohy).

3.3 Realizace automatického zpracovávání objednávek

3.3.1 Konfigurace podskladu

Jako první část realizace celého návrhu, popsaného v podkapitole 2.2.4, jsem se rozhodl zvolit vytvoření konfigurace automatického zpracovávání pro každý podsklad. Toto rozhodnutí je poměrně logické z toho důvodu, že při implementaci samotné služby, která bude mít automatické zpracovávání na starosti, budeme tyto údaje potřebovat a budou při zpracovávání podstatné.

V prvním kroku jsem se tedy rozhodl přidat veškeré mnou navržené konfigurační atributy přímo do entitního typu podskladu – bez pochyby se toto nastavení váže vždy právě k němu. V tabulce 3.1 jsem zrekapituloval, o jaké atributy jde a za jakým účelem jsem se je rozhodl přidat.

Tabulka 3.1: Nové atributy entity podskladu

Atribut	**Účel při zpracovávání**
min_orders_per_automatic_set	Min. počet objednávek ve skupině
max_orders_per_automatic_set	Max. počet objednávek ve skupině
max_automatic_sets	Max počet připravených skupin
automatic_order_processing	Automatické spouštění zpracování

3.3.2 Ukládání informací o používaných strategiích

Mezi další informace, které bude služba pro zpracovávání objednávek při svém běhu potřebovat, jsou strategie, které má podsklad zaregistrované a to včetně jejich priorit, aby bylo zřejmé, v jakém pořadí se mají na objednávky aplikovat.

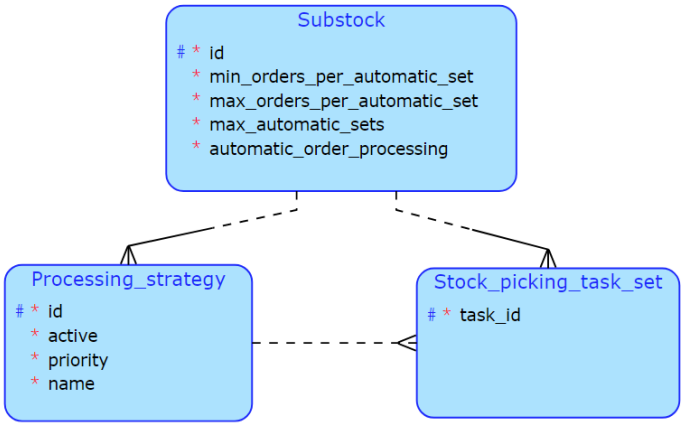
Změny v databázi jsem proto provedl po vzoru ER modelu 3.4, který popisuje vazby mezi jednotlivými sklady a jejich strategiemi. Každá strategie má vyplněnou prioritu, na základně které ji služba později zařadí do fronty strategií, které se mají na objednávky aplikovat. Mimo to má strategie ještě povinný atribut názvu, podle kterého by měla služba najít příslušnou implementaci funkcionality této strategie (toto chování budu popisovat detailněji později v práci). Strategie mají rovněž vazbu na veškeré skupiny vyskladnění, které byly na pomocí nich sestaveny.

3.3.3 Realizace API endpointů

Pomocí endpointů by na základě návrhů mělo být možné z frontendové části aplikace:

- Získat konfiguraci a strategie, které využívá konkrétní podsklad
- Vyvolat zpracování pro konkrétní podsklad

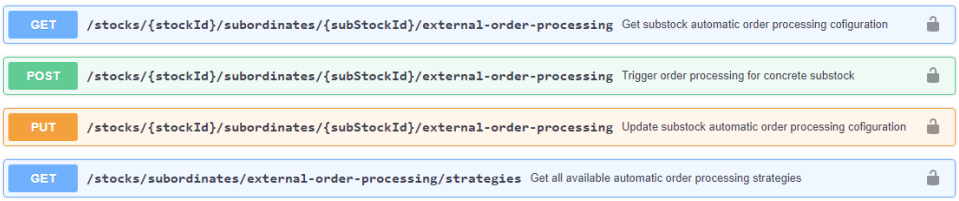
Obrázek 3.4: ER model strategií a konfigurace podskladu



- Nastavit konfiguraci a strategie pro konkrétní podsklad
- Získat veškeré dostupné strategie (z kterých si může podsklad vybrat)

Výsledná podoba dokumentace endpointů je dostupná na obrázku 3.5, jedná se (téměř) o typické CRUD operace [25] pro cestu *. . .subStockId/external-order-processing* doplněné o jeden endpoint navíc, který bude využíván pro obdržení veškerých dostupných strategií (není potřeba při žádosti žádný z identifikátorů skladu či podskladu).

Obrázek 3.5: Endpointy pro konfiguraci zpracovávání



3.3.4 Implementace strategií

Při implementaci funkcionalit strategií, které mají z principu na základě své implementované logiky vybírat objednávky, ze kterých by bylo možné vytvořit nové skupiny a případně i vybrat existující skupiny, do kterých by se zvolené objednávky daly přidat, jsem se snažil co nejvíce dbát na potenciální budoucí rozšíření o nové implementace. Proto jsem se rozhodl pro objektově orientovaný přístup, kdy by pro přidání nové strategie mělo programátorovi stačit pouze vytvořit novou třídu, která dědí z abstraktní třídy strategie a implementovat veškeré veřejné abstraktní metody z tohoto rodiče. Dále by pak mělo

zbývat tuto strategii pouze jednoduše zaregistrovat ve službě, která se stará o automatické zpracovávání a strategie by v tu chvíli měla být dostupná ke konfiguraci pro všechny podsklady – na základě názvu této implementace by se pak automaticky vytvářely korespondující databázové záznamy s příslušnými prioritami u konkrétních podskladů.

Abstraktní třídu **Strategy**, která poskytuje rozhraní pro všechny implementace strategií, je možné si prohlédnout na ukázce 3.3.

Výpis kódu 3.3: Abstraktní třída Strategy

```
abstract class Strategy
{
    /**
     * Attempt to choose orders using implemented strategy.
     * @param QueryBuilder $externalOrderQueryBuilder
     * @param int $numberOfAttempt
     * @return ExternalOrder[]
     */
    public abstract function chooseOrdersForNewSet(QueryBuilder
        $externalOrderQueryBuilder, int $numberOfAttempt): array;

    /**
     * Attempt to append orders to one of existing sets using implemented strategy.
     * @param QueryBuilder $externalOrderQueryBuilder
     * @param StockPickingSet[] $sets
     * @return Mixed[] orders that have been appended on first and selected set on the
         second position
     */
    public abstract function appendOrdersToExistingSet(QueryBuilder
        $externalOrderQueryBuilder, array $sets): array;

    /**
     * Get a strategy name.
     * @return string
     */
    public abstract function getName(): string;


    /**
     * Filter sets that have not been created by implemented strategy.
     * @param StockPickingSet[] $sets
     * @return StockPickingSet[]
     */
    protected function filterSets(array $sets): array
    {
        $filtered = array_filter($sets, function ($v): bool {
            return $v->getUsedStrategy()->getName() === $this->getName();
        }, ARRAY_FILTER_USE_BOTH);
        return array_values($filtered);
    }

    /**
     * Indicates whether the strategy returns grouped results from most favorable to
         least.
     * If false, strategy always returns all the possible orders united.
     * @return bool
     */
    public abstract function givesPartialResult(): bool;
}
```

Z ukázky je možné si všimnout všech abstraktních metod, které jsou pro přidání nové strategie nutné implementovat.

V první řadě například metoda **getName**, která je využívána pro identifikaci strategie – mimo jiné pro párování implementace strategie se záznamy v databázi spojených s konkrétním podskladem.

Dále zde můžeme nalézt metody **chooseOrdersForNewSet** a **appendOrdersToExistingSet**, které slouží pro samotný výběr objednávek k seskupení, nebo jejich přidání do již existujících skupin, které byly dříve vytvořené pomocí totožné strategie.

V poslední řadě je zde signatura metody **givesPartialResult** – návratové hodnota této metody značí, že při vybíraní vhodných objednávek jsou tyto objednávky rozděleny na disjunktní množiny, které není možné dohromady míchat (k tomu dochází například u strategie rozdělující objednávky do skupin na základě identických produktů). Pokud má některá z implementací tento příznak, je možné výběr výsledné množiny ovlivnit pomocí parametru **numberOfAttempt** u metody **chooseOrdersForNewSet**, který jí pro výběr objednávek řekne, kolikátou množinu má zvolit (tyto množiny jsou typicky seřazeny sestupně dle počtu objednávek, které obsahují – od nejvýhodnějších).

3.3.5 Služba zpracovávání objednávek

Po implementaci všech závislostí zbývalo realizovat samotnou službu, která měla za úkol seskupovat objendávky jednotlivých podskladů pomocí konfigurace a strategií, které mají dotyčné podsklady nastavené.

Vzniklá služba **ExternalOrderProcessingService** prochází při spuštění veškeré podsklady. U každého podskladu si služba zažádá o jeho konfiguraci a seřazené strategie, poté, s ohledem na tuto konfiguraci, na objednávky aplikuje jednotlivé strategie. Tento průběh pro jednotlivé strategie byl již dříve zachycen v návrhu 2.15.

3.3.6 Grafické rozhraní pro konfiguraci podskladu

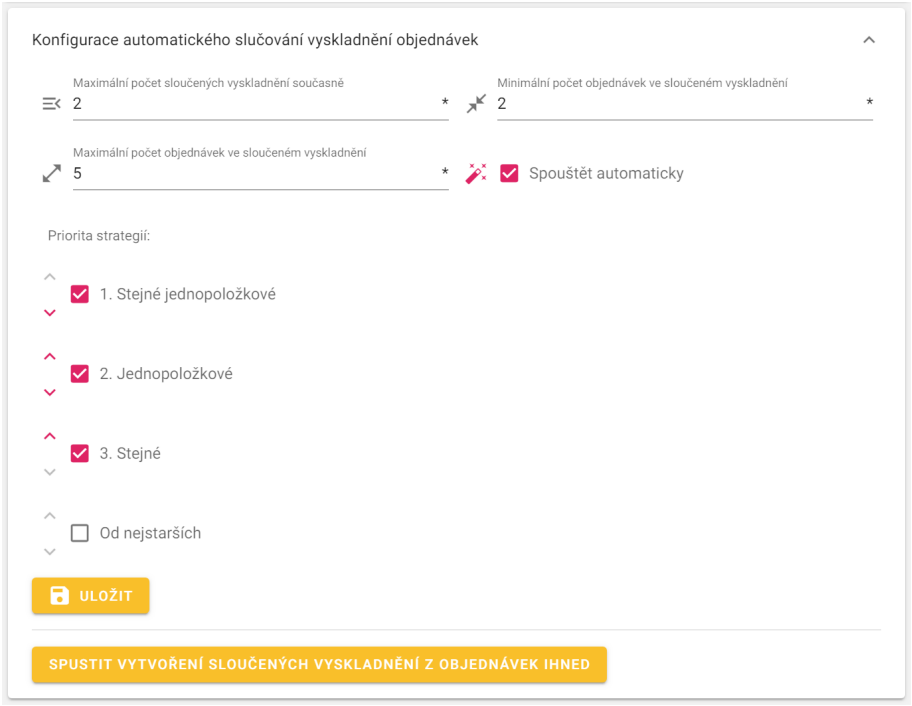
Během dokončení implementací v backendové části aplikace jsem z důvodu časové úspory poprosil Ing. Oldřicha Malce [13] o vypracování konfigurační karty podskladu ve frontendové části aplikace na základě mnou poskytnutých endpointů (3.6). Bylo tak možné ihned po dokončení implementace vyzkoušet, jak bude fungovat nastavování konfigurace přímo v UI z pohledu uživatele.

3.4 Informace objednávek pro proces balení

Posledním z realizovaných návrhů je návrh z kapitoly 2.3, ve kterém jsem naznačil, jak by měla vypadat strukturální úprava aplikace pro podporu ukládání veškerých informací, které budou později využívány při tištění štítku v rámci procesu balení.

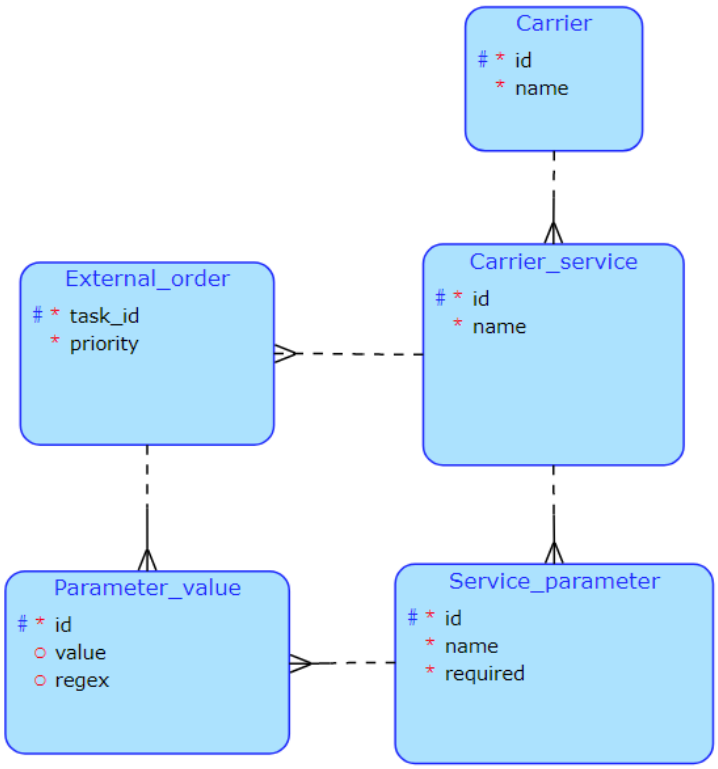
Mým cílem bylo tedy upravit databázový model takovým způsobem, aby podporoval ukládání veškerých informací z návrhu. Při těchto úpravách jsem však narazil na jeden menší nedostatek původního návrhu a tím byla entita parametrů služby a s ní související vazba se samotnou objednávkou. Správně jsem v návrhu sice identifikoval kardinalitu vazby jako N:N, ovšem nebral

Obrázek 3.6: UI konfigurace zpracovávání od Ing. Malce



jsem v potaz fakt, že některé parametry služby dopravců mohou v reálném světě vyžadovat hodnoty pro tyto parametry. Dále pak jednotlivé parametry dané služby nemusí být vždy nutně vyžadované – bylo potřeba rozlišit povinné parametry od volitelných. Finální podobu ER modelu je možné zhlédnout na obrázku 3.7.

Obrázek 3.7: ER model informací o dopravě objednávky



Kapitola 4

Testování

Jak jsem již uvedl v úvodu práce, původním cílem bylo otestovat mnou realizované návrhy za pomoci nějakého reálného uživatele skladového systému. Naneštěstí nebylo časové období, kdy jsem návrhy implementoval a chtěl je testovat, zcela vhodné ke kontaktnímu uživatelskému testování přímo ve skladu, protože byla v České republice zavedena uzávěra veškerých okresů z důvodu prevence proti šíření COVID-19.

Později jsem si však uvědomil, že návrhy, které jsem v rámci této práce realizoval, z principu ani tak moc uživatelské testování nevyžadují, protože se jedná zejména o funkcionalitu, která je implementována v rámci backendové části aplikace a její správné výstupy se dají mnohem lépe otestovat přímo softwarově. Co se týče frontendové části, tak je zde výhodou skutečnost, že nové komponenty jsou vetšinou postaveny na již existujících komponentách, u kterých bylo UX již hojně otestováno v reálném provozu. Z těchto důvodů jsem se rozhodl otestovat mé výsledky alespoň na softwarové úrovni za pomoci automatizovatelných API testů.

4.1 Metoda testování

Jako nejvhodnější metodu testování jsem, vzhledem k podobě implementovaných funkcionalit, zvolil integrační API testování. Dám se tedy do sestavení sad testů, kde v rámci každého testu zavolám několik endpointů, které by měly způsobit změnu stavu v systémové databázi a poté, pomocí dalších endpointů, přečtu z backendové části tento ovlivněný stav, vůči kterému aplikuji testy, které budou testovat správnou formu a hodnotu odpovědí.

4.2 Výběr nástroje pro testování

Již při vývoji endpointů během realizace backendové části aplikace jsem pro ověřování správnosti implementace používal nástroj Postman. [26] Jedná se o

nástroj, který ve zkratce svou základní funkcionalitou umožňuje tvoru vlastních HTTP požadavků a detailní náhled do odpovědí na tyto požadavky. Jedná se tedy o výborného pomocníka při vývoji jakýchkoli aplikací, které komunikují pomocí tohoto protokolu.

Nedávno jsem však zjistil, že tento HTTP klient umožňuje mimo tuto základní funkcionalitu i tvorbu plnohodnotných testů, které kromě nahlížení do obsahu požadavků umožňují i kontrolu podoby odpovědí – kromě stavového kódu dokonce i kompletní podobu těla odpovědi. Jejich hromadné automatizované spouštění v sekvenci za sebou je možné dosáhnout pomocí funkcionality této aplikace, která se nazývá *Collection Runner*. Ten umožňuje spustit celou kolekci požadavků včetně jejich příslušných testů. Rovněž je možné mezi jednotlivými požadavky předávat různá data a tím je de facto možná mimo jiné i realizace jejich cyklických spouštění – jeli potřebné například uskutečnit více požadavků pro jednu URL, je jednoduše možné vytvořit pouze jeden požadavek, který se poté spouští ve smyčce, kolikrát si jen uživatel zvolí. [27]

4.3 Rozvržení testů

Při návrhu vhodné podoby testů jsem se rozhodl zaměřit zejména na automatické zpracovávání objednávek pomocí definovaných strategií. Jedná se o realizovanou část, která je ze všech implementovaných funkcionalit nejvíce složitá a tedy i náchylná k zanesení chyb. Dbal jsem proto důraz na to, aby se otestovaly veškeré součásti tohoto modulu. Identifikované části, které jsem uznal za potřebné separátně otestovat, jsem sepsal do následujícího seznamu:

Konfigurace podskladu: Sada testů, která má za cíl otestovat správné dodržování konfigurací spjatých s podsklady. V testech bylo testováno například dodržování hranice minimálních a maximálních počtů objednávek ve skupinách vyskladnění nebo například celkový maximální počet skupin ve skladu v jeden čas.

Jednotlivé strategie: Testy pro každou, doposud implementovanou, strategii rozdělování objednávek. V každé sadě testů jsem se zaměřil na konkrétní strategii a testoval správnost jejího seskupování.

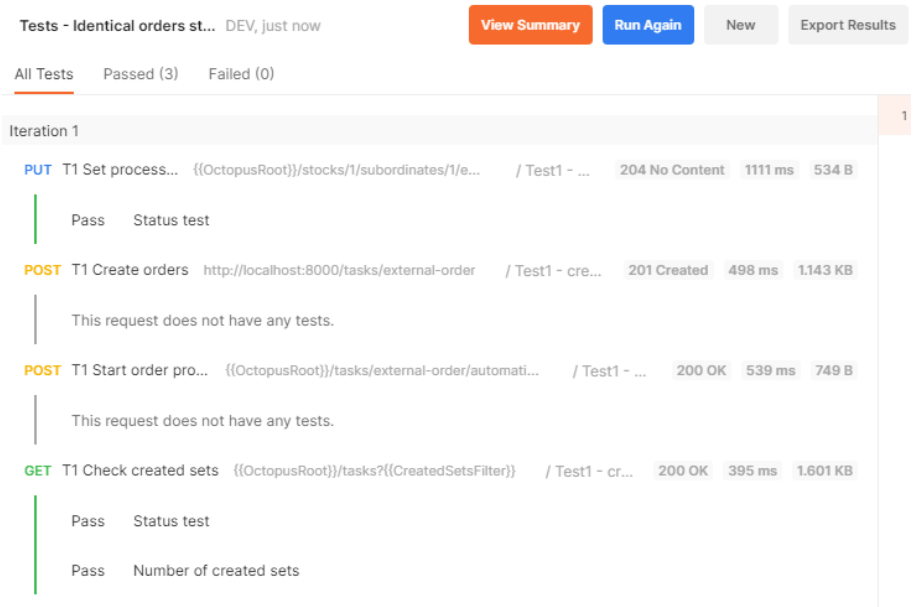
Všechny strategie: Sada testů, ve které jsem se zaměřil na zpracovávání objednávek pomocí všech strategií, které byly doposud implementované. Pro tento typ testování jsem se mimo jiné rozhodl i proto, že separátní testování jednotlivých strategií nemůže přímo simulovat reálný provoz, kdy bude aktivní větší počet strategií, které si budou navzájem jednotlivé objednávky „vykrádat“.

Ostatní: Testy, kde jsem se zaměřil na ostatní aspekty, jako je například správné rozdělování objednávek do skupin na základě jejich priorit.

Každá z těchto sad testů byla tedy realizovaná jako kolekce, kterou pak stačilo spustit. Příklad části výsledků jednoho takového běhu je možné zhlédnout na obrázku 4.1.

Po spuštění veškerých testů jsem musel v reakci na jejich výsledky opravit několik chyb – jedna z nich byla například spojena s problémem přidávání objednávek do existujících skupin (v případě, že byly sestaveny pomocí strategie identických objednávek).

Obrázek 4.1: Část sady testů pro strategii identických objednávek



4.4 Měření rychlosti zpracovávání

Při realizaci celého modulu, který má na starosti automatické zpracovávání objednávek, jsem se nemohl zbavit obav z výsledného výkonu, protože v se v rámci tohoto procesu generuje spousta dotazů na databázi a občas se s daty zachází i ve formě polí uvnitř jednotlivých implementací strategií.

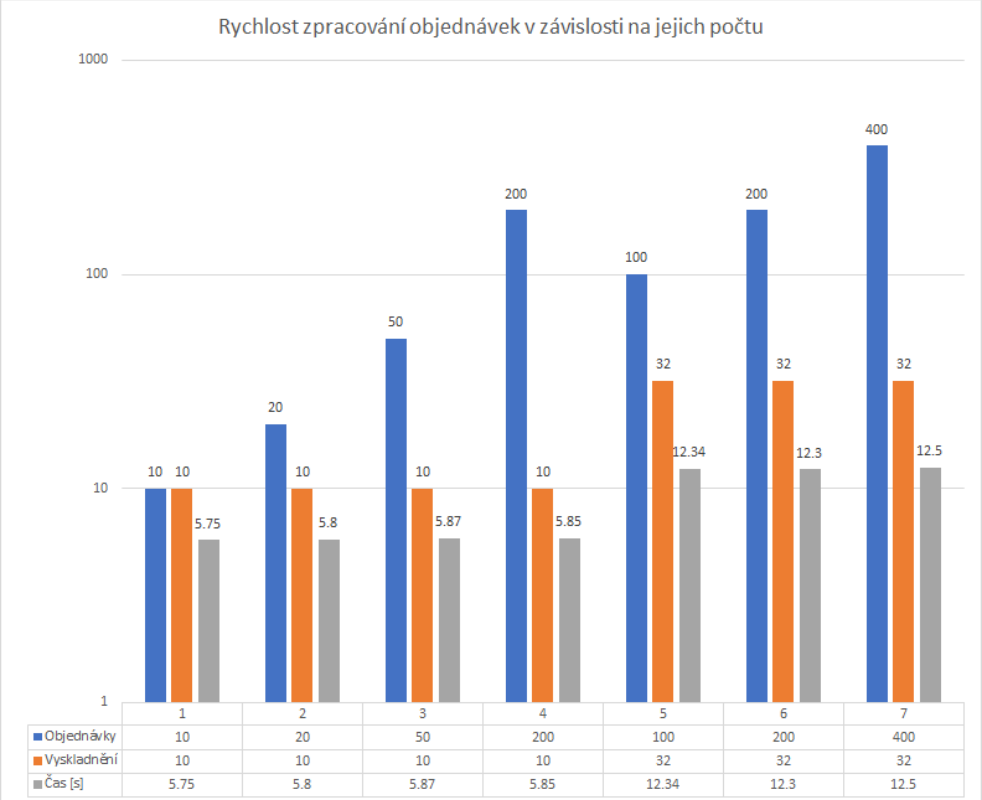
Proto jsem se na závěr testování rozhodl zaměřit na samotnou rychlost zpracovávání a připravil několik kolekcí požadavků, které měly za úkol vytvořit vždy různě velký počet objednávek. Typově jsem volil pro vytvoření objednávky takové produkty, aby bylo zpracovávání objednávek v rámci jednotlivých strategií rovnoměrné a tím jsem potenciálně odhalil i případ, kdy by byla neoptimalizovaná pouze jedna ze strategií.

Následující graf 4.2 popisuje naměřené časové výsledky vůči počtu objednávek, které čekaly před spuštěním na zpracování a zároveň i počtu vysklad-

nění, které reálně v rámci zpracování z objednávek vzniky (počet zpracovaných objednávek).

Z výsledných dat je patrné, že strávený čas je závislý zejména na počtu objednávek, které byly rutinou skutečně zpracovány, nikoliv na počtu objednávek, které čekají na zpracování a ze kterých si automat vybírá. Toto zjištění je velice pozitivní, protože by se díky němu nemělo stát, že by po nahromadění velkého množství objednávek došlo k extrémně dlouhému zpracovávání, vzhledem k tomu, že doba trvání bude úměrná pouze k počtu výsledně vytvořených vyskladnění. Vztah mezi počtem zpracovaných objednávek a trváním se navíc z grafu zdá být lineární.

Obrázek 4.2: Měření rychlosti zpracovávání objednávek



Závěr

Cílem této práce bylo navrhnout a realizovat rozšíření, která by přinesla do stávajícího skladového systému Atlantis nové funkcionality, vyřešila nejvíce prioritní nedostatky a tím maximálně ušetřila uživatelům práci s aplikací a umožnila jim využívat věcí, které do této chvíle v systému chyběly. Tyto vylepšení se měly týkat zejména procesu zpracovávání objednávek.

V práci jsem nejprve důkladně zanalyzoval současný stav a identifikoval místa, kde jsem nalezl prostor ke zlepšení. Pro tyto nedostatky jsem vypracoval celou řadu návrhů, v některých jsem bral v potaz mimo jiné i analýzu a návrhy, které byly vypracovány již přede mnou mým předchůdcem. U každého z návrhů jsem se snažil co nejvíce přihlížet k aktuální podobě systému, aby byla případná realizace možná a dobře zapadla do celkové domény. Spolu se zadavatelem jsme pak určili ty návrhy, které mělo v současné chvíli největší smysl realizovat. Vybrané návrhy byly poté implementovány a systém byl o ně rozšířen. V poslední fázi této práce jsem mnou realizovaná rozšíření patřičně otestoval.

Výsledkem mého snažení je řada zmíněných návrhů pro různé problematiky, kterým jsem se věnoval, ale zejména je jím rozšíření, které se mi podařilo dotáhnout do konce. Jedná se o řešení, které do systému přineslo zcela nový proces zpracovávání objednávek. Toto rozšíření z velké části automatizuje úkony, které byly do této chvíle potřebné dělat manuálně a činní tak aplikaci vhodnou i pro větší spektrum zákazníků. Současně je vysoce rozšiřitelné a upravitelné, takže si jej může každý zákazník nastavit podle své libosti. Rozšíření se bohužel nepodařilo otestovat reálným uživatelem skladu, což na jednu stranu ovšem nemusí být takový problém, vzhledem k tomu, že většina mnou realizované práce není patrná skrze uživatelské rozhraní a její korektní výsledky jsou otestovatelné vhodnými softwarovými testy, které jsem provedl.

Závěrem bych uvedl, že se v této práci vyskytlo plno návrhů, pro které nezbyl prostor k jejich kompletní realizaci, a proto vidím možnost navazování na tuto práci hlavně v jejich případném dalším rozvinutí a realizaci.

Literatura

[1] MALEC, Oldřich. *Frontend skladového systému*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020

[2] Kovář, Pavel. *Backend skladového systému*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019

[3] TALÁR, Bc. Denis. *Optimalizace nejčastějších procesů ve skladovém systému*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020

[4] HUNKA, Ing. Jiří. Zadavatel, vedoucí bakalářské práce [ústní sdělení]. 2021.

[5] ČOI. *Odstoupení od smlouvy do 14 dnů u zboží* [online]. 2017 [cit. 2021-4-18]. Dostupné z: https://www.coi.cz/faq/4-odstoupeni-od-smlouvy-do-14-dnu-u-zbozi-2/

[6] SCALO, Dan. *Methods for Improving Your Picking and Packing Process* [online]. 2019 [cit. 2021-4-25]. Dostupné z: https://www.ecommerce-nation.com/methods-improving-pickingpacking/

[7] *The Ultimate Guide to Pick and Pack Methods!* [online]. 2020 [cit. 2020-4-25]. Dostupné z: https://supplychaingamechanger.com/the-ultimate-guide-topick-and-pack-methods/

[8] Česká pošta. *Balíky - zásilky ČR* [online]. 2021 [cit. 2021-4-18]. Dostupné z: https://www.ceskaposta.cz/sluzby/baliky/cr

[9] SPARX SYSTEMS. *Enterprise Architect* [software]. 2021-4-12 [cit. 2021-4-18]. Dostupné z: https://sparxsystems.com/products/ea/downloads.html

[10] JAYE, Hannah. *What Exactly Is Wireframing? A Comprehensive Guide* [online]. 2021 [cit. 2021-4-18]. Dostupné z: https://careerfoundry.com/en/blog/ux-design/what-is-a-wireframe-guide/

[11] BALSAMIQ. *Balsamiq* [software]. 2021-4-7 [cit. 2021-4-18]. Dostupné z: https://balsamiq.com/wireframes/desktop/

[12] CHURSIN, Oleg. *A Brief Introduction to Domain Modeling* [online]. 2017 [cit. 2021-4-18]. Dostupné z: https://olegchursin.medium.com/a-brief-introduction-to-domain-modeling-862a30b38353

[13] MALEC, Ing. Oldřich. Tvůrce frontendové části skladového systému Atlantis [ústní sdělení]. 2021.

[14] HUMLÍČEK, Matěj, Vedoucí týmu pro optimalizaci pohybů na skladě v rámci BI-SP1 [ústní sdělení]. 2021-3-22.

[15] *Batching strategies* [online]. [cit. 2021-4-25]. Dostupné z: https://www.erim.eur.nl/material-handling-forum/research-education/tools/calc-order-picking-time/what-to-do/batching-strategies/

[16] GLYNN, Fergal. *What is batch picking?* [online]. 2021 [cit. 2021-4-25]. Dostupné z: https://6river.com/what-is-batch-picking/

[17] *PHP: Hypertext Preprocessor* [online]. 2021 [cit. 2021-4-28]. Dostupné z: https://www.php.net

[18] *What is Symfony* [online]. 2021 [cit. 2021-4-28]. Dostupné z: https://symfony.com/what-is-symfony

[19] *PostgreSQL: About* [online]. 2021 [cit. 2021-4-28]. Dostupné z: https://www.postgresql.org/about

[20] *doctrine/orm: Doctrine Object Relational Mapper (ORM)* [online]. 2021 [cit. 2021-4-28]. Dostupné z: https://github.com/doctrine/orm

[21] *API Documentation and Design Tools for Teams* [online]. 2021 [cit. 2021-5-1]. Dostupné z: https://swagger.io/

[22] *ER model* [online]. 2018 [cit. 2021-5-1]. Dostupné z: https://www.javatpoint.com/dbms-er-model-concept

[23] *Transactions and Concurrency* [online]. 2021 [cit. 2021-5-1]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.8/reference/transactions-and-concurrency.html

[24] YOU, Evan. *Vue - The Progressive JavaScript Framework* [online]. 2021 [cit. 2021-5-1]. Dostupné z: https://vuejs.org/.

[25] *What is CRUD?* [online]. 2021 [cit. 2021-5-2]. Dostupné z: https://www.codecademy.com/articles/what-is-crud

[26] POSTMAN. *Postman* [software]. 2021 [cit. 2021-5-6]. Dostupné z: https://www.postman.com/downloads/

[27] *Using the Collection Runner* [online]. 2021 [cit. 2021-5-6]. Dostupné z: https://learning.postman.com/docs/running-collections/intro-to-collection-runs/

Příloha A

Seznam použitých zkratek

API Application Programming Interface

ČVUT České vysoké učení technické

FIFO First in, first out

FIT Fakulta informačních technologií

GUI Graphical user interface

HTTP Hypertext Transfer Protocol

PHP PHP: Hypertext Preprocessor

REST Representational State Transfer

UI User interface

UX User experience

YAML YAML Ain't Markup Language

Příloha B

Seznam použitých pojmů

Sklad: Budova, ve které se skladují kusy produktů.

Podsklad: Virtuální část skladu určující vlastníka daného zboží, veškeré úkony s kusy produktů na skladě se váží právě k podskladu.

Umístění: Fyzické umístění ve skladu, na kterém se skladují kusy produktů identifikované svými instancemi.

Produkt: Skladová položka.

Instance: Identifikovatelná sada kusů produktu. K instanci se přiřazují čárové kódy. Nejedná se o konkrétní fyzický kus nějakého produktu, instance identifikuje 0..N kusů.

Vedoucí: Zadává úkoly, kontroluje jejich splnění.

Skladník: Provádí fyzickou realizaci úkolů.

Načitat/skenovat/napípat: Činnost, při které se naskenováním čárového kódu instance produktu dostanou informace do digitální podoby. Často se pro tuto operaci využívá mobilní telefon se zabudovanou čtečkou od výrobce ZEBRA.

Příloha C

Seznam procesů

Vyskladnění : Proces přesunutí konkrétních kusů produktu z umístění za účelem jejich expedice nebo jiné manipulace. Kusy jsou typicky přesunuty na jiné umístění, kde jsou baleny a expedovány nebo v zjednodušeném režimu, bez zvoleného cílového umístění, rovnou po dokončení úlohy opouští evidenci.

Balení: Balení položek objednávky z produktů, které byly vyskladněny na umístění, které je určené k balení. Položky jsou vloženy do krabic, na které je následně nalepen štítek s informacemi o dopravě.

Expedice: Proces, při kterém zabalené kusy opouští sklad za účelem odeslání odběrateli. Většinou je předcházen vyskladněním a balením.

Příloha D

Katalog požadavků sestavený v rámci analýzy

Funkční požadavky:

F001 – Vytvoření konceptu poptávek jako takových.

F001a – Namísto pomocných funkcí, jako je vyskladnění, představit pro rezervaci poptávaných produktů novou funkcionalitu, která bude pro tyto účely přímo určená.

F002 – Systém by měl umožňovat tyto poptávky různými způsoby upravovat. Měl by být definovaný jasný postup, který se bude při změně poptávky dodržovat v závislosti na tom, ve které fázi se produkty zrovna nachází.

F002a – Poptávky by měly být možné zcela zrušit nehledě na to, v jakém stavu se zrovna poptávané předměty nachází (jsou-li právě vyskladňovány, baleny nebo připraveny k odeslání) – dokud produkty neopustily sklad, poptávka by měla jít stornovat.

F002b – To samé platí pro změnu (přidání / odebrání) poptávaných předmětů, v závislosti na fázi, ve které se produkty nachází, by mělo být možné nějakým efektivním postupem produkty v poptávce změnit.

F003 – Systém by měl mít uložené veškeré informace z eshopu, které budou potřeba pro odeslání produktů, tyto informace pak budou patřit ke konkrétní poptávce.

F003a – Informace pro odeslání by měly jít změnit, pokud to bude možné (typicky dokud produkty nejsou zabaleny a připraveny k odeslání)

F004 – Kromě informací z eshopu k odeslání by měl skladový systém disponovat i informacemi o odběrateli.

F004a – Každá poptávka by měla mít vazbu na odběratele.

F004b – Odběratelé by neměli být v databázi systému pokud možno duplicitní a systém by měl sám rozpoznat při zakládání nového odběratele podle poskytnutých údajů, že tohoto odběratele již má v databázi.

F005 – Poptávané produkty by měly být i před začátkem vyskladňování rezervované, aby jeden konkrétní kus produktu nemohl být zahrnut ve vícero poptávkách.

F005a – Systém by měl mít daný postup k uspokojování poptávek (která poptávka má přednost při rezervování produktu potom, co se daný produkt naskladní).

F005b – Systém by měl umožnit manuální upravování produktů v poptávkách administrátorem (přesouvání rezervovaných produktů z jedné poptávky do druhé).

F006 – Systém by měl podporovat i režim, ve kterém by poptávky naopak nerezervovaly produkty a jen by se po manuálním pokynu přidaly k vyskladnění, pokud by byly všechny položky poptávky skladem.

F007 – Systém by měl umožnit částečné uspokojování poptávek

F007a – Mělo by být dohledatelné, z jaké poptávky daná poptávka vznikla, v tomto případě by poptávka nemusela mít vazu na odběratele a ostatní informace, stačila by jen vazba na rodičovskou poptávku, která tyto informace již má.

F008 – Pokud produkt patří do nějaké poptávky, mělo by být možné v jakékoliv jeho fázi (nehledě na to, zda je v regálu, vyskladňován, vyskladněn či expedován) dohledat, do které poptávky patří.

F008a – Tato vlastnost bude později využita u balení a expedice produktů.

F009 – Systém by měl uchovávat historii veškerých již uspokojených poptávek Na základě tohoto požadavku by mělo být možné dohledat, kdy přesně se které produkty expedovaly a komu byla objednávka odeslána s veškerými dalšími doplňujícími informacemi z eshopu.

F010 – V přehledu naskladněných produktů by mělo být zjistitelné, které z produktů jsou rezervované existujícími poptávkami. V nerezervujícím režimu by zde mělo být alespoň vidět, kolik kusů „ubyde“, uspokojíme-li všechny poptávky.

F011 – Skladový systém by měl při spravování (vytváření, upravování, mazání, informace o stavu) poptávek komunikovat s eshopem.

F011a – Je potřeba vymyslet, jakým způsobem by tato komunikace byla nejlépe realizována – zda se držet současné jednosměrné komunikace eshop -> sklad, kdy se eshop musí cyklicky dotazovat na stav poptávaného zboží, nebo realizovat způsob, kterým by sklad mohl zpětně kontaktovat eshop s aktualizovanými informacemi o stavu.

Nefunkční požadavky:

F001 – Implementace podpory poptávek by měla být provedena vhodným způsobem, aby mohl být později doimplementovaný proces balení objednávek, který bude využívat již existujících datových struktur, které budou do systému přidány v rámci tohoto řešení.

Příloha E

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `wireframes` ..... veškeré wireframes z kapitoly návrh
  - `Detail odběratele s pohyby produktů.pdf`
  - `Historie expedic.pdf`
  - `Nadvyskladnění - pohled skladníka.pdf`
  - `Nadvyskladnění - pohled vedoucího.pdf`
  - `Úloha poptávky s hromadným uspokojením.pdf`
  - `Vrácení produktů odběratelem.pdf`
- `ostatní`
  - `Scénář životního cyklu poptávky.docx`
- `testování` ..... exportované sady testů ve formátu json
  - `Tests - All strategies.postman collection.json`
  - `Tests - From oldest strategy.postman collection`
  - `Tests - Identical orders strategy.postman collection.json`
  - `Tests - Priority grouping.postman collection.json`
  - `Tests - Same single product strategy.postman collection.json`
  - `Tests - Single product strategy.postman collection.json`
  - `Tests - Substock configuration.postman collection.json`