FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Backend optimalizace skladových procesů systému Atlantis
Student:	Daniela Kováčová
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 tvorba backendu rozšíření skladového systému Atlantis, věnované optimalizaci náročnosti skladových operací. Práce bude realizována ve spolupráci s Vítem Urbanem, který v rámci své bakalářské práce realizuje Frontend stejného rozšíření.

Postupujte v těchto krocích:
1. Analyzujte současný stav skladového systému Atlantis se zaměřením na problematiku časové optimalizace procesů skladníků.
2. Na základě analýzy vhodně navrhněte nejpodstatnější rozšíření.
3. Návrh projděte a řádně konzultujte s vhodnými členy týmu skladového systému Atlantis.
4. Na základě návrhu implementujte výsledné rozšíření nebo jeho část.
5. Rozšíření řádně testujte, realizujte vhodné sady testů.
6. Shrňte dosažené výsledky, navrhněte možná budoucí vylepšení dle získaných zkušeností.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 19. ledna 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Backend optimalizace skladových procesů systému Atlantis

Daniela Kováčová

Katedra softwarového inženýrství
Vedúci práce: Ing. Jiří Hunka

12. mája 2022

Poďakovanie

V prvom rade by som chcela poďakovať vedúcemu mojej práce, pánovi Ing. Jiřímu Hunkovi, za jeho čas, dôkladnú spätnú väzbu a cenné rady.
Zároveň by som chcela poďakovať všetkým členom tímu z predmetov BI-SP1 a BI-SP2, vďaka ktorým vznikol dôležitý základ tejto práce.
V neposlednom rade patrí obrovské *ďakujem* môjmu priateľovi, rodine a kamarátom, ktorí ma podporovali nielen počas tvorby tejto práce, ale aj počas celého štúdia.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov. V súlade s ustanovením § 46 odst. 6 tohoto zákona týmto udeľujem bezvýhradné oprávnenie (licenciu) k užívaniu tejto mojej práce, a to vrátane všetkých počítačových programov ktoré sú jej súčasťou alebo prílohou a tiež všetkej ich dokumentácie (ďalej len „Dielo“), a to všetkým osobám, ktoré si prajú Dielo užívať.

Tieto osoby sú oprávnené Dielo používať akýmkoľvek spôsobom, ktorý neznižuje hodnotu Diela, a za akýmkoľvek účelom (vrátane komerčného využitia). Toto oprávnenie je časovo, územne a množstevne neobmedzené. Každá osoba, ktorá využije vyššie uvedenú licenciu, sa však zaväzuje priradiť každému dielu, ktoré vznikne (čo i len čiastočne) na základe Diela, úpravou Diela, spojením Diela s iným dielom, zaradením Diela do diela súborného či zpracovaním Diela (vrátane prekladu), licenciu aspoň vo vyššie uvedenom rozsahu a zároveň sa zaväzuje sprístupniť zdrojový kód takého diela aspoň zrovnateľným spôsobom a v zrovnateľnom rozsahu ako je zprístupnený zdrojový kód Diela.

V Prahe 12. mája 2022 . . . . . .

Odkaz na túto prácu

Kováčová, Daniela. *Backend optimalizace skladových procesů systému Atlantis*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Táto bakalárska práca je zameraná na kompletný proces analýzy, návrhu a implementáci optimalizácií už existujúceho skladového systému Atlantis od spoločnosti Jagu s.r.o., ktorý v rámci svojich záverečných prác navrhli a vyvinuli Ing. Oldřich Malec a Ing. Pavel Kovář. Práca sa venuje najmä zvýšeniu efektivity častých skladových procesov, umiestňovania skladových položiek a ich presunov. V súlade s vykonanou analýzou súčasného stavu a používateľ-ských požiadaviek sú vytvorené vhodné návrhy riešení, ktoré sú dôkladne prekonzultované a následne implementované v jazyku PHP tak, aby čo najhladšie zapadli do už existujúceho projektu. Výsledkom práce je detailný návrh a realizácia optimalizácií, ktoré zjednodušia a zrýchlia fungovanie skladového systému Atlantis a procesov sa v ňom odohrávajúcich.

Klíčová slova sklad, skladový systém, backend, optimalizácie procesov, PHP, Symfony

Abstract

This bachelor thesis is focused on the complete process of analysis, design, and implementation of optimizations of the existing warehouse system Atlantis from Jagu s.r.o., which was designed and developed by Ing. Oldřich Malec and Ing. Pavel Kovář. The thesis is mainly devoted to increasing the efficiency of frequent warehousing processes and the placement of products inside the warehouse. In accordance with the performed analysis of the system's current state and user requirements, suitable designs are created, thoroughly consulted, and subsequently implemented in the PHP language so that they fit into the already existing project as seamlessly as possible. The result of the thesis is a detailed design and implementation of optimizations that will simplify and speed up the operation of the Atlantis warehouse system and the processes taking place in it.

Keywords warehouse, warehouse system, backend, process optimizations, PHP, Symfony

Obsah

Zoznam obrázkov

Úvod

S príchodom počítačov a internetu sa zásadne zmenil spôsob našej existencie ako modernej spoločnosti, priemerný človek vo svojom živote len veľmi ťažko nájde niečo, čo by aspoň do nejakej miery nebolo ovplyvnené využívaním internetu. Jedným z mnoha podstatných pokrokov, ktoré sa dotkli našich každodenných rutín, je spôsob, ktorým v súčasnosti nakupujeme.

Aj keď podstata obchodovania zostala rovnaká, digitalizácia tohoto odvetvia priniesla množstvo zreteľných zmien. Z pohľadu zákazníka je viditeľné najmä znížené využitie kamenných obchodov, keďže veľká skupina ľudí si obľúbila pohodlnejšie a často aj finančne výhodnejšie nakupovanie cez internet. Táto nová forma nákupov však predávajúcim priniesla dávku predtým neexistujúcich povinností, od úpravy marketingových stratégií cez digitálnu inventúru produktov až po samotné dodanie objednávok koncovým zákazníkom.

V internetovej dobe sa však na každý problém takmer instantne nájde nejaké riešenie. V tomto prípade ním boli rôzne systémy a aplikácie na organizáciu procesov, ktoré prebiehajú na pozadí každého uskutočného obchodu.

Jedným z týchto riešení je aj skladový systém Atlantis od spoločnosti Jagu s.r.o., ktorý svojim používateľom ponúka nástroje na správu všetkých základných operácií nutných pri plnení bežných objednávok. Na vzniku tohoto systému sa v roku 2019 v rámci svojich diplomových prác na FIT ČVUT v Prahe podieľali Ing. Pavel Kovář a Ing. Oldřich Malec, ktorí Atlantis vytvorili na základe už existujúceho systému Sysel. [1, 2] Následne bol systém Atlantis podrobený ďalším úpravám v bakalárskych prácach Bc. Jana Cvrčka a Bc. Denisa Talára, ktoré vznikli taktiež na FIT ČVUT v Prahe, a zároveň sa na vývoji systému až do súčasnosti podieľa

tím spoločnosti Jagu s.r.o. [3, 4]

Online svet sa však aj naďalej veľmi rýchlo posúva vpred, čo znamená, že existujúce riešenia sa musia posúvať s ním, pokiaľ nechcú zaostať a stať sa ľahko nahraditeľnými. Z tohoto dôvodu je aj systém Atlantis neustále vylepšovaný, či už po stránke UI a UX dizajnu, alebo po stránke optimalizácie výpočetných algoritmov, prístupu k dátam a rýchlosti vykonávaných procesov.

Cieľ práce

V tejto práci som sa rozhodla venovať backendovému riešeniu optimalizácií najčastejších skladových procesov, ktoré okrem iného zahŕňa napríklad aj rozšírenie už existujúceho systému o údaje o rozmeroch a dostupnosti skladových umiestnení, stave skladových položiek, ich navrhovanom optimálnom umiestnení v rámci skladov a mnoho ďalších. Tento hlavný cieľ je možné rozdeliť na niekoľko menších na seba nadväzujúcich krokov.

Prvým krokom nutným k vypracovaniu práce bude zoznámenie sa so samotným systémom Atlantis a kompletná analýza súčasného stavu celého systému. Bude potrebné zistiť, aké základné procesy sa v systéme dejú, akým spôsobom sa pracuje s priestorom vo fyzických skladoch, a čo všetko zahŕňa manipulácia so skladovými položkami. Potrebné informácie získam formou rozhovorov s používateľmi systému, konzultáciami s vývojármi pracujúcimi na systéme, naštudovaním aktuálnej verzie projektu a záverečných prác týkajúcich sa pôvodnej formy skladového systému Atlantis.

Nasledujúcim krokom bude stanoviť konkrétne požiadavky používateľov systému a pomocou komplexnej analýzy určiť, či bude možné im v rámci optimalizácií vyhovieť, respektíve z nich vybrať a prioritizovať tie, ktorých splnenie by malo najväčšiu váhu pri optimalizácii daných procesov.

Ďalším krokom bude návrh riešenia vybraných požiadaviek v súlade so štýlom a formou implementácie stávajúcich procesov v systéme. Na týchto návrhoch sa okrem mňa budú podieľať aj členovia tímu z predmetov BI-SP1 a BI-SP2, menovite Vít Urban, Jiří Folprecht, Jana Karafiátová, Michal Kovář, Dominik Kyjevský, Jakub Meinlschmidt a Jakub Volák. Spoločne sa pokúsime čo najefektívnejšie vyriešiť dané požiadavky a zároveň tieto riešenia čo najmenej rušivo zapojiť do už existujúceho kódu. Tieto návrhy budú následne predložené vhodným členom vývojového tímu systému Atlantis, skonzultované a v prípade potreby upravené tak, aby dosahovali štandard porovnateľný so zvyšnou časťou implementácie.

Práca bude pokračovať realizačnou časťou, v rámci ktorej prebehne implementácia vopred analyzovaných a následne navrhnutých požiadaviek.

Riešenie bude v maximálnej možnej forme kopírovať schválený návrh, v prípade nutnosti odchýlenia sa od neho budú všetky potenciálne zmeny znovu konzultované s vhodnými členmi vývojového tímu systému.

V momente, keď bude hotová implementácia vybraných požiadaviek, príde na radu testovanie všetkých nových rozšírení systému. Na kontrolu správnosti návrhu a implementácie bude realizovaná vhodná sada testov a všetky nájdené chyby bude opravené.

Finálnou časťou celej práce bude zhrnutie dosiahnutých výsledkov, popis prínosu rozšírenia a návrh možných budúcich vylepšení na základe získaných skúseností.

Kapitola 1

Analýza

Skladový systém Atlantis je vo svojom súčasnom stave veľmi komplexným a prepracovaným nástrojom. O to náročnejšou úlohou je rozšíriť ho o potrebné funkcionality tak, aby tieto nové časti do systému bezchybne zapadli. Práve preto je jednou z najdôležitejších častí vývojového procesu analýza aktuálneho stavu softwaru, priamo nasledovaná dôkladnou analýzou všetkých požiadaviek. Táto kapitola práce sa bude venovať obom uvedeným častiam analýzy.

V prvej časti prebehne detailná analýza aktuálneho stavu softwaru, s váhou kladenou najmä na časti systému a procesy, ktorých sa budú týkať optimalizácie. Výsledkom bude zhrnutie funkcií, ktoré systém ponúka svojim používateľom, a zároveň popis procesov, ktoré sú nutné k jeho každodennému behu.

Druhá časť kapitoly sa bude venovať zberu nových zmenových požiadaviek, či už od členov vývojového tímu, alebo od aktívnych používateľov systému, a ich následnej analýze. Je nutné zo všetkých požiadaviek vybrať tie, ktorých implementácia je možná v realistickom časovom okne, a ktoré zároveň budú mať čo najväčší vplyv na lepšie rýchlejšie fungovanie systému. Porovnaním súčasného riešenia s navrhovanými zmenami sa určí priorita jednotlivých zmenových požiadaviek, a tie sa podrobnou špecifikáciou pripravia na následný návrh a samotnú implementáciu.

1.1 Analýza súčasného stavu

Skladový systém je implementovaný formou webovej aplikácie, čím svojim používateľom prináša výhody jednoduchého ovládania a zároveň ich neobmedzuje pri výbere zariadenia, z ktorého systém chcú používať. Aplikácia poskytuje možnosť ovládania a kontroly všetkých základných skladových

procesov, avšak množstvo z nich sa pri využití v skladoch s väčším objemom produktov stáva značne neefektívnymi.

Po získaní prístupu k projektu od zadávateľa sme v rámci tímu z predmetu BI-SP1 vykonali rešerš poskytnutých materiálov a na spoločných schôdzkach sme konzultovali najviac využívané časti systému a ich aktuálne problémy. Zároveň sme dostali príležitosť navštíviť jeden zo skladov, v ktorých je aktuálna verzia systému aktívne používaná, a zistiť priamo od jeho zamestnancov ich pripomienky a nápady k budúcim verziám skladu. Následne sme vyfiltrovali tie požiadavky, ktorých realizácia by mohla priniesť čo najviac pozitív a ušetriť čo najviac času pri bežnom fungovaní skladu.

V nasledujúcej podrobnej analýze sa budem sústrediť iba na vybrané časti systému, ktorým sme sa pri realizácií zmien venovali najviac.

1.1.1 Skladové položky a skladové umiestnenia

Systém z pohľadu implementácie rozlišuje skladové položky a skladové umiestnenia ako samostatné entity, ktoré majú svoje príslušné parametre, ako napríklad výrobcu, model alebo kód. U skladových položiek je možné pridať takzvané atribúty, ktoré obsahujú názov, hodnotu a popis. Toto riešenie je univerzálny spôsob uchovávania akejkoľvek číselnej hodnoty pre konkrétny produkt, aktuálne využívaný napríklad pre zaznamenanie rozmerov, v ich prípade to ale nie je optimálne riešenie, keďže ďalšia manipulácia s dátami uchovanými v atribútoch je komplikovaná.

1.1.2 Proces naskladnenia skladových položiek

Naskladňovanie skladových položiek bolo jednou z tém, ktorým sme sa pri analýze venovali najpodrobnejšie. V súčasnom stave systému prebieha naskladňovanie tak, že skladník naskenuje čiarový kód produktu a kód skladového umiestnenia, nie nutne v tomto poradí, a tým systému dá najavo, kde je položka uložená. Výber skladového umiestnenia je však kompletne na ňom, nepodlieha žiadnym pravidlám a zároveň je často kontraproduktívny pre ďalšie pohyby konkrétnej skladovej položky.

Optimalizácia naskladňovania sa teda stala jednou z najvyšších priorít a v priebehu celého vytvárania práce to bol pravdepodobne ten bod, ku ktorému sme sa vracali najčastejšie, keďže na nej závisia takmer všetky ostatné skladové procesy.

1.1.3 Proces vyskladnenia skladových položiek

Vyskladnenie si nezainteresovaný čitateľ môže predstaviť ako plnenie objednávky internetovým obchodom a jej následné predanie doručovacej spoločnosti. V zásade sa teda jedná o proces, pri ktorom je pracujúcemu skladníkovi pridelená úloha obsahujúca skladové položky z objednávky, respektíve jej časti alebo naopak niekoľkých objednávok zoskupených dohromady, a jeho povinnosťou je všetky spomínané skladové položky zozbierať zo skladu, presunúť na baliace miesto a pripraviť na odovzdanie spoločnosti, ktorá ich bude prepravovať do ich cieľovej destinácie.

Problém nastáva v momente, keď je plocha skladu príliš veľká a potrebné položky sú rozmiestnené neusporiadane, skladník teda často na splnenie svojej priradenej úlohy musí navštíviť skladové umiestnenia na opačných stranách skladu a jeho práca sa kvôli nutnosti presúvať sa medzi umiestneniami stáva pomalou a neefektívnou.

1.2 Požiadavky používateľov

Prvé používateľské požiadavky sme dostali hneď na začiatku predmetu BI-SP1 od vedúceho projektu. Tie boli však najmä pomocnými ukazovateľmi, akým smerom by sme sa s projektom mali vydať, a bolo na nás ako tíme, ktoré z nich sa rozhodneme navrhnúť a implementovať.

Dôležitým zdrojom informácií sa ale stala komunikácia s reálnymi používateľmi skladového systému. V rámci predmetu BI-SP1 sme spolu s celým tímom dostali možnosť navštíviť jeden zo skladov, ktorý systém Atlantis denne využíva, a zároveň sme mali šancu od zamestnancov tohoto skladu zistiť, aké úpravy by oni osobne považovali za prínosné alebo dokonca nutné.

Vďaka tejto návšteve sme získali množstvo podnetov, ktoré sme dôkladne rozobrali a následne sme z nich vybrali tie, ktoré podľa nás mali najväčší potenciál zlepšiť zážitok z používania systému a súčasne mali najväčší vplyv pri optimalizácií rýchlosti vykonávania procesov.

1.2.1 Rozmery skladových položiek a umiestnení

Pri diskusii požiadaviek s používateľmi systému sme narazili na problém s uchovávaním rozmerov produktov a umiestnení. Atribútu u skladových položiek sa nezdali ako dlhodobo využiteľné riešenie, a skladové umiestenia dokonca žiadnu možnosť uchovávania ich rozmerov nemali. Po konzultácii so zadávateľom a vývojovým tímom sme sa rozhodli k obom entitám pridať ďalšie parametre, ktoré používateľom umožnia zadať výšku, šírku a hĺbku produktov a umiestnení. Zároveň sme sa rozhodli pre skladové položky

uchovávať ich hmotnosť a pre skladové umiestnenia ich nosnosť, čo nám uľahčilo návrh a implementáciu ďalších optimalizácií.

1.2.2 Dostupnosť skladových umiestnení

Jednou z ďalších požiadaviek, ktorá vznikla pri konzultáciách s používateľmi systému, bolo pridanie evidencie dostupnosti skladových umiestnení. V sú-časnej verzii skladového systému používateľ nemá ako zistiť, kde sa dané skladové umiestnenie nachádza, a či je prístupné bez použitia akejkoľvek techniky, alebo je k jeho dosiahnutiu nutné použiť napríklad vysokozdvižný vozík. Môže teda nastať situácia, že skladník má za úlohu vyzdvihnúť konkrétnu skladovú položku, no keď sa dostane k skladovému umiestneniu, na ktorom by sa daná položka mala nachádzať, zistí, že je uložená v rukou nedosiahnuteľnej výške, a musí sa vrátiť po vysokozdvižný vozík.

Aby sa podobným situáciám dalo čo najlepšie predísť, používatelia systému nás požiadali o pridanie parametra, ktorý bude značiť, či sa dané skladové umiestnenie nachádza vo výške dosiahnuteľnej voľnou rukou, alebo je na jeho dosiahnutie potrebná technika. Zároveň skladové umiestnenia získajú číselný atribút, ktorého hodnota bude značiť, ako blízko sa nachádzajú ku kľúčovým bodom skladu.

1.2.3 Frekvencia vyskladňovania skladových položiek

Používatelia systému aktuálne v užívateľskom prostredí nevidia žiadne štatistiky týkajúce sa vyskladňovania jednotlivých skladových položiek. Nie sú teda schopní reagovať na zmeny vo vyskladňovaní, ani porovnávať jednotlivé položky medzi sebou. Na prvý pohľad to nemusí znieť ako dôležitá funkcia systému, avšak existuje mnoho situácií, v ktorých by toto malé rozšírenie pomohlo organizácii skladov. Ako konkrétny príklad uvediem prípad dvoch takmer identických produktov od dvoch rôznych výrobcov. Oba produkty majú takmer identickú nákupnú cenu a rovnako má vlastník internetového obchodu takmer identicky čistý zisk z ich predaja. Kvôli rozšíreniu iného sortimentu však predajca nemá dostatok skladového miesta, aby uskladnil oba druhy produktov, chcel by teda ukončiť predaj jedného z nich. Vie, že najvýhodnejšie bude ukončiť predaj produktu, ktorý si kupuje menej zákazníkov. Ako však získa informácie o ich predajoch bez toho, aby manuálne prechádzal všetky objednávky a predaje porovnával? Tu prichádza výhoda tejto funkcie. Pokiaľ všetky skladové položky v systéme už obsahujú údaje o frekvencii ich vyskladňovania, je pre používateľa veľmi

jednoduché si všetky položky vyfiltrovať podľa tohoto údaja a následne len prosto vybrať produkt, ktorý sa predáva menej než ten konkurenčný.

1.2.4 Doplnenie kusových balení skladových položiek

Väčšina veľkých skladov funguje tým spôsobom, že na skladových umiestneniach bližšie k zemi sú uložené menšie balenia skladových položiek, respektíve takzvané *kusovky*, a na skladových umiestneniach, ktoré už nie je možné dosiahnuť voľnou rukou, sú zväčša umiestnené paletové balenia. Pre bežný beh skladu je už toto uloženie veľkou pomocou, keďže pri plnení úloh skladníci nemusia príliš často používať vysokozdvižné vozíky. Problém však nastáva v momente, keď všetky malé balenia produktov skladníci vyskladnia, a aby ďalej mohli pokračovať vo svojej práci, musia otvoriť a rozdeliť väčšie balenie položky a doplniť ho na ľahšie dostupné miesto. Táto úloha však v systéme nie je žiadnym spôsobom automatizovaná a počet jednotlivých balení produktov nie je nijak zaznamenávaný. Zamestnanci teda popri svojich pridelených úlohách majú navyše ešte povinnosť kontrolovať stav *kusoviek*, a prípade, že zistia, že nejakému produktu dochádzajú kusové balenia, je ich extra povinnosťou tieto balenia doplniť. Pre skladový systém by teda bolo veľkým zlepšením, keby kontrola stavu *kusoviek* by prebiehala každý deň automaticky a systém na ich nedostatok upozorňoval.

1.3 Systémové požiadavky

Dôsledná analýza systému a zber informácií od používateľov vedú k formulácii funkčných a nefunkčných požiadaviek. Táto časť práce sa bude venovať práve im a jej cieľom bude presne vytýčiť všetky funkcie, ktoré toto nové rozšírenie systému bude, alebo naopak nebude, mať. Najskôr v krátkosti zhrniem, čo to vlastne sú systémové požiadavky a na čo slúžia.

Systémové požiadavky slúžia k presnému vymedzeniu hraníc systému a možno ich zaradiť do dvoch kategórií, funkčné požiadavky a nefunkčné požiadavky. Funkčné požiadavky môžeme definovať ako všetko, čo systém musí vykonať, od konkrétnych krokov, ktorými musí systém prejsť v rámci istého procesu, až po špecifiká právnych predpisov. Nefunkčné požiadavky opisujú ako systém funguje, aj keď na fungovanie systému nemajú priamy dopad. Môže sa jednať o požiadavky na jednoduchosť používania systému alebo limity používateľov, ktorí systém môžu využívať v jeden moment. [5]

1.3.1 Funkčné požiadavky

1.3.1.1 FP1 Rozmery skladových položiek

Systém bude uchovávať informácie o rozmeroch skladových položiek. U každej skladovej položky bude systém evidovať informácie o jej výške, šírke, hĺbke a hmotnosti. Zároveň bude systém svojim používateľom umožňovať skladovým položkám rozmery pridávať a následne ich upravovať.

1.3.1.2 FP2 Rozmery skladových umiestnení

Systém bude uchovávať informácie o rozmeroch skladových umiestnení. U každého skladového umiestnenia bude systém evidovať informácie o jeho výške, šírke, hĺbke a nosnosti. Zároveň bude systém svojim používateľom umožňovať skladovým umiestneniam rozmery pridávať a následne ich upravovať.

1.3.1.3 FP3 Dostupnosť skladových umiestnení

Systém bude evidovať informácie o dostupnosti skladového umiestnenia. Pojmom dostupnosť sa rozumie hodnota získaná na základe vzdialenosti daného skladového umiestnenia od vybraných dôležitých bodov v sklade. Dôležitým bodom môže byť napríklad baliace miesto, miesto naskladňovania a vyskladňovania produktov alebo stanovisko vysokozdvižných vozíkov.

1.3.1.4 FP4 Frekvencia vyskladňovania skladových položiek

Systém bude evidovať informácie o frekvencii vyskladňovania skladových položiek. Frekvenciou vyskladňovania sa rozumie hodnota získaná na základe počtu vyskladnení danej skladovej položky voči počtu vyskladnení ostatných skladových položiek za určitý uplynulý časový úsek.

1.3.1.5 FP5 Doplnenie kusových balení skladových položiek

Systém automaticky vykoná kontrolu počtu kusových balení skladových položiek a bude svojich používateľov upozorňovať v prípade nízkeho počtu spomínaných balení.

1.3.1.6 FP6 Návrhy pri naskladňovaní

Systém bude pri naskladnení skladových položiek analyzovať a navrhovať optimálne skladové umiestnenie, na ktoré by položka mala byť vložená.

Systém pri tomto procese bude pracovať s novými atribútmi skladových položiek a skladových umiestnení, ktoré boli pridané v tomto rozšírení.

1.3.1.7 FP7 Návrhy na presuny skladových položiek

Systém zobrazí vedúcemu skladu zoznam návrhov na presun sklado-vých položiek. Systém bude analyzovať polohu skladových položiek a v prípade nájdenia vhodnejšieho skladového umiestnenia pre konkrétnu položku zobrazí vedúcemu skladu návrh na presun tejto položky na nové umiestnenie. V prípade, že sa na novom umiestnení už nachádza iná skladová položka, systém zistí, či je možné položku na toto umiestnenie pridať. V prípade, že to možné nie je, systém vedúcemu skladu navrhne výmenu skladových položiek medzi dvoma skladovými umiestneniami.

1.3.1.8 FP8 Zoskupenie skladových položiek, ktoré sa vyskladňujú často spolu

Systém zobrazí vedúcemu skladu zoznam návrhov na zoskupenie skladových položiek, ktoré sa za určitý uplynulý časový úsek často vyskladňovali spolu. Systém načíta dáta o možných skupinách produktov a následne nájde vhodné možnosti ich zoskupenia.

1.3.2 Nefunkčné požiadavky

1.3.2.1 NP1 Začlenenie do systému

Rozšírenie bude implementované ako súčasť existujúceho skladového systému Atlantis a plynulo nadviaže na jeho súčasné funkcie.

1.3.2.2 NP2 Voľba technológií

Pri implementácii budú zvolené vhodné technológie vzhľadom na súčasný stav skladového systému. Backend rozšírenia bude implementovaný v jazyku PHP, konkrétne pomocou frameworku Symfony.

1.3.2.3 NP3 Možná nadväznosť ďalších rozšírení

Rozšírenie bude implementované tak, aby naň v budúcnosti bolo možné nadviazať ďalšími úpravami.

1.4 Analýza cieľového stavu

Po konzultácii s vedúcim práce, vývojovým tímom a používateľmi systému sme sa v rámci tímu v predmetoch BI-SP1 a BI-SP2 rozhodli vybrané požiadavky analyzovať hlbšie a zistiť, z implementácie ktorých z nich by systém naozaj benefitoval. Ako pomocné vodítko sme si určili niekoľko bodov, ktoré by finálne riešenie malo spĺňať.

- Všetky súčasné funkcie skladového systému zostanú aj naďalej použiteľné. Rozšírenie žiadnym spôsobom nenaruší súčasný stav a svojou formou sa mu čo najviac prispôsobí.
- U skladových položiek a skladových umiestnení budú evidované ich rozmery ako samostatné parametre zobraziteľné a upraviteľné v UI skladového systému.
- U skladovej položky bude evidovaná frekvencia jej vyskladňovania a zároveň bude možné zoznam položiek zoradiť podľa tohto údaju.
- U skladového umiestnenia bude evidovaná jeho dostupnosť voľnou rukou a zároveň jeho pozícia v rámci konkrétneho skladu.
- Skladový systém automaticky vykoná kontrolu kusových balení skladových položiek.
- Skladový systém pri naskladnení podľa novopridaných atribútov analyzuje a navrhne skladníkovi optimálne umiestnenie pre naskladňovanú položku.
- Skladový systém analyzuje umiestnenie skladových položiek v sklade a podľa novopridaných atribútov vedúcemu skladu navrhne presuny konkrétnych položiek na nové umiestnenia.

Kapitola 2

Návrh

Návrh implementácie je jednou z najkritickejších častí postupu riešenia akéhokoľvek projektu. Tento segment vývojového procesu zužitkuje všetky informácie získané v rámci analýzy a preformuluje ich do jasného plánu, podľa ktorého sa budú odvíjať nasledujúce fázy projektu.

Toto rozšírenie nebolo výnimkou, aj keď sa jedná o projekt menšieho rozsahu, návrh bol kľúčovou časťou celého procesu. Keďže počiatočné štádiá projektu vznikali v rámci predmetov BI-SP1 a BI-SP2 a celý jeho priebeh trval viac než rok, návrh bol priebežne podrobený úpravám, aby sa prispôsobil zmenám v už existujúcej implementácii, ktorá sa s uplynulým časom viac a viac menila. Nasledujúcu kapitolu som sa teda rozhodla rozdeliť do troch častí.

V prvej časti kapitoly priblížim vybrané návrhy, ktoré vznikli v rámci predmetov BI-SP1 a BI-SP2. Tieto návrhy vznikli po počiatočnej analýze projektu a jeho dokumentácie a po úvodných schôdzkach so zadávateľom projektu. Tieto návrhy odrážajú vtedajší stav skladového systému a nie sú v súlade aktuálnym návrhom implementácie optimalizácii.

V druhej časti kapitoly sa budem venovať súčasným návrhom riešenia jednotlivých funkčných a nefunkčných požiadaviek, ktoré som vytýčila v predošlej kapitole. Jedná sa o kompletný návrh adresujúci všetky systémové požiadavky, ktorého základ vyplýva z návrhov vytvorených v rámci tímového projektu, tie však boli podrobené mnohým podstatným úpravám, aby odrážali súčasný stav skladového systému.

V poslednej časti popíšem technológie, ktoré boli pri implementácii rozšírenia využité, a priblížim ich históriu a účel ich všeobecného použitia.

2.1 Návrh v rámci tímového projektu

Predmet BI-SP1 bol mojím prvým kontaktom s vytváraním konkrétnych návrhov implementácie a rovnako tomu bolo aj v prípade niektorých mojich tímových kolegov. Celý tento proces pre nás teda bol vzhľadom na naše chýbajúce skúsenosti značne náročný. Výsledkom však bola séria návrhov a diagramov, ktoré zodpovedali vtedajšiemu stavu systému, a bola možná ich realizovateľnosť aj napriek nášmu nedokonalému pochopeniu niektorých nuáns týkajúcich sa procesov prebiehajúcich v skladovom systéme.

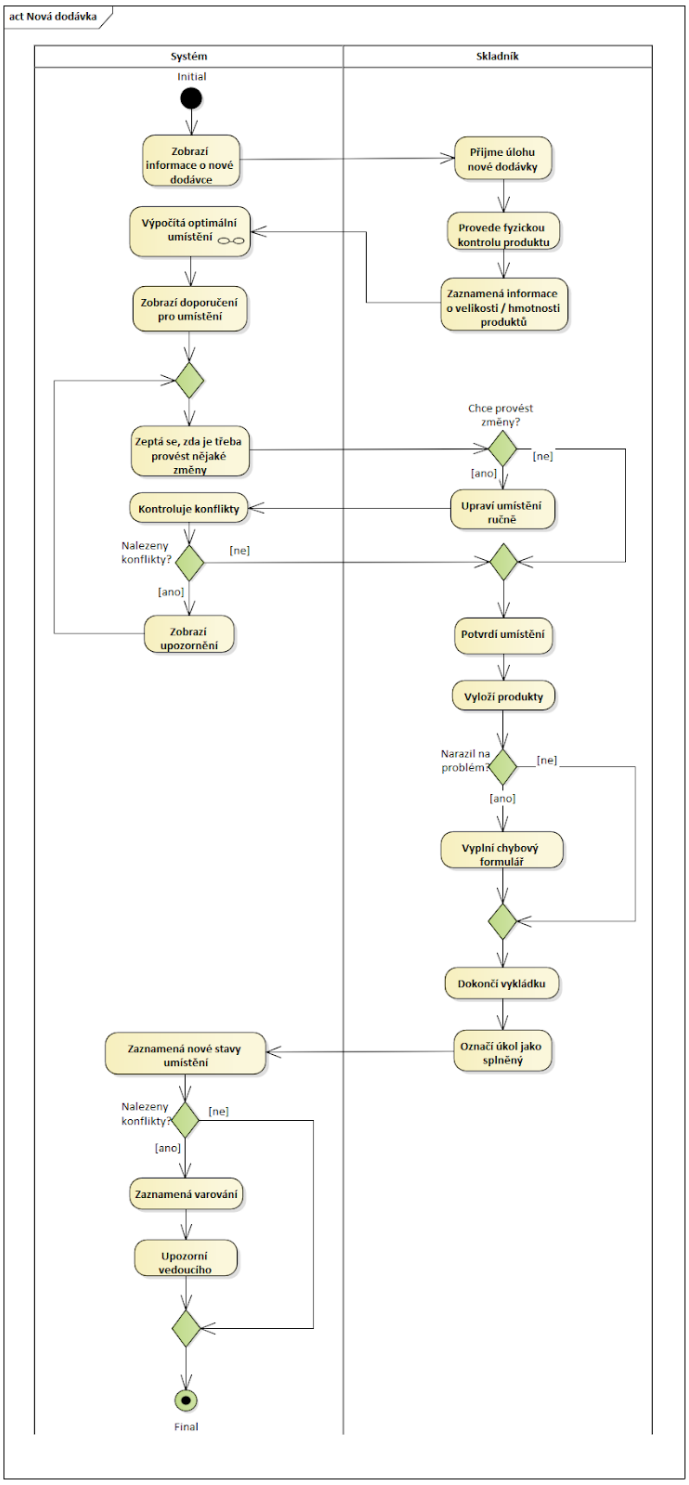
2.1.1 Návrhy pri naskladnení skladových položiek

Jednou z našich hlavných priorít už od úplného začiatku projektu bolo pomôcť s procesom naskladňovania. V súčasnosti systém iba zaznamenáva to, čo skladník vykoná, teda do databáze uloží, aká skladová položka bola novo umiestnená na aké skladové umiestnenie. Výber umiestnenia je však plne v moci skladníka ďalej a pri nevhodnom výbere komplikuje a znižuje efektivitu procesu vyskladňovania skladových položiek.

Naším pôvodným plánom bolo po začatí úlohy naskladňovania zamestnancovi pre každú naskladňovanú položku po jej naskenovaní zobraziť konkrétne umiestnenie, ktoré systém zvolil ako optimálne. Zamestnanec by túto voľbu mohol potvrdiť, alebo ju manuálne prepísať, a následne položku na toto umiestnenie uložiť.

Tento návrh však má niekoľko problémových častí. Úloha naskladňovania by bola prerušená každým odporučením umiestnenia pre všetky naskladňované položky a skladník by musel každý jeden návrh ručne potvrdiť alebo upraviť. Ďalším problémom je, že pri neoptimálnej implementácii by to znamenalo aj to, že po naskenovaní jedného balenia by skladník musel položku odniesť na priradené skladové umiestnenie, a až potom by sa mohol vrátiť k zvyšným naskladňovaným položkám.

Tieto nedostatky vyplývajú z nášho vtedajšieho povrchného porozumenia fungovania systému a po dlhšej práci s ním bolo jednoduché ich spozorovať a finálny návrh realizovať tak, aby odporúčanie ďalej skladových umiestnení pri naskladnení čo najmenej narušovalo prácu zamestnancov skladu.



Obr. 2.1: Diagram pôvodného návrhu naskladnenia skladových položiek

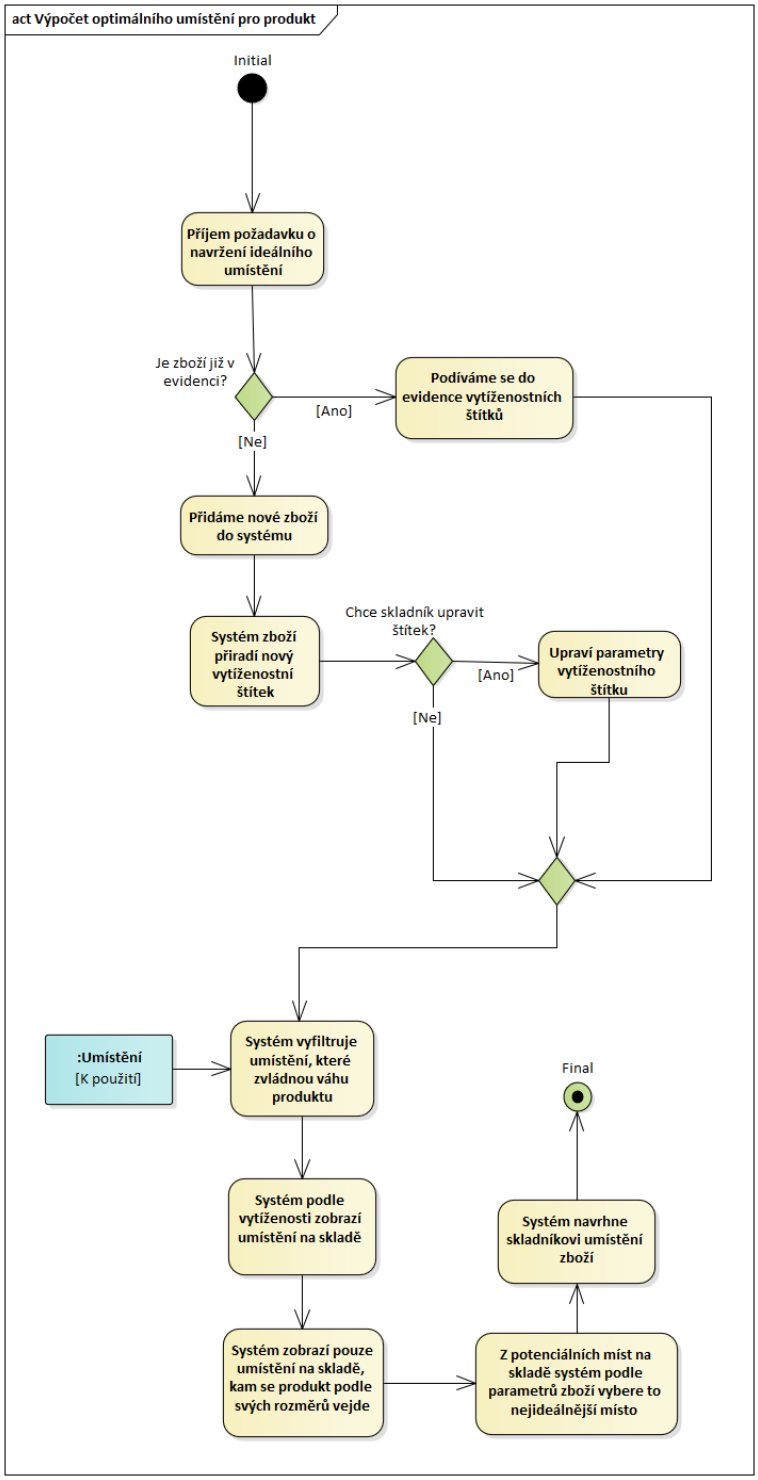
2.1.2 Výpočet optimálneho skladového umiestnenia pre skladovú položku

Ďalším z dôležitých návrhovm ktorým sme sa v rámci práce na tímovom projekte venovali, bolo riešenie optimalizácie uskladnenia položiek v rámci skladu. Znamenalo to pre nás nutnosť nájdenia spôsobu priraďovania skladových položiek na skladové umiestnenia tak, aby bol proces vysklad-ňovania čo najrýchlejší. Náš prvý finálny návrh bol značne odlišný od toho súčasného.

Rozhodli sme sa pre produkty evidovať takzvané *štítky*, ktoré by pomocou troch farieb odlišovali 3 rôzne levely frekvencie vyskladňovania všetkých skladových položiek. Farba štítku konkrétnej skladovej položky by sa priamo odvíjala od počtu kusov, ktoré boli vyskladnené za určitú uplynulú dobu. Počet vyskladnených kusov by bol naškálovaný oproti počtu vyskladnených kusov ostatných skladových položiek, aby bola získaná ďalej percentuálna hodnota, ktorá by následne bola prevedená na jeden z troch farebných štítkov.

Aby bolo možné položky uložiť na vhodné skladové umiestnenia, rozhodli sme sa umiestnení evidovať takzvaný index dostupnosti. Tento parameter sa vo istej svojej podobe zachoval aj v súčasnom návrhu, v tom pôvodnom bol však jeho výpočet značne zjednodušený. Keďže sme pri tvorbe pôvodného návrhu ešte nemali možnosť navštíviť osobne sklad, ktorý skladový systém využíva, návrh výpočtu indexu dostupnosti spočíval v prepočte vzdialenosti od jedného konkrétneho miesta v sklade na percentuálnu hodnotu. Tento výpočet nebral do úvahy ani výšku, v ktorej sa skladové umiestnenie nachádza, a teda či je dostupné voľnou rukou.

Značná nepresnosť týchto dvoch parametrov by po implementácii viedla k nedostatočnému zlepšeniu umiestňovania skladových položiek, takže by proti vyskladňovania takmer vôbec nezefektívnila, a naopak by mohlo dôjsť k problémom v prípade, že by systém navrhoval umiestnenie malých alebo krehkých položiek k paletovým baleniam, keďže index dostupnosti umiestnení neuchovával informáciu o výške umiestnenia od zeme.



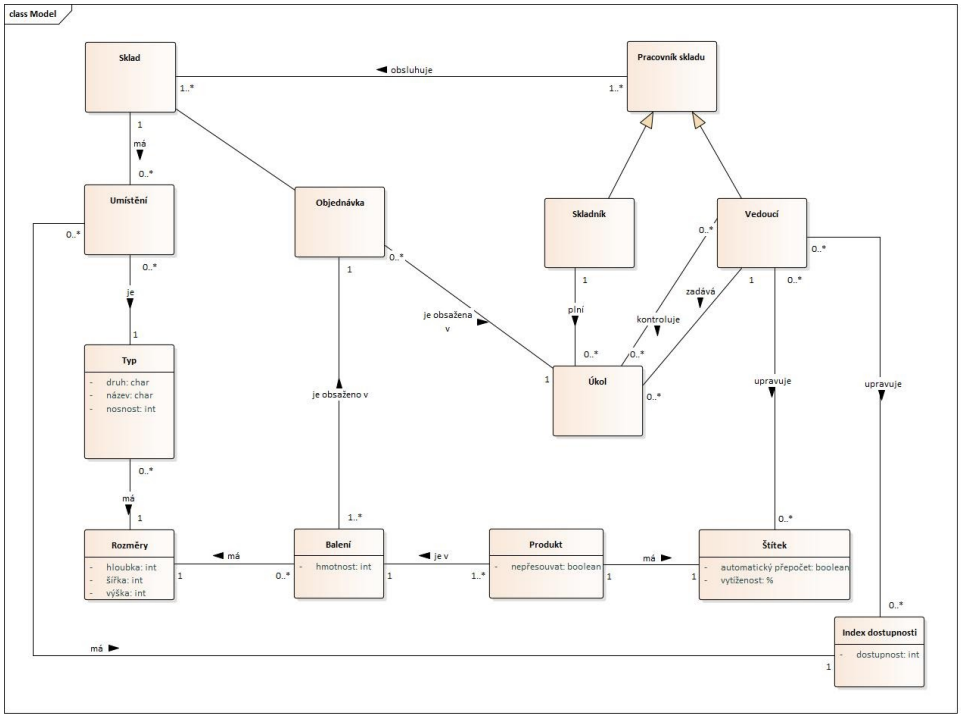
Obr. 2.2: Diagram pôvodného návrhu výpočtu optimálneho skladového umiestnenia pre skladovú položku

2.1.3 Zasadenie do existujúceho kódu

Pri analýze štruktúry vtedajšieho stavu projektu sme sa rozhodli hlavné rozšírenia do kódu zasadiť v podobe nových entít. V praxi to teda znamenalo, že napríklad rozmery bolo nutné pridať aj k umiestneniam rovnako ako k položkám, a teda sme navrhli vytvorenie novej entity s názvom *Rozmery*, ktorá by obsahovala atribúty výška, šírka a hĺbka. Nosnosť a hmotnosť, ktoré bolo potreba pridať k umiestneniam a položkám, v tomto poradí, sme sa v návrhu rozhodli zaradiť priamo k už existujúcim entitám skladovej položky a skladového umiestnenia, keďže nie sú opakovane priradené k žiadnej inej entite v celom skladovom systéme a sú len pomocnými atribútmi.

Podobne sme postupovali aj pri návrhu indexu dostupnosti a štítkov frekventovanosti vyskladňovania. Po zvážení všetkých pre a proti sme sa rozhodli ich vytvoriť ako samostatné entity, keďže sa jedná o jedny z hlavných častí tohto rozšírenia používané naprieč mnohými procesmi.

S odstupom času a po konzultácia s vývojovým tímom som sa však dostala do bodu, kde som všetky tieto závery prehodnotila, vzala do úvahy, či sú novo pridané entity naozaj tak dôležité, a návrhy som zásadne upravila.



Obr. 2.3: Doménový model pôvodného návrhu

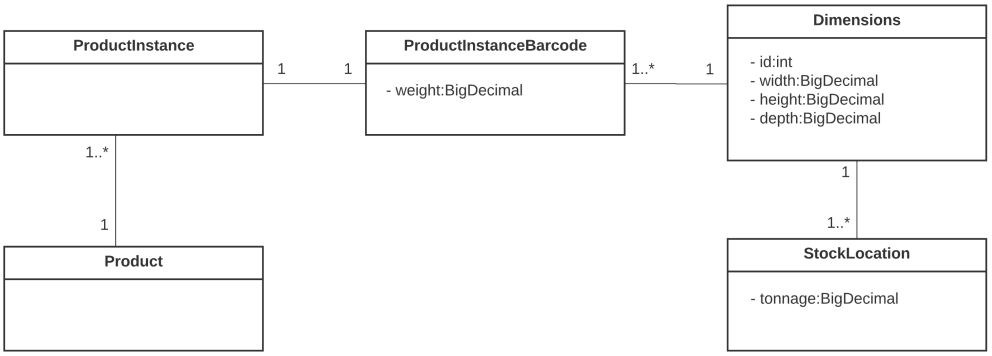
2.2 Návrh riešenia jednotlivých požiadaviek

V tejto podčasti priblížim finálne návrhy rozšírení založené na jednotlivých funkčných a nefunkčných požiadavkách z predošlej kapitoly. Tieto návrhy boli počas procesu ich tvorby a celého písania tejto bakalárskej práce niekoľkokrát zmenené a upravené tak, aby čo najefektívnejšie využívali súčasný stav systému, a následne boli podrobené ďalším úpravám po konzultáciách s vedúcim práce a vývojovým tímom skladového systému.

2.2.1 Rozmery skladových položiek a skladových umiestnení

Z dôslednej analýzy systému vyplynulo, že pridanie rozmerov položiek a umiestnení je jedným z najviac kľúčových rozšírení. V úvodnom návrhu z predmetu BI-SP1 sme pre súčasné požiadavky FP1 a FP2 navrhli pridanie novej entity rozmery a jej prepojenie položkami a umiestneniami. Na tomto pôvodnom návrhu bola založená aj jeho druhá vylepšená verzia zjednodušene zobrazená na obrázku 2.4.

Diagram zobrazuje 5 samostatných, entít ktoré sú medzi sebou prepojené. Už z ich pomenovania je vidieť rozdiel oproti pôvodnej prvej verzii návrhu, pri ktorej sme spolu s tímom ešte presne nevedeli, kam konkrétne do kódu rozšírenie zasadiť. Upravená verzia už naopak obsahuje presné pomenovania tried, ktorých sa rozšírenie malo týkať.



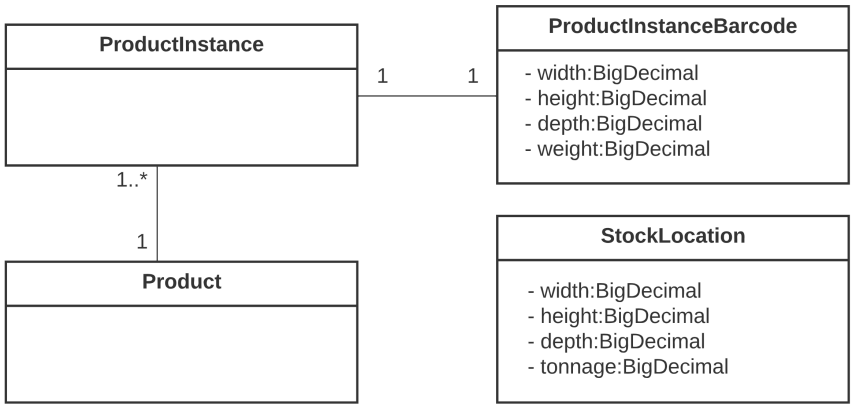
Obr. 2.4: Zjednodušený doménový model pôvodného návrhu rozmerov

Táto verzia návrhu spočívala v pridaní už spomínanej novej triedy *Dimensions*, ktorá by obsahovala atribúty id, width, height a depth, teda identifikátor každej skupiny rozmerov, šírku, výšku, a hĺbku. Hmotnosť skladovej položky by bola evidovaná ako atribút pridaný k triede *ProductInstanceBarcode*, ktorá reprezentuje jeden čiarový kód konkrétnej skladovej položky. Nosnosť skladového umiestnenia by bola obdobne evidovaná ako atribút pridaný k triede *StockLocation*, ktorá reprezentuje jedno konkrétne skladové umiestnenie. Všetky spomínané atribúty boli navrhnuté ako privátne a triedy, v ktorých sa nachádzali, teda obsahovali metódy pre nastavenie a získanie ich hodnôt.

Po čiastočnej implementácii tohoto návrhu sme však po konzultácii s vývojovým tímom skladového systému dospeli k záveru, že toto riešenie nie je optimálne, a bolo by lepšie rozmery kompletne integrovať do už existujúcich tried. Vzhľadom na toto rozhodnutie som teda pristúpila k tvorbe alternatívneho návrhu, ktorý ozmery začleňuje priamo do príslušných tried pre skladové položky a skladové umiestnenia.

Nový návrh, zjednodušene zobrazený na obrázku 2.5, teda úplne vynecháva pridanie novej triedy obsahujúcej tri rozmerové atribúty, namiesto toho tieto hodnoty pridáva priamo k triedam *ProductInstanceBarcode* a *StockLocation* k už spomínaným atribútom nosnosti a hmotnosti. Rovnako ako v predošlom návrhu sú tieto atribúty nastavené ako privátne a ich príslušné triedy teda musia obsahovať metódy pre nastavenie a získanie ich hodnôt.

Tento nový upravený návrh sa nakoniec stáva finálnym a je pripravený na to, aby implementácia požiadaviek FP1 a FP2 prebehla podľa neho.



Obr. 2.5: Zjednodušený doménový model nového upraveného návrhu rozmerov

2.2.2 Dostupnosť skladových umiestnení

Index dostupnosti skladového umiestnenia je percentuálna hodnota vyjadrujúca vzdialenosť daného umiestnenia od dôležitých bodov v sklade vzhľadom na ostatné umiestnenia. Dôležitým bodom v sklade môže byť napríklad baliace miesto alebo časť skladu, v ktorej sa prijímajú a odosielajú skladové položky. Používatelia skladového systému túto hodnotu budú mať možnosť vidieť vo výpise skladových umiestnení konkrétneho skladu a taktiež budú môcť podľa tohoto atribútu skladové umiestnenia zoradiť.

Pri vytváraní návrhu indexu dostupnosti sa objavilo niekoľko problémov, ktoré stručne popíšem, a návrh kvôli nim prebehol v niekoľkých iteráciách, podobne, ako tomu bolo pri návrhu rozmerov.

Pri prvej verzii návrhu sme sa sústredili hlavne na vzdialenosť skladových umiestnení od baliaceho miesta. Keďže priamo zo skladového systému nie je možné zistiť vzdialenosť jednotlivých umiestnení, oslovili sme spolupracujúci tím, ktorý má za úlohu spracovať mapu skladu spolu s jej presnými rozmermi, a spolu sme vytvorili návrh API, ktoré zabezpečuje predanie informácií o vzdialenostiach umiestnení z mapy. Jeho definícia je dostupná ako príloha tejto práce.

Táto verzia však príliš dobre nepracovala s rozlíšením umiestení, ktoré sa nachádzajú ďalej od baliaceho miesta, a umiestení, ktoré sa síce nachádzajú bližšie, ale na ich dosiahnutie je potrebné vysokozdvižný vozík. Pre systém podľa nášho návrhu tieto dve umiestnenia mohli byť ekvivalentné, ale pri reálnej prevádzke skladu použitie vysokozdvižného vozíka zaberie značne viac času než chôdza k umiestneniu, ktoré je síce ďalej, ale nachádza sa bližšie k zemi.

Tomuto problému sme sa v rámci tímu venovali na mnohých schôdzkach a po niekoľkých konzultáciách s vedúcim projektu a vývojovým tímom sme sa cez niekoľko iterácií dostali až k finálnemu návrhu.

Index dostupnosti bude realizovaný ako samostatná trieda obsahujúca priemernú vzdialenosť daného umiestnenia od dôležitých bodov v sklade a finálnu percentuálnu hodnotu dostupnosti. Pre každé umiestnenie bude taktiež evidovaná boolean hodnota značiaca jeho dostupnosť voľnou rukou. Ak sa skladové umiestnenie nachádza vo výške nižšej než 180 cm, táto hodnota bude nastavená ako nepravdivá, v opačnom prípade bude nastavená ako pravdivá.

Systém pri výpočte indexu dostupnosti konkrétneho umiestnenia odošle požiadavku s jeho identifikačným číslom na API mapy skladu, a to mu vzápätí vráti zoznam dôležitých miest v sklade a priemernú vzdialenosť toho konkrétneho umiestnenia od všetkých z nich. Táto hodnota sa pre každé umiestnenie uloží a systém následne prepočíta percentuálnu hod-

notu dostupnosti tak, že naškáluje priemerné vzdialenosti všetkých umiestnení v sklade vzhľadom na umiestnenie, ktoré je najďalej. Výsledkom bude hodnota, ktorá čím viac sa bude blížiť číslu 100, tým lepšie bude umiestnenie postavené v rámci celého skladu.

Celý tento proces výpočtu však prebehne iba pre skladové umiestnenia najbližšie k zemi. Pre ostatné umiestnenia sa hodnota dostupnosti počítať nebude, a naopak im systém priradí rovnakú hodnotu, akú majú umiestnenia pod nimi. Spolu so spomínaným boolean atribútom tak dosiahneme rozlíšenie umiestnení na základe ich vzdialenosti, ale zároveň sa nevyskytne problém s rozlišovaním ďalekého umiestnenia vo výške 50 cm a blízkeho umiestnenia vo výške 4 m, čo splní požiadavku FP3.

2.2.3 Frekvencia vyskladňovania skladových položiek

Pre splnenie požiadavky FP4 pridám do systému evidenciu hodnoty frekvencie vyskladňovania pre každú skladovú položku. Frekvencia bude realizovaná pomocou novej samostatnej triedy, ktorá bude obsahovať dva atribúty, percentuálnu hodnotu frekvencie a boolean hodnotu značiacu, či sa frekvencia má automaticky prepočítavať.

Ďalej do systému pridám novú entitu, ktorej účelom bude evidencia historickej frekvencie vyskladňovania pre všetky položky. Prakticky to teda bude znamenať novú databázovú tabuľku, z ktorej sa ku každému páru skladovej položky a dátumu priradí príslušný počet vyskladnení pre danú položku v daný deň.

Systém bude frekvenciu vyskladňovania prepočítavať pre každú položku každý deň. Proces výpočtu prebehne tak, že systém najskôr preiteruje vyskladnenia za posledných 24 hodín, a pre každú položku do spomínanej tabuľky zaznamená, v akom počte vyskladnení sa objavila. Jedným z návrhov bolo evidovať počet vyskladnených kusov každej položky, v tomto prípade by však výpočet rozhodili položky, ktoré sú pravidelne vyskladňované vo veľkých množstvách, napríklad drobné predmety ako gombíky alebo klince.

Následne systém zo spomínanej tabuľky získa pre každú položku počet jej vyskladnení za uplynulých 30 dní a tieto hodnoty naškáluje vzhľadom na položku, ktorá bola najviac vyskladňovaná. Získa tým percentuálnu hodnotu, kde hodnota 100 bude znamenať najvyskladňovanejšiu položku a nižšie hodnoty budú priamo úmerné počtu vyskladnení daných položiek vzhľadom na maximum.

Vedúci skladu bude mať možnosť túto percentuálnu hodnotu pri každej položke manuálne upraviť a pozastaviť jej automatický prepočet. Predíde

tak tým situáciám, kedy by za iných okolností nárazový predaj jednej skladovej položky ovplyvnil všetky hodnoty na nasledujúci mesiac.

2.2.4 Doplnenie kusových balení skladových položiek

Požiadavka FP5 je úzko spojená s predchádzajúcou. Cieľom je, aby systém monitoroval stav kusových balení skladových položiek a bol schopný vedúcemu skladu ukázať ich aktuálny počet. Keďže môžu existovať položky, ktoré sú vyskladňované minimálne, bolo by nevhodné, keby systém vedúceho upozorňoval na ich nedostatok, aj keď by na sklade bolo viac než je potrebné na najbližších niekoľko mesiacov. Rozhodla som sa teda využiť záznamy frekvencie vyskladňovania položiek a založiť na nich aj túto časť rozšírenia.

Systém preiteruje skladové umiestnenia dosiahnuteľné voľnou rukou a skladové položky na nich a následne pre každú položku porovná počet aktuálne naskladnených kusových balení s priemerným počtom denných vyskladnení danej položky za určitý uplynulý čas. V prípade, že počet kusov danej skladovej položky je značne nižší než počet priemerných denných vyskladnení, bude táto položka zobrazená na zozname skladových položiek, ktoré budú potrebovať čoskoro doplniť. Vedúci skladu teda bude mať informácie o stave kusoviek, no zároveň nebude obťažovaný nutnosťou potvrdzovať alebo rušiť konkrétne návrhy doplnenia.

2.2.5 Návrhy pri naskladňovaní

Proces naskladňovania nových skladových položiek je aktuálne úplne v rukách skladníka, ktorý túto úlohu vykonáva. Je iba na ňom, kam dané položky umiestni, a systém sa nestará, či je vybrané umiestnenie vhodné. Pre splnenie požiadavky FP6 je teda nutné vytvoriť proces, ktorý dá skladníkovi možnosť skladové položky umiestňovať efektívnejšie.

Samotnú úlohu naskladnenia je možné previesť dvoma spôsobmi. V tom prvom skladník najskôr naskenuje kódy všetkých skladových položiek, ktoré sú v novej dodávke, a až následne ich umiestňuje do skladu. V detaile úlohy pritom vidí, na ktoré umiestnenia je položky možné uložiť, avšak toto zobrazenie berie do úvahy všetky umiestnenia, ktoré nie sú blokované. Druhý spôsob naskladnenia je vykonaný naopak. Skladník najskôr naskenuje kód umiestnenia a následne kód skladovej položky, ktorú naň plánuje umiestniť. V tomto prípade pre položku nevidí žiaden zoznam možných umiestnení. Návrh realizácie teda tiež rozdelím do dvoch prípadov.

V prvom prípade systém po naskenovaní položky preiteruje všetky umiestnenia, ktoré nie sú blokované, a pokúsi sa nájsť umiestnenie s vyhovujúcimi

rozmermi a nosnosťou, ktoré zároveň svojou dostupnosťou bude zodpovedať frekvencii vyskladňovania danej položky. Ak vyhovujúcich umiestnení systém nájde viac, zoradí ich podľa zodpovedajúcej dostupnosti. Tieto umiestnenia sa budú naďalej zobrazovať v detaile úlohy ako návrhy a skladník nebude nútený ich využiť, budú však vizuálne oddelené. V prípade, že skladník umiestni položku na umiestnenie, ktoré systém bude považovať za neefektívne, teda frekvencia vyskladňovania položky a dostupnosť tohoto umiestnenia si budú protirečiť, systém skladníka upozorní na tento fakt a navrhne mu lepšie umiestnenie.

V druhom prípade, keď skladník rovno skenuje kód umiestnenia, nie je možné využiť prvú časť tohoto procesu. Systém v tom prípade iba skontroluje, či vybrané umiestnenie vyhovuje, a ak ho bude považovať za neefektívne, skladníka upozorní rovnako ako v prvom prípade a navrhne mu lepšie umiestnenie.

2.2.6 Návrhy na presuny skladových položiek

Aby bolo možné využívať skladový systém čo najviac efektívne, je potreba, aby aj fyzický sklad bol upravený. Z tohoto dôvodu sme zvolili možnosť presúvania skladových položiek v rámci jednotlivých skladov. Skladový systém už aktuálne podporuje zadanie úlohy obsahujúcej presun položiek z jedného umiestnenia na iné. V rámci optimalizácie sme sa rozhodli túto funkciu využiť a zefektívniť polohu všetkých položiek v sklade. Hlavným výsledkom bude zrýchlenie procesu vyskladnenia a zároveň lepšia organizácia uskladnenia položiek.

Systém presuny nebude vykonávať náhodne, naopak. Bude sa jednať o návrhy, ktoré budú dôkladne prepočítané a používateľom sa zobrazia len tie, ktoré skladu budú naozaj benefitovať.

Návrhy na presuny skladových položiek sú z hľadiska rozsahu najväčšou singulárnou časťou tejto práce. Zahŕňajú totiž niekoľko častí, ktoré sú následne používateľovi skladového systému prezentované ako jeden úplný celok. Pre potreby návrhov však tento celok môžeme rozdeliť na spomínané menšie časti a venovať sa každej z nich zvlášť. Prvé tri typy pokrývajú rozsah požiadavky FP7, a posledná časť rieši požiadavku FP8.

2.2.6.1 Presun skladových položiek

Najzákladnejší presun skladových položiek je celkom presne vysvetlený už svojím názvom. Jedná sa o kompletné presunutie všetkých kusov jednej skladovej položky na iné vyhovujúce umiestnenie v sklade.

Systém preiteruje všetky rukou dostupné umiestnenia a položky na nich s využitím indexu dostupnosti a hodnoty frekvencie vyskladňovania prepočíta, či sú položky vhodne umiestnené. V prípade, že nájde skladovú položku, ktorej frekvencia vyskladňovania zásadne nezodpovedá dostupnosti umiestnenia, na ktorom je uložená, pokúsi sa jej nájsť nové optimálnejšie umiestnenie.

To znamená, že postupne skontroluje umiestnenia, ktorých dostupnosť zodpovedá frekvencií vyskladňovania položky, v prípade, že vhodné umiestnenie nenájde, rozšíri hranice hodnôt indexu dostupnosti kontrolovaných umiestnení oboma smermi, a tento proces bude opakovať, kým nenájde umiestnenie vhodnejšie než to, na ktorom sa položka aktuálne nachádza, respektíve kým nezistí, že takéto umiestnenie neexistuje. Nové umiestnenie je vhodné v prípade, že jeho index dostupnosti zodpovedá frekvencii vyskladňovania položky viac než to aktuálne a zároveň je na novom umiestnení dostatok miesta na to, aby tam boli pridané ďalšie položky.

Kontrolu voľného miesta systém prevedie pomocou pomocného algoritmu, ktorý vezme do úvahy rozmery položky, rozmery umiestnenia a rozmery prípadných položiek, ktoré sa už na umiestnení nachádzajú, a jednoduchým výpočtom zistí, či je voľný objem umiestnenia dostatočný.

Ak táto kontrola prebehne úspešne, a teda je presun možné vykonať, bude zaradený do zoznamu možných presunov, ktoré sa finálne zoradia podľa svojej nápomocnosť, a tento zoradený zoznam sa následne zobrazí vedúcemu skladu v návrhoch na presuny.

2.2.6.2 Výmena skladových položiek

Výmena položiek je špeciálny typ presunu, pri ktorom systém v rámci procesu kontroly nájde rôzne dve skladové položky na nevyhovujúcich umiestneniach, a určí, že by obe mali byť presunuté na umiestenie tej druhj položky z tohoto páru. Systém teda tieto dva presuny zlúči a vytvorí výmenu. Výmena sa obdobne ako bežný presun zobrazí vedúcemu skladu v návrhoch na presuny.

2.2.6.3 Zlúčenie skladových položiek

V prípade, že systém detekuje, že rovnaký typ skladovej položky sa nachádza na niekoľkých rôznych skladových umiestneniach, pokúsi sa nájsť spôsob, ako ich zlúčiť na čo najmenej umiestení. Jedná sa teda o ďalšiu variáciu na bežný presun, systém ale tentokrát nebude hľadať akékoľvek umiestnenie s voľným miestom, ale bude sa snažiť nájsť najoptimálnejšie umiestnenie, na ktorom sa už tento druh skladovej položky vyskytuje. Podstatnou

časťou algoritmu bude spojenie položiek na čo najmenší počet umiestnení, no zároveň systém bude dbať na to, aby sa nepresunula iba časť položiek z jedného umiestnenia na druhé, ale aby mohli byť presunuté všetky.

2.2.6.4 Zoskupenie skladových položiek, ktoré sú často vyskladňované spolu

Posledným podtypom presunov skladových položiek je zoskupovanie tých, ktoré sú často vyskladňované spoločne. Podklady pre rozhodnutie, o ktoré položky sa jedná, budú získané od spolupracujúceho tímu, ktorý má na starosti predikcie objednávok v rámci skladového systému.

Systém v tomto prípade nebude hľadať vhodné umiestnenie na základe indexu dostupnosti a hodnoty frekvencie vyskladňovania daných položiek, musí sa nájsť vhodné umiestnenie pre jednu z položiek tak, aby sa nachádzala čo najbližšie tej druhej. Primárne sa systém bude snažiť presunúť tú z položiek, ktorej presun sa bude dať vykonať bez akýchkoľvek ďalších presunov v rámci skladu. Výsledný návrh bude takisto prezentovaný vedúcemu skladu a bude mať vlastnú kategóriu na obrazovke návrhov na presuny.

2.3 Použité technológie

Vzhľadom na skutočnosť, že celá práca sa zaoberá rozšírením už existujúceho riešenia, všetky použiteľné technológie boli už vopred dané. Backend skladového systému Atlantis je od roku 2019 vyvíjaný v jazyku PHP, konkrétne v jeho frameworku Symfony, objektovo-relačné mapovanie zabezpečuje knižnica Doctrine a všetky dáta systém ukladá do databáze používajúcej systém PostgreSQL. V následujúcich častiach analýzy priblížim všetky tieto technológie a ich využitie.

2.3.1 PHP

PHP je skriptovací programovací jazyk využívaný najmä na programovanie dynamických internetových stránok a webových aplikácií. Jeho syntax je zo značnej časti podobná jazykom C, Java a Pearl, za účelom priblížiť sa čo najviac webovým vývojárom a svojou familiárnosťou im tak zjednodušiť prechod na nový programovací jazyk. Jeho prvá verzia vznikla už v roku 1995 a odvtedy je stále aktívne vyvíjaný. [6]

Keďže celý backend skladového systému Atlantis je navrhnutý a realizovaný v tomto jazyku a moja bakalárska práca je sústredná na rozšírenie stávajúceho riešenia, znamenalo to, že aj implementačná časť mojej bakalárskej práce bude celá realizovaná v jazyku PHP.

2.3.2 Symfony

Symfony je framework zameraný na tvorbu webových aplikácií v jazyku PHP a zároveň súbor opakovane použiteľných PHP komponentov. Práve tieto komponenty sú jeho najväčšou prednosťou, vďaka ich možnosti opakovaného použitia sú základom pre mnohé iné frameworky jazyka PHP. Jeho prvá verzia bola publikovaná v roku 2005 a až do súčasnosti je stále udržiavaná spoluprácou viac než troch tisíc dobrovoľných prispievateľov a spoločnosti SensioLabs. [7]

2.3.3 Doctrine ORM

The Doctrine Project je súborom knižníc pre jazyk PHP primárne zameraných na databázové riešenia a objektovo relačné mapovanie. Dvoma najhlavnejšími knižnicami tohoto projektu sú ORM a DBAL.

Doctrine ORM je objektovo relačný mapovač, ktorý pracuje s vlastnou abstraktnou databázovou vrstvou (DBAL) a jednou z jeho hlavných vlastností je, že dokáže vytvárať databázové dotazy vo svojom proprietárnom objektovo orientovanom dialekte SQL s názvom Doctrine Query Language. [8]

2.3.4 PostgreSQL

PostgreSQL je open source objektovo relačný databázový systém, ktorý využíva jazyk SQL. Vznikol pod názvom POSTGRES na University of California v Berkeley za sebou viac ako 30 rokov aktívneho vývoja od svojich dobrovoľných prispievateľov. Medzi jeho mnohé výhody patria hlavne schopnosť bežať na všetkých bežných operačných systémoch, definovať svoje vlastné dátové typy, zostavovať vlastné funkcie a najmä fakt, že pokrýva minimálne 170 zo 179 funkcií špecifikácie SQL:2016. [9]

Kapitola 3

Realizácia

V tejto časti práce sa budem venovať procesu implementácie rozšírenia, ku ktorému viedli predošlé dve kapitoly. Táto kapitola *nepopisuje* kompletný proces realizácie funkčných požiadaviek, ale iba jeho kľúčové časti, respektíve situácie, ktoré viedli k zásadným zmenám v návrhu, keďže proces implementácie prebehol v súlade s plánom vytýčeným v kapitolách analýzy a návrhu, respektíve niektoré časti procesu boli zahájené už počas predmetov BI-SP1 a BI-SP2, no veľká časť ich výsledkov sa s plynutím času stala kvôli zmenám v skladovom systéme nepoužiteľnou a boli nahradené novými riešeniami.

V druhej časti tejto kapitoly popíšem stav optimalizácií po odovzdaní bakalárskej práce a následne zhrniem možné rozšírenia, ktorými by bolo možné nadviazať na túto bakalársku prácu.

3.1 Implementácia optimalizácií

Ako som už spomenula v úvode tejto kapitoly a niekoľkokrát naprieč celou prácou, táto práca začala ako tímový projekt v predmete BI-SP1 a pokračovala aj v priebehu BI-SP2, kedy už som vedela, že táto téma bude mojou bakalárskou prácou, a teda som sa podieľala aj na implementácii, ktorú mali za úlohu moji tímoví kolegovia.

Keďže sa celý projekt vyvíjal v čase dlhšom než jeden rok, pri všetkých častiach procesu softvérového vývoja, ktorými sme si prešli, nastali problémy. Inak tomu nebolo ani pri samotnej implementácii. V nasledujúcich podčastiach popíšem dôležité časti implementácie rozdelené do tématických celkov, niektoré z problémov, ktoré zásadne ovplyvnili priebeh implementácie, a ich následné riešenie.

3.1.1 Súbežná implementácia backend a frontend častí

Keďže táto práca je len jednou polovicou kompletného riešenia, a tú druhú vo svojej práci popisuje Vít Urban, pôvodný plán bol backend a frontend každého rozšírenia implementovať zároveň. To sa ale ukázalo takmer nemožným s pribúdajúcim počtom nutných úprav už existujúcich backend riešení.

Pristúpili sme teda k definovaniu návrhu API požiadaviek, ktorý zaistil, že aj keď sa naše implementácie časovo rozchádzali, stále boli schopné medzi sebou vzájomne komunikovať. Nevýhodou tohoto riešenia bolo, že rozšírenia nemohli byť ihneď otestované, a aj v kompletne dokončenom stave budú musieť byť obe implementácie podrobené zmenám, aby na seba korektne nadväzovali a spracovávali správne dáta. Je to však len drobná obeť
na úkor možnosti pracovať vlastným tempom a venovať sa požiadavkám v tom poradí, aké každý z nás uznal za vhodné. Spomínaný návrh API pripájam v prílohách práce.

3.1.2 Rozmery skladových položiek a skladových umiestnení

Pridanie rozmerov skladovým položkám a skladovým umiestneniam bol prvý implementačný krok, na ktorý sme sa ako tím odhodlali, keďže sme ich implementáciu brali iba ako pridanie niekoľkých atribútov k dvom triedam. Čas však ukázal, že to nie až tak jednoduché.

V úplne prvej fáze implementácie sme postupovali podľa ranného návrhu, a tieto atribúty pridali priamo do tried *Product* a *StockLocation*. Počas priebežných konzultácií a podrobnejšej analýze projektu sme však usúdili, že bude lepšie rozmery ako také oddeliť, a vytvoriť pre ne samostatnú triedu *Dimensions*, ako už som ukázala v kapitole *Návrh*.

V druhej fáze implementácie rozmerov sme teda vytvorili samostatnú triedu obsahujúcu tri atribúty, výšku, šírku a hĺbku, a príslušné metódy, ktoré zabezpečovali nastavovanie a získavanie hodnôt z týchto atribútov, keďže tie boli nastavené ako privátne. K triede *Dimensions* bolo nutné vytvoriť pomocné súbory *DimensionsFormatter, DimensionsRepository a DimensionsService*, aby bolo možné rozmery zobraziť na frontend, a zároveň k nim pristupovať z iných častí kódu. *DimensionsService* bola napríklad mimo iné zodpovedná za vytvorenie alebo upravenie rozmerov konkrétneho umiestnenia alebo položky, keď na túto akciu prišla požiadavka z frontendu, ako je ukázané v ukážke kódu 3.1.

```
public function create(BigDecimal $width, BigDecimal $height,
                       BigDecimal $depth): ?Dimensions
{
    $dimensions = new Dimensions();
    $dimensions->setWidth($width);
    $dimensions->setHeight($height);
    $dimensions->setDepth($depth);

    $this->entityManager->persist($dimensions);
    $this->entityManager->flush();
    return $dimensions;
}
```

Obr. 3.1: Ukážka vytvorenia novej entity Dimensions z parametrov z requestu

Po jednej z finálnych konzultácií ohľadom tejto bakalárskej práce sme však spolu s vedúcim práce prišli k záveru, Že pôvodný plán pridania atribútov k triedam zaoberajúcim sa skladovými položkami a skladovými umiestneniami bol optimálnejší a aktuálna forma implementácie by mala byť upravená do tejto podoby. To znamená preradenie atribútov výšky, šírky a hĺbky do tried *ProductInstanceBarcode a StockLocation*. Túto skutočnosť som uviedla v upravenom návrhu, ktorý je popísaný v príslušnej kapitole.

Ďalší problém však nastal pri spätnej väzbe od používateľov skladu. Pridanie rozmerov k skladovým položkám pri aktuálnom návrhu závisí na možnosti čo najpresnejšie zmerať konkrétne balenie skladovej položky a následne jej tieto hodnoty priradiť. Alternatívou pri nemožnosti zmerať všetky druhy balení konkrétnej položky bolo získanie rozmerov čo najmenšieho balenia a následné násobenie rozmerov pre získanie údajov o väčších baleniach. Zásadný problém prichádza v momente, keď zamestnanci skladu nie sú schopní zmerať presné rozmery malého balenia, respektíve väčšie balenie nezodpovedá násobku malého balenia, a teda systém nedokáže získať presné informácie rozmeroch skladových položiek, čím činí toto rozšírenie nevyužiteľným a taktiež eliminuje časti ostatných rozšírení, ktoré s rozmermi položiek a umiestnení pracujú vo svojich výpočetných algoritmoch.

Rozmery teda aktuálne zostávajú realizované vo svojej poslednej plnej verzii, a vhodná verzia ich implementácie je veľmi komplikovaná na určenie, keďže všetky návrhy, ktoré sa zdali byť funkčné, po dokončení ich imple-

mentácie narazili na zásadný problém, ktorý využite tohoto rozšírenia znížil alebo dokonca úplne znemožnil.

3.1.3 Index dostupnosti skladových umiestnení

Index dostupnosti počas procesu svojej implementácie taktiež prešiel niekoľkými, zmenami podobne ako rozmery, našťastie však tieto zmeny neboli až tak drastické. Jeho výpočet čerpá informácie z mapy skladu. Jej implementácia však v čase návrhu a realizácie nebola hotová a teda sme s tímom, ktorý ju mal na starosti, vytvorili návrh API, podľa ktorého by tieto dve časti spolu mali komunikovať. Ten sa však niekoľkokrát zmenil na základe požiadaviek nášho a aj spolupracujúceho tímu, teda implementácia na oboch stranách sa musela čiastočne meniť. Finálnu verziu návrhu API je možné nájsť v prílohách práce.

Po zmenách prístupu, ktoré som už uviedla v návrhu indexu dostupnosti vkladových umiestnení, sme v rámci tímu BI-SP2 skompletizovali aktuálne finálnu verziu implementácie, ktorá sa zatiaľ spolieha na dáta z mockovaného API, keďže finálna implementácia mapy skladu ešte nie je hotová.

Index dostupnosti je realizovaný ako samostatná trieda spolu so svojimi príslušnými *Repository a Service* súbormi, ktoré sa starajú o komunikáciu so zvyškom systému. Táto trieda je napojená na konkrétne skladové umiestnenie, a obsahuje atribúty *accessibilityRaw*, ktorý eviduje vzdialenosť získanú z mapy, *accessibility*, ktorý eviduje finálnu percentuálnu hodnotu dostupnosti, atribút *lock*, ktorý svojou hodnotou odráža, či je prepočet indexu dostupnosti zamknutý alebo nie, a atribút *reachable*, ktorý odráža, či je umiestnenie dosiahnuteľné voľnou rukou. Všetky spomínané atribúty majú príslušné metódy na získanie a nastavenie ich hodnôt.

Výpočet indexu dostupnosti prebehne súlade s navrhnutým algoritmom, teda systém získa z mapy vzdialenosti od dôležitých miest, a následne ich naškáluje vzhľadom ďalej na umiestnenie s najvyššou hodnotou vzdialenosti.

3.1.4 Frekvencia vyskladňovania skladových položiek

Implementácia frekvencie vyskladňovania skladových položiek prebehla podľa očakávaní získaných z predošlej analýzy a návrhu. V jej aktuálnej forme sú realizované obe navrhnuté časti.

Prvá z nich je pomocná trieda *DailyTurnover* spomínaná v návrhu, vrátane *Repository, Service a Formatter* súborov, ktorej úlohou je ukladať informácie o vyskladneniach každej skladovej položky v konkrétny deň. Tieto dáta získa zo všetkých vyskladnení konkrétneho skladu pomocou SQL dotazu, ktorý je ukázaný v ukážke kódu 3.2.

Druhá časť je tá hlavná, trieda *Turnover*, ktorá z dát pomocnej triedy získa počet vyskladnení položiek za posledných 30 dní a následne ich naškáluje tak, že najvyskladňovanejšia skladová položka dostane priradenú hodnotu 100 a ostatným budú priradené percentuálne hodnoty priamo úmerné pomeru počtu ich vyskladnení voči maximu.

```
public function findFromDate(Carbon $datetime): array
{
    return $this ->getEntityManager()
        ->createQueryBuilder()
        ->select('s')
        ->from(StockPicking::class, 's')
        ->where('s.createdAt >= :date')
        ->andWhere('s.createdAt < :nextDay')
        ->setParameter('date', $datetime)
        ->setParameter('nextDay', $datetime->addDay())
        ->getQuery()
        ->execute();
}
```

Obr. 3.2: Ukážka SQL dotazu vracajúceho vyskladnenia od určitého dátumu

3.1.5 Návrhy na presuny skladových položiek

V rámci prípravy na implementáciu presunov skladových položiek vznikol pomocný návrh a následne realizácia algoritmu, ktorý ako parametre príjme skladovú položku, počet jej kusov a skladové umiestnenie, a jeho úlohou je zistiž, či sa daný počet kusov skladových položiek zmestí na cieľové skladové umiestnenie. Výpočet prebieha veľmi jednoduchou formou, a to porovnaním objemov spolu s prirátaním percentuálnej rezervy. Tento algoritmus sa zavolá vždy, keď systém bude chcieť presunúť položky z jedného umiestnenia na iné.

Zároveň vzniklo riešenie presunov položiek implementované ako nová entita *MovementItem* spolu s príslušným *Service* súborom, ktorá sa zaoberá presunom položiek na nové umiestnenie.

Obe tieto časti sú však silno závislé na funkčnosti a presnosti rozmerov zadaných do systému. V prípade, že funkcia ďalšie rozmerov nebude v praxi fungovať tak, ako bolo navrhnuté, presuny skladových položiek budú v najlepšom prípade nepresné. Ich implementáciu by tak bolo nutné upraviť, pokiaľ by sa zmenila implementácia rozmerov.

3.1.6 Doplnenie kusových balení skladových položiek

Implementácia doplnenia kusových balení skladových položiek je čiastočne závislá na implementácii frekvencie vyskladňovania, respektíve jej pomocnej časti. Je to preto, že algoritmus funguje na základe porovnávania priemerného počtu vyskladnených skladových položiek, a teda využíva niektoré funkcie tejto triedy.

Systém pri kontrole kusov najprv zistí ich počet nachádzajúci sa aktuálne v sklade a potom ho porovná s priemerným počtom vyskladnených položiek za posledných niekoľko dní. V prípade, že aktuálny počet je kriticky nízky, systém zaradí túto položku na zoznam, ktorý sa následne odošle na frontend a zobrazí vedúcemu skladu.

3.2 Možné rozšírenia

Aktuálny stav práce si vyžaduje sfunkčniť prepojenie frontend a backend častí, čo zahŕňa aj drobné úpravy v oboch implementáciách. Zároveň je možné, že súčasný stav skladového systému nemusí podporovať niektoré z implementovaných rozšírení, keďže jeho stav sa stále vyvíja a mení.

Na prácu je možné nadviazať niekoľkými rôznymi spôsobmi. Jedným z nich je rozvinutie implementácie častí, ktoré v tejto práci neboli kompletne implementované. Jedná sa o podtypy presunov, keďže v aktuálnej fáze je kompletne implementovaný iba základný presun skladovej položky na nové umiestnenie. Je teda možnosť nadviazať implementáciou výmen, zlúčení a zoskupení.

Ďalšou doplniteľnou časťou je návrh vhodného umiestnenia už pri naskladnení produktu, keďže na toto rozšírenie v rámci tejto práce nezostal časový priestor, a rovnako toto rozšírenie nemá ani frontend časť.

Implementovať by taktiež bolo možné ďalšie rozšírenia, ktoré neboli analyzované a navrhnuté v tejto práci. Jedná sa napríklad o súčasný presun

rovnakého typu skladovej položky z viacerých umiestnení alebo detailnejšie zachádzanie s dopĺňaním kusových balení skladových položiek.

Kapitola 4

Testovanie

Testovanie softvéru sa dá popísať ako overovanie, či softvér neobsahuje chyby, spĺňa všetky navrhnuté technické a používateľské požiadavky pri riešení všetkých, no najmä hraničných, prípadov. Testovanie sa nezameriava iba na hľadanie chýb v existujúcom softvéri, ale hľadá aj spôsoby zlepšenia softvéru a jeho efektivity. Testovanie softvéru môžeme rozdeliť na dva druhy, automatické a manuálne. [10, 11]

V tejto kapitole sa pokúsim v krátkosti priblížiť oba druhy testovania, rozdiely medzi nimi, a popíšem, aké nástroje sa na testovanie používajú v rámci projektu skladového systému Atlantis a aké kroky som v rámci oboch druhov testovania podnikla, aby som minimalizovala chyby a maximalizovala efektivitu mojej bakalárskej práce.

4.1 Automatizované testovanie

Automatizácia testov znamená použitie softvéru iného než je ten testovaný, aby vykonal a skontroloval testy a porovnal ich reálne výsledky s tými očakávanými. Je výhodou je možnosť sadu testov opakovať často a rýchlo bez nutnosti testy spúšťať a vyhodnocovať manuálne. [12]

Skladový systém Atlantis používa na pomoc pri automatizovanom testovaní nástroj s návom Gitlab CI/CD a jeho funkciu Automated Testing Pipeline. Táto funkcia sa stará o automatické zostavenie kódu, jeho otestovanie a následné nasadenie. [13, 14] V rámci skladového systému sa zameriam na dva druhy automatizovaných testov, ktoré sú spúšťané a vyhodnocované. Dôvod použitia týchto testov a ich dôsledky a účinky popísal Ing. Pavel Kovář vo svojej diplomovej práci na FIT ČVUT. [1]

Prvou kategóriou sú testy, ktoré zisťujú správnosť formy kódu a kontrolujú, ako kód na prvý pohľad vyzerá. K overeniu kódu sa používa takzvaný

kódovací štandard, ktorý je súborom pravidiel, postupov a techník na vytvorenie čistejšieho, čitateľnejšieho a efektívnejšieho kódu s čo najmenším množstvom chýb. [15] V projekte skladového systému na správnosť kódu dohliada nástroj PHP_CodeSniffer [16], využívajúci kódovací štandard, ktorý vo svojej bakalárskej práci na FIT ČVUT zaviedol Ing. Pavel Kovář. [17]

Druhou kategóriou sú testy, ktoré kontrolujú obsah kódu. V prípade skladového systému Atlantis je využívaný nástroj PHPStan [18], ktorý analyzuje celý kód a na základe pravidiel syntaxe jazyka ho vyhodnotí. Skontroluje viditeľnosť metód a funkcií, počet predávaných parametrov, existenciu premenných a mnoho ďalších nutností, čím zabezpečí, že o potenciálnych chybách užívateľ vie ešte predtým, než kód spustí. [19]

Pri implementácii rozšírení bol teda vždy nový kód hneď po pridaní na Gitlab skladového systému Atlantis podrobený automatizovanému testovaniu a jeho výsledky mi do pár momentov ukázali, čo je na kóde nutné opraviť. Nájdené chyby som vždy opravila a kód znova nahrala na Gitlab projektu, kde bol znova podrobený testom, a v optimálnom prípade tentokrát už ich výsledky súhlasili s tými očakávanými, teda bol kód bez formálnych chýb.

4.2 Manuálne testovanie

Manuálne testovanie znamená testovanie bez použitia akéhokoľvek automatizovaného nástroja alebo akéhokoľvek skriptu. Hlavnou časťou tohoto typu testovania je samotný tester, teda niekto, kto manuálne navrhuje, píše, spúšťa a vyhodnocuje potrebné cesty. Pri testovaní používa testovacie plány, testovacie prípady alebo testovacie scenáre, aby zaručil jeho úplnosť a komplexnosť. Manuálne testovanie zahŕňa aj prieskumné testovanie, keď testeri skúmajú softvér, aby v ňom identifikovali chyby. [12]

Počas implementácie rozšírenia bola väčšina nášho testovania práve manuálna, či už v rámci tímu, alebo samostatne. Často sme svoj napísaný kód odovzdávali inému členovi tímu, aby na ňom skúsil nájsť chyby a nedostatky. Týmto spôsobom bolo nájdených množstvo chýb, ktoré boli inak pre autora kódu takmer neviditeľné, keďže ich neodhalilo automatizované testovanie, ale ovplyvňovali napríklad správnosť systémom vypočítaných hodnôt. Častým problémom bolo taktiež to, že pri oprave chýb detekovaných automatizovanými testami, som z funkčného kódu neúmyselne odmazala jeho časti, čo sa už neprejavilo na druhom kole automatizovaných testov, avšak spôsobilo to chyby práve v postupe algoritmov rozšírenia. V tom prípade nasledovalo kolo manuálneho testovania, ktoré túto chybu bez problémov odhalilo.

V rámci predmetu BI-SP2 som osobne manuálne testovala celý kód, ktorý vznikol, či už svoj, alebo ten od ostatných kolegov v tíme, keďže aj časti, ktorými prispeli oni, boli kľúčové pre vznik tejto bakalárskej práce.

V neskorších fázach implementácie prišiel na radu nástroj Postman. [20] Jedná sa o platformu na vytváranie, používanie a testovanie API. Postman celý tento proces značne zjednodušuje a zefektívňuje. [21] V prípade tejto práce sme využili jeho možnosť vytvárať vlastné HTTP požiadavky a skúmať detailné odpovede na ne.

Záver

Cieľom tejto práce bolo analyzovať aktuálny stav skladového systému Atlantis a navrhnúť a implementovať vhodné optimalizácie častých procesov, ktoré jeho používateľom uľahčia prácu s ním a zároveň im ušetria maximálne možné množstvo času. Prvá časť práce prebiehala v spolupráci s ostatnými členmi tímov z predmetov BI-SP1 a BI-SP2. Po úvodných konzultáciách sme sa rozhodli, že spomínané optimalizácie sa mali týkať najmä procesov naskladnenia a vyskladnenia a samotného ukladania produktov na skladové umiestnenia.

Tento cieľ som v práci splnila. Práca sa najskôr venovala dôkladnej analýze systému, z ktorej vznikol zoznam zmenových požiadaviek, ktoré by systému mohli prospieť. Tie som následne transformovala na funkčné a nefunkčné požiadavky, čo uľahčilo návrh konkrétnych riešení.

Následne sa práca zaoberala presným návrhom riešenia všetkých optimalizácií. V momente, keď bol tento návrh hotový a prekonzultovaný so zadávateľom a vývojovým tímom, postúpila som do realizačnej časti. Úspešnou implementáciou vybraných optimalizácií bol systém rozšírený o podstatné časti, ktoré používateľom zjednodušia jeho používanie.

Všetky implementované rozšírenia boli v poslednej fáze práce dostatočne otestované vhodnou sadou testov, všetky nedostatky, ktoré boli v tejto časti testovaním odhalené, som v priebehu tejto časti promptne opravila.

Výstupom práce je teda rozšírenie skladového systému Atlantis. Navyše realizačná časť obsahuje niekoľko návrhov na ďalšie úpravy, na ktoré v rámci implementácie nebol dostatočný priestor, a teda sa tu vyskytol potenciál pre ďalšie rozšírenie a naviazanie na túto prácu.

Bibliografia

1. KOVÁŘ, Pavel. *Backend skladového systému* [online]. 2019 [cit. 2022-04-01]. Dostupné z : https://dspace.cvut.cz/handle/10467/82591. Dipl. pr. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.
2. MALEC, Oldřich. *Frontend skladového systému* [online]. 2019 [cit. 2022-04-01]. Dostupné z : https://dspace.cvut.cz/handle/10467/86593. Dipl. pr. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.
3. CVRČEK, Jan. *Rozšíření skladového systému Atlantis* [online]. 2021 [cit. 2022-04-01]. Dostupné z : https://dspace.cvut.cz/handle/10467/95049. Bak. pr. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.
4. TALÁR, Denis. *Optimalizace nejčastějších procesů ve skladovém systému* [online]. 2020 [cit. 2022-04-01]. Dostupné z : https://dspace.cvut.cz/handle/10467/88290. Bak. pr. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.
5. GORBACHENKO, Pavel. *What are Functional and Non-Functional Requirements and How to Document These* [online]. 2021 [cit. 2022-04-25]. Dostupné z : https://enkonix.com/blog/functional-requirements-vs-non-functional/.
6. *PHP: Hypertext Preprocessor* [online]. 2022 [cit. 2022-04-20]. Dostupné z : https://www.php.net/.
7. *What is Symfony* [online]. 2022 [cit. 2022-04-20]. Dostupné z : https://symfony.com/what-is-symfony.

8. *The Doctrine Project* [online]. 2022 [cit. 2022-04-20]. Dostupné z : https://www.doctrine-project.org/.
9. *PostgreSQL: About* [online]. 2022 [cit. 2022-04-20]. Dostupné z : https://www.postgresql.org/about/.
10. HAMILTON, Thomas. *What is software testing? definition, basics & types in software engineering* [online]. 2022 [cit. 2022-04-25]. Dostupné z : https://www.guru99.com/software-testing-introduction-importance.html.
11. *What is software testing and how does it work?* [Online]. 2022 [cit. 2022-04-25]. Dostupné z : https://www.ibm.com/topics/software-testing.
12. *Software testing: Basics* [online]. 2021 [cit. 2022-04-25]. Dostupné z : https://www.geeksforgeeks.org/software-testing-basics/.
13. *Gitlab CI/CD* [online]. 2022 [cit. 2022-04-03]. Dostupné z : https://docs.gitlab.com/ee/ci/.
14. GILL, Navdeep Singh. *Automated Testing Pipeline with gitlab CI: Overview* [online]. XenonStack, 2022 [cit. 2022-04-03]. Dostupné z : https://www.xenonstack.com/blog/automated-testing-pipeline-gitlab.
15. *Coding standards and best practices to follow* [online]. 2021 [cit. 2022-04-08]. Dostupné z : https://www.browserstack.com/guide/coding-standards-best-practices.
16. *Squizlabs/php_codesniffer: Php_codesniffer tokenizes PHP files and detects violations of a defined set of coding standards.* [Online]. 2006 [cit. 2022-04-08]. Dostupné z : https://github.com/squizlabs/PHP_CodeSniffer.
17. KOVÁŘ, Pavel. *Automatizované testování webového portálu dbs.fit.cvut.cz* [online]. 2017 [cit. 2022-04-02]. Dostupné z : https://dspace.cvut.cz/handle/10467/69954. Bak. pr. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství.
18. *Phpstan/phpstan: PHP static analysis tool - discover bugs in your code without running it!* [Online]. 2016 [cit. 2022-04-08]. Dostupné z : https://github.com/phpstan/phpstan.
19. *Find bugs in your code without writing tests!* [Online]. 2016 [cit. 2022-04-08]. Dostupné z : https://phpstan.org/blog/find-bugs-in-your-code-without-writing-tests.

20. *Postman API Platform* [online]. 2022 [cit. 2022-04-15]. Dostupné z : https://www.postman.com/.

21. SHARIR, Jacob. *How to use Postman for API Testing Automation* [online]. 2020 [cit. 2022-04-22]. Dostupné z : https://www.blazemeter.com/blog/how-use-postman-manage-and-execute-your-apis.

Dodatok A

Zoznam použitých skratiek

API Application Programming Interface, v preklade rozhranie pre programovanie aplikácií

BI-SP1 Predmet Softwarový týmový projekt 1 vyučovaný v rámci bakalárskeho študijného programu na FIT ČVUT v Praze

BI-SP2 Predmet Softwarový týmový projekt 2 vyučovaný v rámci bakalárskeho študijného programu na FIT ČVUT v Praze

ČVUT České vysoké učení technické

DBAL Database Abstraction Layer

FIT Fakulta informačních technologií

ORM Object Relational Mapper

UI User interface, v preklade užívateľské rozhranie

UX User experience, v preklade užívateľská skúsenosť

Dodatok B

Obsah priloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
  - `be_fe_api` ..... definícia API pre komunikáciu s frontendom
  - `map_api` ..... definícia API pre komunikáciu s mapou
- `text`
  - `thesis.pdf` ..... text práce vo formáte PDF