FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Backend skladového systému
Student:	Bc. Pavel Kovář
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem práce je kompletní tvorba backendu skladového systému. Návrh nového backendu bude zohledňovat (a rozšiřovat) funkcionalitu starého skladového systému Sysel od společnosti Jagu s.r.o. Při návrhu spolupracujte s Bc. Oldřichem Malcem, který bude připravovat klientskou část systému. Zpracujte kompletně minimálně roli skladníka a manažera skladu.

Postupujte dle následujících kroků:
1. Analyzujte všechny případy užití současného řešení a zjistěte, jaké nové funkce by měl systém také podporovat. Při řešení neopomeňte zhodnotit konkurenční řešení.
2. Analyzujte možnosti evidence logistiky - správy zboží připraveného k vyskladnění.
3. Na základě analýzy proveďte vhodný návrh.
4. Návrh zrealizujte v podobě funkčního prototypu.
5. Prototyp podrobte vhodnými Vámi navrženými testy.
6. Na základě testování prototyp upravte.
7. Společně s Bc. Oldřichem Malcem zajistěte vydání alfa verze.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 13. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Backend skladového systému

Bc. Pavel Kovář

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

7. května 2019

Poděkování

V první řadě bych chtěl poděkovat svému vedoucímu, panu Ing. Jiřímu Hunkovi, jednak za příležitost, jenž mi umožnila zvolit si toto téma jako diplomovou práci, ale také za cenné rady, které mi v průběhu tvorby práce poskytl. Zároveň bych chtěl poděkovat Bc. Oldřichu Malcovi, jelikož bez jeho spolupráce by tato práce vůbec nemohla vzniknout. Dále bych chtěl poděkovat také Ing. Janu Matouškovi za poskytnutí informací ohledně současného skladového systému Sysel. Na závěr patří velké díky také mé rodině za jejich podporu během celého mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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 předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 7. května 2019 .....

Odkaz na tuto práci

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

Abstrakt

Tato diplomová práce se zabývá kompletním vývojem nového backendu skladového systému. Nejprve se práce zabývá analýzou již existujících řešení, jenž zahrnuje analýzu současného řešení od společnosti Jagu s. r. o. a dalších konkurenčních řešení. Na základě provedené analýzy je vytvořen vhodný návrh, který je následně implementován v jazyce PHP. Práce se dále zabývá také testováním a automatizovaným nasazením vytvořeného systému do produkčního prostředí. V práci je zmíněn mimo jiné také vývoj několika doplňkových komponent, a to autorizačního serveru s využitím protokolu OAuth 2.0 a mobilní aplikace pro mobilní zařízení Zebra s operačním systémem Android.

Klíčová slova skladový systém, analýza, návrh, OAuth 2.0, PHP

Abstract

This diploma thesis deals with complete process of developing warehouse system backend. Firstly this thesis deals with the analysis of existing solutions, which includes analysis of current solution from Jagu s. r. o. and other competive solutions. Based on this analysis, a suitable design is created, which is subsequently implemented in PHP language. Thesis also deals with testing and automated deployment of the created system into production environment. Work also mentions the development of several additional components, namely authorization server using the OAuth 2.0 framework and mobile application for Zebra mobile devices with Android operating system.

Keywords warehouse system, analyse, design, OAuth 2.0, PHP

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Skladové hospodářství je tu s námi od nepaměti. Již v raných dobách obchodu si každý obchodník vedl záznamy o tom, jaké zboží v průběhu dne prodal a jaké zboží mu na konci dne zůstalo. Díky tomu věděl, jaké zboží musí vyrobit, aby uspokojil budoucí potřeby svých zákazníků. Od počátků obchodu však již uplynula řada let a za tu dobu se změnilo mnohé. Myšlenka skladového hospodářství však zůstala víceméně stále stejná. Snad nejvíce změn, jenž jsme mohli pozorovat, se odehrálo s příchodem počítačů a internetu. Kamenné obchody se začaly vylidňovat a lidé si více a více začali oblibovat nakupování po internetu. Obchodníkům tak přibyla další starost, jíž představuje distribuce zboží k zákazníkům. Naštěstí, právě díky rozmachu počítačů, se objevilo také mnoho různých aplikací od různých společností, které se na problém logistiky a skladového hospodářství specializují. Některé aplikace jsou lepší, některé jsou horší, avšak všechny mají stejný cíl, zjednodušit práci obchodníkům.

Jednou z těchto aplikací je také skladový systém Sysel od společnosti Jagu s. r. o., který tento problém řeší netradičním způsobem. Zatímco mnoho jiných systémů předpokládá, že uživatelé budou pouze zaznamenávat již proběhlé operace, tento systém umožňuje svým uživatelům vytvářet úlohy, které zaznamenávají průběh celé operace. Naneštěstí, během dalšího rozvoje tohoto systému se objevily problémy spojené s jeho rozšiřitelností, a tak vznikla myšlenka vytvořit zcela nový skladový systém.

Cílem této práce bude vytvořit backend pro nový skladový systém, jenž bude vycházet z funkčnosti nabízené skladovým systémem Sysel a bude ji dále rozšiřovat. Nejprve se tato práce bude zabývat analýzou existujících řešení, což bude zahrnovat jednak analýzu skladového systému Sysel, ale také několika dalších vybraných konkurenčních řešení. Na základě této analýzy bude proveden vhodný návrh, jenž bude sloužit jako podklad pro následnou implementaci. V závěru se práce bude zabývat také testováním a možnostmi nasazení nově vytvořeného systému do produkčního prostředí.

Kapitola 1

Analýza

Nedílnou součástí vývojového cyklu jakéhokoli software je bez pochyb analýza problémové domény a sběr požadavků na vývoj nového systému. Ačkoli se některým lidem může zdát, že analýza je jen jedním z mnoha dílčích, stejně náročných, kroků celého vývojového cyklu, opak je pravdou. Kvalita provedené analýzy má významný dopad na další fáze vývojového cyklu. Chybná či neúplná analýza může mít za následek nesprávné pochopení problémové domény, jejímž důsledkem je vytvoření software, který se značně liší od přání a představ zákazníka.

Cílem této kapitoly bude provést důkladnou analýzu problémové domény, jejímž výstupem bude ucelený seznam funkčních a nefunkčních požadavků, které budou základním stavebním kamenem pro návrh nového systému. Jelikož jedním z cílů nového systému je zohlednění funkcionalit současného řešení, bude během analýzy kladen velký důraz právě na analýzu chování současného řešení. Zaměřím se jednak na detailní popis funkčnosti, kterou stávající systém nabízí, ale také na to, jaké uživatelské role mohou se systémem pracovat.

V dalších částech kapitoly se budu věnovat porovnání současného řešení s vybranými konkurenčními řešeními a následné specifikaci požadavků. Během analýzy konkurenčních řešení bude kladen velký důraz na analýzu možností, jenž souvisí s evidencí vyskladněného zboží čekajícího na převzetí přepravní společností.

1.1 Současné řešení

Současné řešení přestavuje skladový systém Sysel od společnosti Jagu s. r. o., jehož první nasazení do reálného provozu se datuje k říjnu roku 2012. Od té doby se na vývoji systému příležitostně pracovalo až do začátku roku 2017, kdy vývoj systému víceméně utichl (dle informací na interním repositáři). Vývoj byl znovu obnoven až v půlce roku 2018, kdy se na vývoji systému začalo opět aktivně pracovat. Naneštěstí během příprav nových funkcí bylo zjištěno, že

jejich implementace do stávajícího řešení bude velice náročná, a tak vznikla myšlenka vytvořit nový skladový systém.

Vraťme se ale zpět k tématu. Skladový systém Sysel je, na rozdíl od tradičních řešení, realizován formou webové aplikace. Díky tomu nejsou uživatelé nuceni používat zařízení s konkrétním operačním systémem, nýbrž mohou systém svobodně používat z jakéhokoli zařízení, jenž disponuje internetovým připojením a webovým prohlížečem. Uživatelům, kterým nevadí vazba na konkrétní zařízení, nabízí společnost Jagu s.r.o. doplňkovou mobilní aplikaci, jejímž cílem je usnadnit práci se skladovým systémem. Funkce této mobilní aplikace budou popsány v podkapitole 1.1.2.

Systém pokrývá běžnou skladovou funkčnost, a navíc nabízí moduly pro účetnictví a pro napojení na dopravce DPD. Tyto doplňkové moduly však budou z následující analýzy vynechány, jelikož jejich analýza není cílem této práce.

V několika následujících podkapitolách bude provedena podrobná analýza skladového systému Sysel, která bude zaměřena především na analýzu chování. Nejprve se ale podívejme na to, jací uživatelé mohou se skladovým systémem pracovat.

1.1.1 Účastníci

Účastník, někdy označován (v závislosti na překladu) jako aktér, popisuje skupinu uživatelů, která disponuje určitou uživatelskou rolí, jenž jim propůjčuje přístup k vybraným částem systému. Současný skladový systém rozlišuje celkem čtyři uživatelské role: *nepřihlášený uživatel*, *skladník*, *vedoucí* a *zákazník*.

Vzhledem k tomu, že nepřihlášený uživatel má, podobně jako ve většině jiných aplikací, právo pouze na přihlášení do systému, nebudu jej dále v analýze více rozebírat.

1.1.1.1 Vedoucí

Vedoucí je osoba, která je zodpovědná za provoz skladu. Systém vedoucím umožňuje spravovat evidence skladů, skladových umístění, zákazníků, zboží a uživatelů. Dále jim nabízí rozhraní pro zadávání a následnou kontrolu úloh, které jim prostřednictvím skladového systému odevzdávají skladníci.

Pro práci se skladovým systémem používají tito uživatelé typicky standardní webový prohlížeč, který mají nainstalovaný na svém počítači.

1.1.1.2 Skladník

Osoba, která po přihlášení získává přístup k rozhraní pro plnění úloh zadaných vedoucím. Mimo plnění úloh, může skladník oznámit příjem dodávky zboží, nebo iniciovat úlohu pro přesun celého umístění.

Skladníci používají při práci, na rozdíl od vedoucího, primárně mobilní zařízení s mobilní aplikací (viz podkapitola 1.1.2), která jim umožňuje rychlejší práci se skladovým systémem.

1.1.1.3 Zákazník

Obsluha osob s rolí zákazník je v současném skladovém systému řešena trochu jiným způsobem, nežli je tomu v případě vedoucího a skladníka. Zatímco tyto osoby mohou pro interakci se skladovým systémem používat grafické rozhraní, osoba s rolí zákazník musí pro interakci využívat výhradně programové aplikační rozhraní (API). Toto rozhraní umožňuje zákazníkovi vytvářet či rušit objednávky, případně mu umožňuje zjistit stav již vytvořené objednávky.

1.1.2 Mobilní aplikace

Jak již bylo několikrát řečeno, současné řešení nabízí také doplňkovou mobilní aplikaci pro snazší práci se skladovým systémem. Tato aplikace je určena pro mobilní zařízení od společnosti Zebra Technologies s operačním systémem Android a nabízí následující klíčovou funkčnost:

- Automatický tisk dokumentů

  Skladový systém nabízí uživatelům mnoho příležitostí ke stažení různých dokumentů, které je často potřebné vytisknout. Zatímco na běžném zařízení by typicky došlo ke stažení souboru, jenž by musel uživatel následně zpracovat, aplikace vše zařídí automaticky. Uživateli tedy stačí kliknout na odkaz pro stažení souboru a aplikace provede potřebné kroky k vytištění dokumentu.

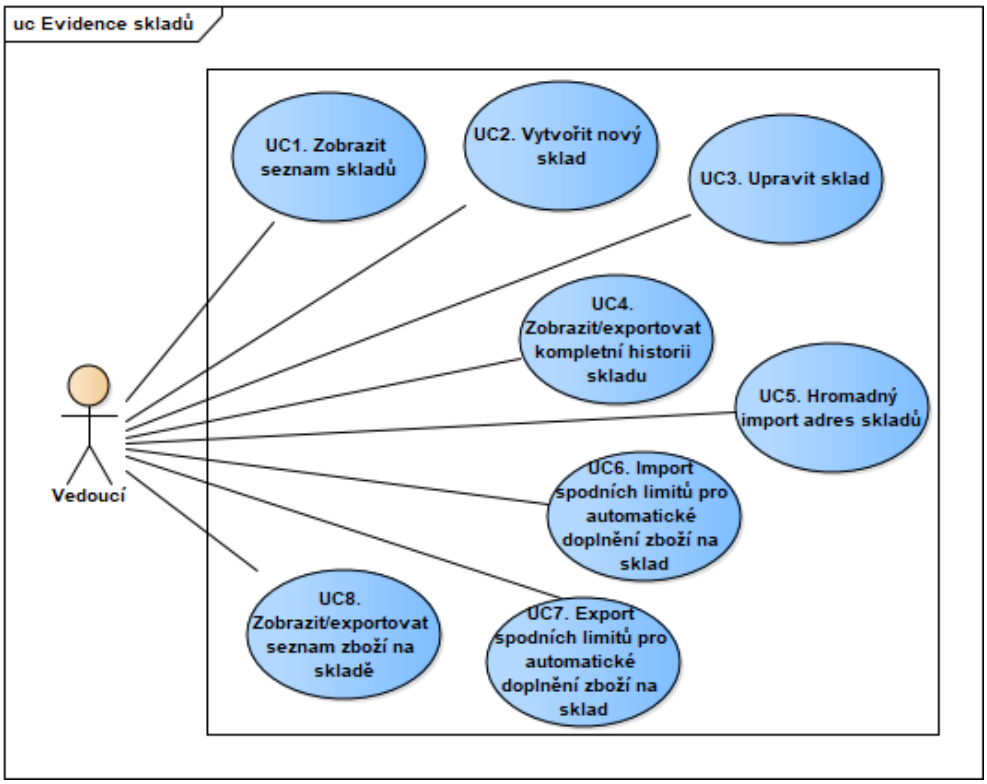
- Podpora čteček čárových kódů

  Aplikace umožňuje práci jednak s integrovanou čtečkou čárových kódů, ale také se čtečkami kódů, jenž jsou k zařízení připojené pomocí technologie Bluetooth. Díky tomu nemusí uživatelé ručně zadávat čárové kódy do příslušných formulářových polí, ale mohou je naskenovat a aplikace je automaticky doplní na příslušná místa.

Přestože se tato aplikace osvědčila v reálném prostředí, pro nově vyvíjený skladový systém ji nebude možné použit. Důvodem je její vysoká provázanost se stávajícím řešením, což ji činí nepoužitelnou v jiných řešeních.

1.1.3 Evidence skladů

Tato podkapitola popisuje funkce související s evidencí skladů. Diagram případů užití je zachycen na obrázku 1.1.



Obrázek 1.1: Diagram případu užití pro evidenci skladů

1.1.3.1 UC1 Zobrazit seznam skladů

Každý vedoucí může zobrazit seznam skladů, které systém eviduje. U každého skladu vidí jeho název, adresu a IČO majitele. Seznam může řadit či filtrovat dle zobrazených informací.

1.1.3.2 UC2 Vytvořit nový sklad

Vedoucí může vytvářet nové sklady. Po kliknutí na tlačítko *Přidat sklad*, nachází se v seznamu skladů, se uživateli zobrazí formulář, kam může zadat následující údaje (všechny jsou volitelné):

- název skladu
- ulice
- město
- PSČ

- stát
- IČO majitele
- DIČ majitele
- číslo bankovního účtu majitele

Pokud formulář potvrdí, je přesměrován na stránku pro úpravu skladu, kde se mu zobrazí hláška informující o úspěšném vytvoření skladu.

1.1.3.3 UC3 Upravit sklad

Uživatel s rolí vedoucí může libovolně upravovat informace o již vytvořených skladech. Uživatel může upravit shodné údaje jako mohl zadat při tvorbě nového skladu, avšak navíc může nastavit telefon a email na majitele skladu.

1.1.3.4 UC4 Zobrazit/exportovat kompletní historii skladu

Vedoucí může zobrazit či exportovat skladové pohyby týkající se konkrétního skladu. Ve výsledném přehledu jsou zobrazeny pouze takové pohyby, které byly vytvořeny úlohou naskladnění, vyskladnění či přeskladnění. Mezi zobrazenými údaji můžeme nalézt:

- číslo úlohy, která vytvořila skladový pohyb
- typ úlohy
- zdrojový a cílový sklad
- model a množství přesunutého zboží
- datum a čas operace

Export tohoto přehledu lze provést pouze do formátu Excelu (tj. XLSX).

1.1.3.5 UC5 Hromadný import adres skladů

Systém umožňuje uživatelům s rolí vedoucí provést hromadnou aktualizaci adres skladů a kontaktních informací na majitele skladů (telefon a email) dle datového souboru ve formátu XLSX či CSV.

Průběh importu je následující:

1. Uživatel odešle na server soubor ve stanoveném formátu.
2. Systém ověří platnost vstupních dat a zobrazí seznam plánovaných změn uživateli. Obsahují-li vstupní data nějaký problém, systém informuje uživatele.

3. Uživatel může změny buď přijmout, nebo odmítnout. Rozhodne-li se změny přijmout, poté systém provede aktualizaci příslušných záznamů. V opačném případě jsou změny zahozeny.

1.1.3.6 UC6 Import spodních limitů pro automatické doplnění zboží na sklad

Uživatelům s rolí vedoucího je umožněno provést hromadné nastavení spodních limitů pro automatické doplnění zboží na sklad. Systém od uživatele očekává vstupní soubor ve formátu CSV či XLSX.

Průběh importu je následující:

1. Uživatel odešle na server soubor, který má následující formát:
   - první řádek obsahuje seznam modelů zboží, pro které bude limit nastaven
   - první sloupec obsahuje seznam skladů, pro které bude limit nastaven
   - obsah vzniklé matice je vyplněn množstvím zboží, jenž má být na skladě udržováno
2. Systém ověří platnost vstupních dat a zobrazí seznam plánovaných změn uživateli. Obsahují-li vstupní data nějakou chybu, systém zobrazí chybovou zprávu.
3. Uživatel může změny buď přijmout, nebo odmítnout. Rozhodne-li se změny přijmout, poté systém provede aktualizaci příslušných záznamů. V opačném případě jsou změny zahozeny.

1.1.3.7 UC7 Export spodních limitů pro automatické doplnění zboží na sklad

Vedoucí může provést export spodních limitů pro automatické doplnění zboží na sklad. Export je možné provést do datových souborů ve formátu CSV a XLSX. Struktura vytvořeného souboru je shodná se strukturou, která se používá pro import těchto limitů – viz UC6.

1.1.3.8 UC8 Zobrazit/exportovat seznam zboží na skladě

Každý uživatel s rolí vedoucí může zobrazit či exportovat přehled zboží na skladě. V přehledu jsou zobrazeny následující údaje:

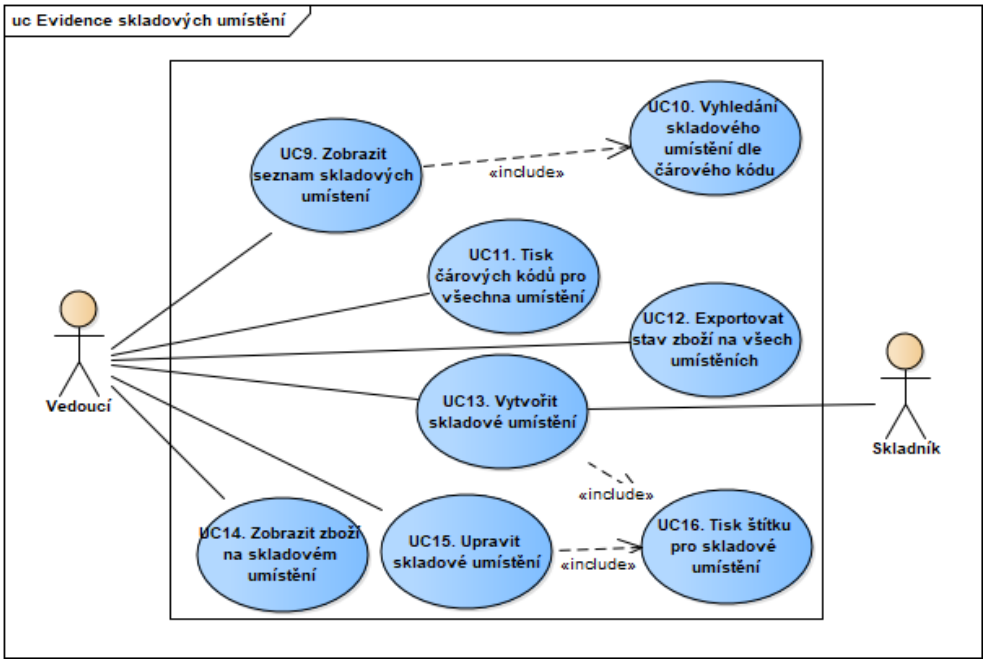
- název zboží
- model zboží
- počet kusů na skladě

- nákupní cena bez DPH

Případně může získat podrobnější exporty, ve kterých je navíc uveden EAN kód zboží, název výrobce či prodejní cena. Export je možné provést do datových souborů ve formátů PDF, XLSX či CSV.

1.1.4 Evidence skladových umístění

V této podkapitole je zachycen popis chování funkcí souvisejících s evidencí skladových umístění. Na obrázku 1.2 je zachycen diagram případů užití.



Obrázek 1.2: Diagram případu užití pro evidenci skladových umístění

1.1.4.1 UC9 Zobrazit seznam skladových umístění

Uživatelé s rolí vedoucího mohou zobrazit seznam všech skladových umístění, jež jsou v systému evidovány. U každého umístění vidí jeho název, stav (schválený / neschválený), název skladu, do kterého umístění patří, a jeho čárový kód. V seznamu mohou vyhledávat či řadit dle všech zobrazených údajů.

Uživatelé mohou využít také rychlého vyhledání skladového umístění dle čárového kódu – viz UC10. Je-li načtený kód v systému evidován, poté je uživatel přesměrován na úpravu konkrétního skladového umístění. V opačném případě je uživateli zobrazena chybová hláška.

1.1.4.2 UC10 Vyhledání skladového umístěné dle čárového kódu

Systém umožňuje rychlé vyhledání skladového umístění dle zadaného čárového kódu. Uživatelé, kteří používají mobilní aplikaci, mohou daný čárový kód načíst pomocí čtečky čárových kódů. Ostatní uživatelé musí čárový kód do systému zadat ručně.

1.1.4.3 UC11 Tisk čárových kódů pro všechna umístění

Uživatelé, kteří jsou ve skladovém systému evidováni jako vedoucí, mohou provést tisk čárových kódů pro všechna evidovaná umístění. Mohou si vybrat z následujících možností:

- 6 kódů na umístění, 1 umístění na stránku
- 4 kódy na umístění, 4 umístění na stránku, 2 sloupce
- 2 kódy na umístění, 8 umístění na stránku, 2 sloupce

Výstupem je soubor ve formátu PDF.

1.1.4.4 UC12 Exportovat stav zboží na všech umístěních

Uživatelé s rolí vedoucí mohou provést export stavu zboží na všech umístěních. Výstupem je datový soubor ve formátu XLSX, kde na prvním řádku je definována struktura tabulky a na zbylých řádcích jsou uvedeny položky přehledu.

Formát tabulky je následující (ve stejném pořadí od prvního sloupce): název umístění, název výrobce, model zboží, název zboží, počet kusů a odkaz na fotografii.

1.1.4.5 UC13 Vytvořit skladové umístění

Uživatelé, kteří mají roli vedoucí nebo skladník, mohou vytvářet nová skladová umístění. Systém uživateli zobrazí formulář, kam může zadat následující údaje:

- název umístění (volitelné)
- sklad, ke kterému umístění patří (volitelné)
- čárový kód (musí být unikátní)

Uživatel může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, zobrazí se hláška informující o úspěšném vytvoření skladového umístění. V opačném případě je přesměrován zpět na původní stránku.

Skladová umístění, vytvořená uživatelem s rolí skladník, jsou označena jako neschválená. Schválit je mohou pouze uživatelé s rolí vedoucí. Během tvorby je uživateli také umožněno vytisknout štítek pro označení umístění.

1.1.4.6 UC14 Zobrazit/exportovat zboží na skladovém umístění

Uživatel s rolí vedoucí může zobrazit nebo exportovat seznam zboží, jenž se nachází na daném skladovém umístění. V přehledu jsou zobrazeny následující informace: model zboží, název zboží a počet kusů. Export je možné provést pouze do souboru ve formátu PDF.

1.1.4.7 UC15 Upravit skladové umístění

Uživatel s rolí vedoucí může libovolně upravovat informace o již vytvořených skladových umístěních. Během úpravy může změnit název umístění, sklad, ke kterému umístění patří, a čárový kód. Případně může vytisknout nové štítky pro označení fyzických skladových umístění.

1.1.4.8 UC16 Tisk štítku pro skladové umístění

Systém umožňuje tisk štítku skladového umístění do datového souboru ve formátu PDF.

1.1.5 Evidence zboží a výrobců

Tato podkapitola se zabývá analýzou funkcí souvisejících s evidencí zboží a výrobců. Diagram případů užití je zachycen na obrázku 1.3.

1.1.5.1 UC17 Zobrazit seznam zboží

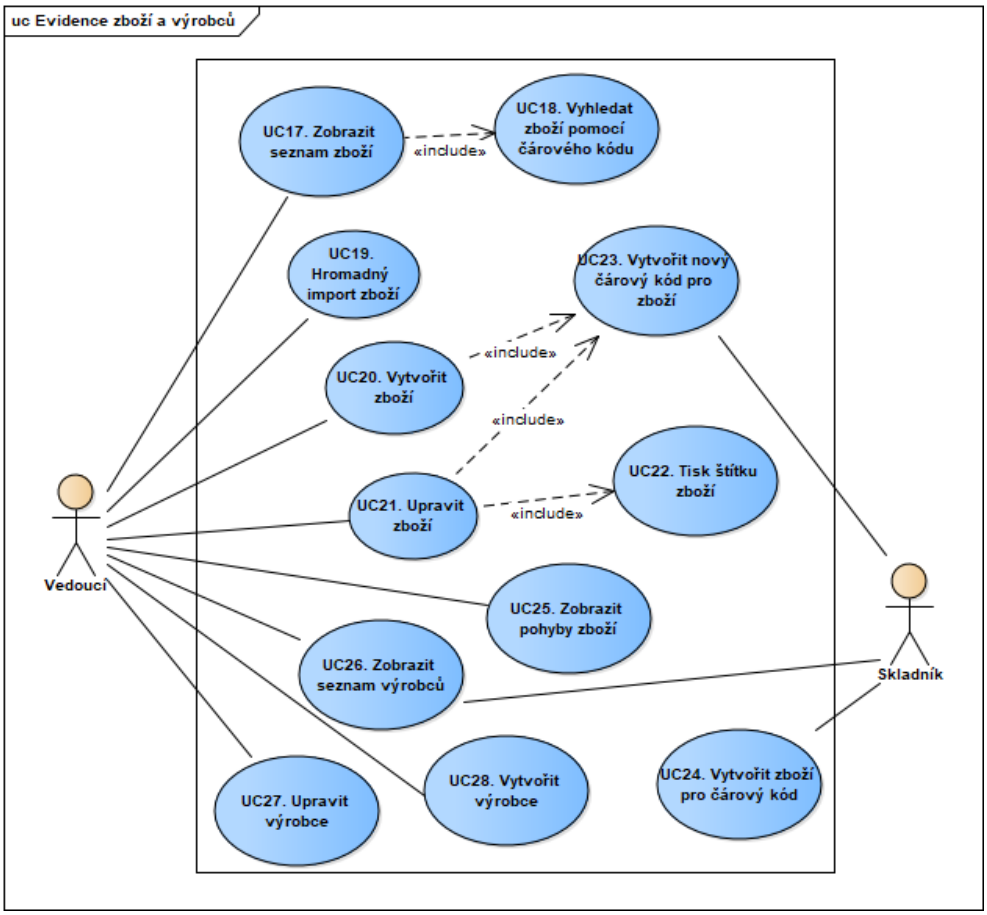
Uživatelé s rolí vedoucí mohou zobrazit seznam zboží, jenž se v systému nachází. U každého záznamu vidí:

- jméno výrobce
- název a model zboží
- počet kusů skladem (souhrn za všechny sklady)
- doporučenou maloobchodní cenu
- nákupní cenu

Seznam mohou dle libosti filtrovat dle výrobce, názvu a modelu zboží. Případně mohou seznam seřadit, mimo skladovosti, dle všech zobrazených údajů.

1.1.5.2 UC18 Vyhledání zboží dle čárového kódu

Systém umožňuje rychlé vyhledání zboží dle zadaného čárového kódu. Uživatelé, kteří používají mobilní aplikaci, mohou daný čárový kód načíst pomocí čtečky čárových kódů. Ostatní uživatelé musí čárový kód do systému zadat ručně.



Obrázek 1.3: Diagram případu užití pro evidenci zboží a výrobců

1.1.5.3 UC19 Hromadný import zboží

Uživatelé s rolí vedoucí mohou provést hromadný import zboží do systému. Systém od uživatele očekává vstupní soubor ve formátu CSV, který splňuje určenou vnitřní strukturu. Po nahrání vstupního souboru, systém ihned provede import obdržených dat. Obsahuje-li vstupní soubor nevalidní data, zobrazí systém uživateli chybové hlášení a import se přeruší.

Struktura vstupního souboru je následující (data jsou oddělena středníkem):

- název zboží
- model zboží
- popis zboží
- čárový kód pro jeden kus

- výrobce
- zkratka výrobce
- prodejní cena bez DPH
- sazba DPH
- nákupní cena bez DPH

1.1.5.4 UC20 Vytvořit zboží

Vedoucí může vytvářet nové zboží. Systém uživateli zobrazí formulář, kam může zadat následující údaje (všechny jsou, mimo výrobce, volitelné):

- název
- model
- výrobce
- popis
- prodejní cenu bez DPH
- nákupní cenu bez DPH
- sazbu DPH
- stav (aktivní / neaktivní)

Dále může libovolně přidávat čárové kód ke zboží či je od zboží odebírat– viz UC23.

Uživatel může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, zobrazí se hláška informující o úspěšném vytvoření zboží a proběhne přesměrování na stránku pro tvorbu nového zboží.

1.1.5.5 UC21 Upravit zboží

Uživatel s rolí vedoucí může libovolně upravovat informace o již vytvořeném zboží. Změnit může shodné údaje jako mohl zadat při tvorbě nového zboží.

1.1.5.6 UC22 Tisk štítku zboží

Systém umožňuje tisk štítku zboží do datových souborů ve formátu PDF a EPL.

1.1.5.7 UC23 Vytvořit nový čárový kód pro zboží

Systém umožňuje přiřadit jednomu zboží více čárových kódů. Uživatelé mohou tyto čárové kódy nastavovat během tvorby či úpravy zboží.

Každý čárový kód pro zboží se skládá ze dvou informací, a to unikátního čárového kódu a počtu kusů. Tento počet kusů ovšem neříká nic o tom, kolik štítků má být vytištěno, nýbrž říká, kolik kusů zboží bude naskenováno při přečtení daného čárového kódu.

Během tvorby nových čárových kódů může uživatel libovolně tisknout nové štítky pro polep zboží – viz UC22.

1.1.5.8 UC24 Vytvořit zboží pro čárový kód

V okamžiku, kdy uživatel s rolí skladník naskenuje neznámý čárový kód, umožňuje systém uživateli buď přiřadit načtený kód k existujícímu zboží (viz UC23), nebo vytvořit nové zboží, které tento kód získá.

Během tvorby nového zboží je uživateli zobrazen formulář, kam může zadat následující údaje:

- název
- model
- výrobce
- popis
- fotografii zboží
- počet kusů (kolik kusů bude naskenováno při přečtení štítku)

1.1.5.9 UC25 Zobrazit pohyby zboží

Vedoucí může zobrazit veškeré skladové pohyby týkající se konkrétní skladové karty zboží. Mezi zobrazenými údaji může nalézt:

- datum a čas operace
- zdrojová a cílová lokace (může být sklad či skladové umístění)
- množství přesunutého zboží
- jméno uživatele, který operaci provedl
- odkaz na úkol, ve kterém byla operace provedena

1.1.5.10 UC26 Zobrazit seznam výrobců

Všichni uživatelé, kteří mají roli skladník nebo vedoucí, mohou zobrazit seznam evidovaných výrobců. U každého výrobce vidí jeho název a zkratku. Seznam mohou filtrovat dle obou údajů.

1.1.5.11 UC27 Vytvořit výrobce

Uživatelé s rolí vedoucí mohou vytvářet nové výrobce. Systém uživateli zobrazí formulář, kam může zadat následující údaje:

- název výrobce (volitelné)
- zkratku výrobce (volitelné)

Uživatel může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, je přesměrován na úpravu výrobce, kde se mu zobrazí hláška informující jej o úspěšném vytvoření výrobce. V opačném případě je přesměrován zpět na původní stránku.

1.1.5.12 UC28 Upravit výrobce

Uživatel s rolí vedoucí může libovolně upravovat informace o již vytvořených výrobcích. Upravit může shodné parametry jako mohl zadat při tvorbě nového výrobce, avšak navíc může nastavovat cenové úrovně.

Každá cenová úroveň má své jméno, koeficient, který říká o kolik procent bude cena navýšena, a druh ceny, ze které bude konečná cena během vyskladnění vypočítána. Vybrat lze z nákupní či doporučené maloobchodní ceny.

1.1.6 Evidence uživatelů

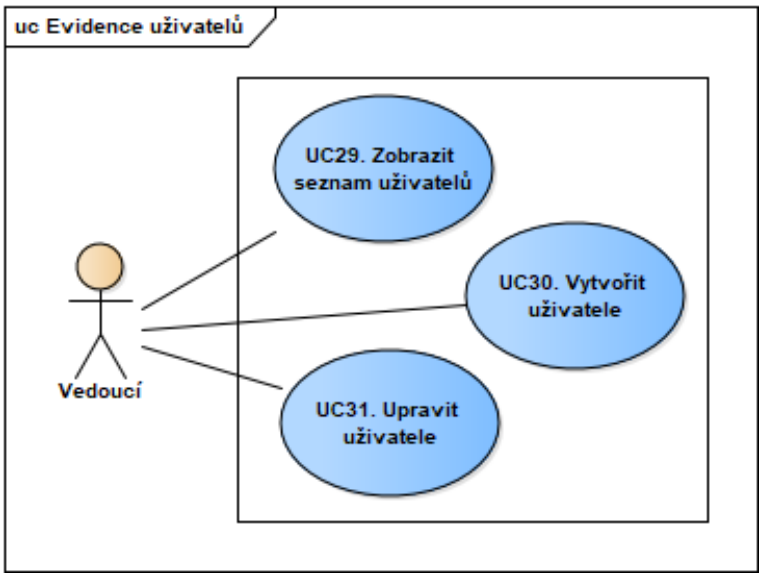
Podkapitola obsahuje popis funkčností souvisejících s evidencí uživatelů. Diagram případů užití je zachycen na obrázku 1.4.

1.1.6.1 UC29 Zobrazit seznam uživatelů

Každý vedoucí může zobrazit seznam uživatelů evidovaných skladovým systémem. U každého uživatele vidí jeho uživatelské jméno, jméno, příjmení a uživatelskou roli. Zobrazený seznam může filtrovat dle všech zobrazených údajů.

1.1.6.2 UC30 Vytvořit uživatele

Uživatel s rolí vedoucí může vytvářet nové uživatele. Systém uživateli zobrazí formulář, kam může vyplnit následující údaje:



Obrázek 1.4: Diagram případu užití pro evidenci uživatelů

- uživatelské jméno (unikátní)
- jméno a příjmení uživatele
- uživatelskou roli (vedoucí nebo skladník)
- heslo a heslo pro kontrolu

Uživatel může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, je přesměrován na stránku pro tvorbu nového uživatele, kde se zobrazí hláška informující jej o úspěšném vytvoření uživatele.

1.1.6.3 UC31 Upravit uživatele

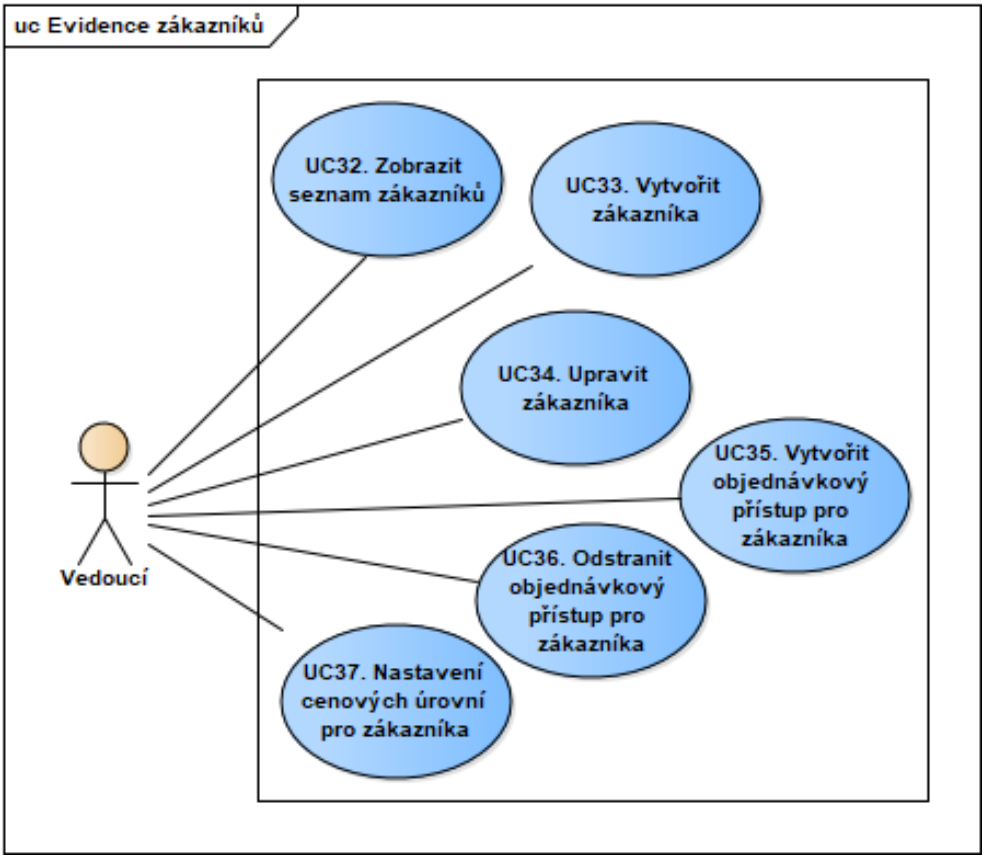
Uživatel s rolí vedoucí může libovolně upravovat informace o již vytvořených uživatelích. Změnit může všechny údaje jako mohl zadat při tvorbě nového uživatele.

1.1.7 Evidence zákazníků

V této podkapitole se budu věnovat analýze funkcí souvisejících s evidencí zákazníků. Diagram případů užití je možné nalézt na obrázku 1.5.

1.1.7.1 UC32 Zobrazit seznam zákazníků

Uživatelé s rolí vedoucí mohou zobrazit seznam všech evidovaných zákazníků. Mezi zobrazenými informacemi mohou nalézt:



Obrázek 1.5: Diagram případu užití pro evidenci zákazníků

- jméno a příjmení zákazníka
- dodací adresu
- IČO
- štítek
- prioritu adresy

Zobrazenou tabulku mohou filtrovat a řadit dle všech zobrazených údajů.

1.1.7.2 UC33 Vytvořit zákazníka

Vedoucí může vytvářet nové zákazníky. Systém zobrazí uživateli formulář, kam může vyplnit následující údaje: jméno zákazníka, štítek a lhůtu splatnosti. Dále může k zákazníkovi přiřadit jednu či více adres, které mohou obsahovat následující údaje:

- adresáta
- ulici
- PSČ
- město
- stát
- telefon
- email
- IČO a DIČ
- prioritu

Vedoucí může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, zobrazí se hláška informující jej o úspěšném vytvoření zákazníka a dále je přesměrován na stránku, kde může upravovat nově vytvořený záznam.

1.1.7.3 UC34 Upravit zákazníka

Uživatel s rolí vedoucí může upravit informace o libovolném zákazníkovi. Během úprav může změnit všechny údaje, které mohl zadat během tvorby.

1.1.7.4 UC35 Vytvořit objednávkový přístup pro zákazníka

Vedoucí může pro každého zákazníka vytvořit nový aplikační přístup, který umožní vstup do správy objednávek. Během vytváření aplikačního přístupu určí, ve kterém skladu budou objednávky vytvářeny, jaká bude dodací adresa (pokud jich má zákazník více) a případně může omezit odběr zboží na vybrané výrobce. Potvrzením formuláře dochází k vytvoření nové přístupové URL adresy, jenž bude zákazník používat pro zasílání objednávek.

1.1.7.5 UC36 Odstranit objednávkový přístup pro zákazníka

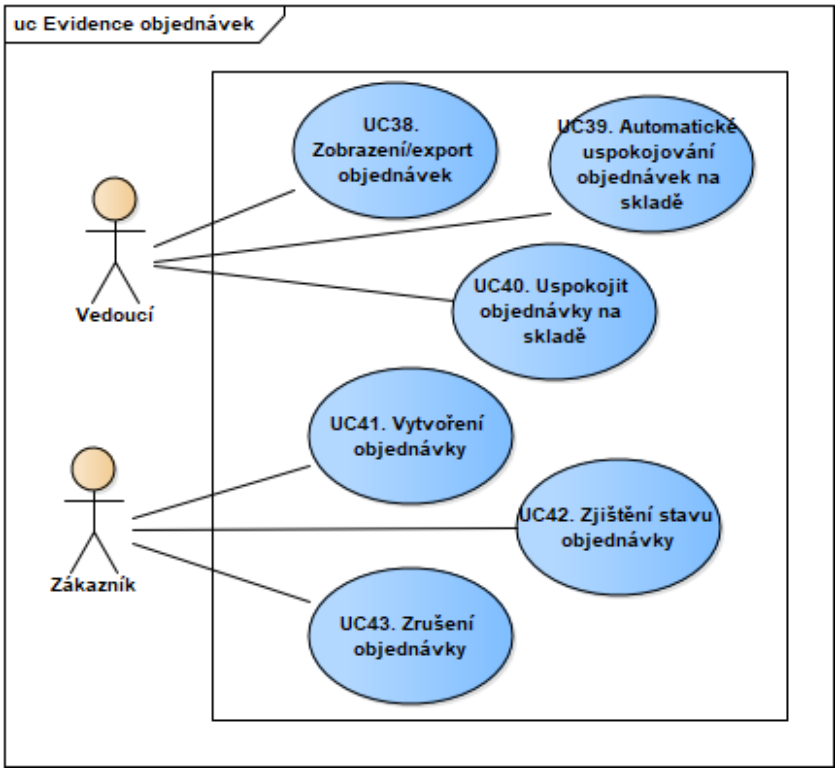
Vedoucí může zákazníkovi zrušit aplikační přístup pro tvorbu nových objednávek.

1.1.7.6 UC37 Nastavení cenových úrovní pro zákazníka

Uživatelé s rolí vedoucí mohou každému zákazníkovi nastavit cenovou úroveň u zboží konkrétního výrobce. Vybírat mohou z předem definovaných cenových úrovní, jež vytvořili během úprav výrobce – viz UC28.

1.1.8 Evidence objednávek

Tato podkapitola se zabývá popisem chování systému souvisejícím s evidencí objednávek. Diagram případů užití je zachycen na obrázku 1.6.



Obrázek 1.6: Diagram případu užití pro evidenci objednávek

1.1.8.1 UC38 Zobrazení/export objednávek

Každý uživatel s rolí vedoucí může zobrazit či exportovat přehled objednávek na skladě. V přehledu jsou zobrazeny následující údaje:

- zkratka výrobce zboží
- název a model zboží
- dodací adresa zákazníka
- komentář
- datum vytvoření
- počet objednaných kusů

Dále může data stáhnout ve strojově čitelném formátu XLSX.

1.1.8.2 UC39 Automatické uspokojování objednávek na skladě

Uživatel s rolí vedoucí může u každého skladu aktivovat či deaktivovat automatické vyřizovaní objednávek. Bude-li tato funkce aktivní, poté budou veškeré objednávky automaticky vyřešeny, jakmile se na skladě objeví dostatečné množství kusů objednaného zboží. V opačném případě musí uživatel vyřizovat příchozí objednávky manuálně – viz UC40.

1.1.8.3 UC40 Uspokojit objednávky na skladě

Každý uživatel s rolí vedoucí může u skladů, které nemají aktivní automatické uspokojování objednávek, manuálně přikázat systému, aby se pokusil vyřídit co nejvíce čekajících objednávek na zboží. Najde-li systém na skladě dostatečné množství objednaného zboží, pokusí se objednávku vyřešit. V opačném případě systém informuje uživatele, že nebylo možné uspokojit žádnou objednávku.

1.1.8.4 UC41 Vytvoření objednávky

Vytvářet objednávky může každý zákazník, který má přidělený objednávkový přístup (viz tvorba přístupu – UC35). Vytvořit objednávku může tak, že odešle POST požadavek na přidělenou adresu, kde obsah požadavku musí mít následující strukturu:

```
{
  "action": "order",
  "data": {
    "items": [
      {
        "ean": "<carovy kod zbozi>",
        "qty": "<mnozstvi objednaneho zbozi>"
      }
    ],
    "customer": {
      "name": "<jmeno zakaznika>",
      "street": "<ulice>",
      "postalCode": "<psc>",
      "country": "<kod zeme>",
      "phone": "<telefon>",
      "email": "<email>"
    }
  }
}
```

V případě úspěchu je výstupem požadavku JSON objekt obsahující číslo vytvořené objednávky *orderId* a identifikátor adresy zákazníka *directoryId*.

1.1.8.5 UC42 Zjištění stavu objednávky

U vytvořených objednávek může zákazník zjišťovat jejich stav. Učinit tak může odesláním požadavku s metodou POST na přidělenou adresu objednávkového vstupu, jehož obsahem bude následující JSON struktura:

```
{
  "action": "state",
  "data": {
    "orderId": <cislo objednavky>,
    "directoryId": <identifikator adresy zakaznika>
  }
}
```

Je-li požadavek úspěšný, poté je výstupem požadavku JSON struktura, která obsahuje kolekci objednaného zboží, kde u každého zboží je uveden současný stav vyřešení.

1.1.8.6 UC43 Zrušení objednávky

Systém umožňuje zákazníkům zrušit dosud nevyřešené objednávky. Rozhodne-li se zákazník objednávku zrušit, poté musí odeslat POST požadavek na adresu objednávkového vstupu, kde obsahem požadavku bude následující JSON struktura:

```
{
  "action": "cancel",
  "data": {
    "orderId": <cislo objednavky>,
    "directoryId": <identifikator adresy zakaznika>
  }
}
```

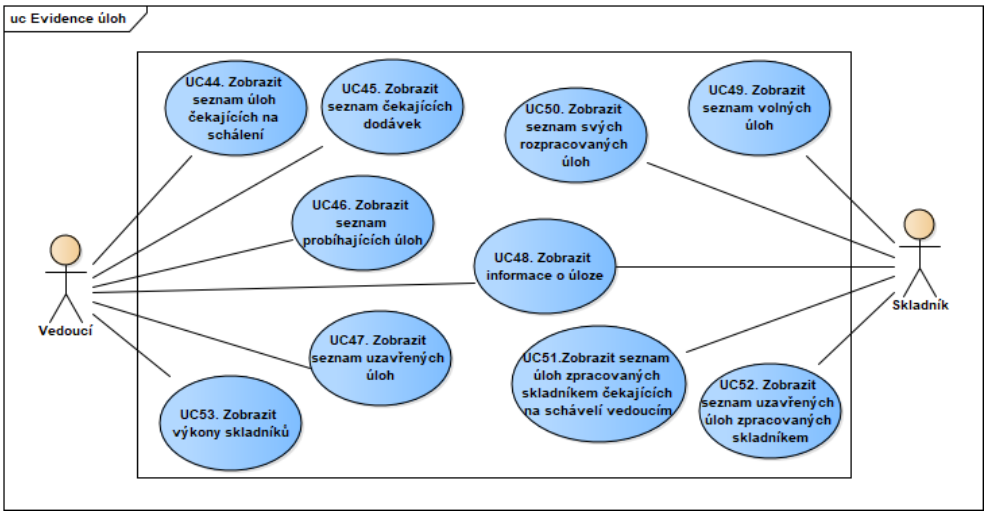
Je-li objednávka úspěšně zrušena, API vrátí kód 200 (OK). V opačném případě API vrátí kód 400 (Bad request).

1.1.9 Evidence úloh

Tato podkapitola se zabývá analýzou přehledových funkcí spojených s evidencí úloh. Popis funkcí spojených s tvorbou a řešením úloh bude popsán v následující podkapitole. Diagram případů užití je zachycen na obrázku 1.7.

1.1.9.1 UC44 Zobrazit seznam úloh čekajících na schválení

Uživatelé s rolí vedoucí vidí na hlavní stránce seznam úloh, které čekají na schválení. Mezi zobrazenými údaji mohou nalézt: číslo a typ úlohy, popis, jméno skladníka, který úlohu zpracoval, a datum a čas dokončení. Kliknutím



Obrázek 1.7: Diagram případu užití pro evidenci úloh

na úlohu dochází k přesměrování uživatele na stránku, kde může úlohu schválit či odmítnout.

1.1.9.2 UC45 Zobrazit seznam čekajících dodávek

Každý vedoucí vidí na své úvodní stránce seznam dodávek zboží čekajících na schválení. U každé dodávky je zobrazeno její číslo, datum a čas přijetí a jméno skladníka, který dodávku oznámil. Po kliknutí na některý čekající záznam dochází k přesměrování uživatele na stránku, kde může danou dodávku zboží zpracovat.

1.1.9.3 UC46 Zobrazit seznam probíhajících úloh

Uživatelé s rolí vedoucí vidí na své úvodní stránce seznam právě řešených úloh. Mezi zobrazenými údaji mohou nalézt: číslo a typ úlohy, popis, jméno skladníka, který úlohu zpracovává, a datum a čas poslední změny. Klikne-li uživatel na některou ze zobrazených úloh, je přesměrován na stránku, kde může nalézt podrobnější informace o dané úloze.

1.1.9.4 UC47 Zobrazit seznam uzavřených úloh

Uživatelé s rolí vedoucí mohou zobrazit seznam uzavřených úloh. Mezi zobrazenými údaji mohou nalézt: číslo a typ úlohy, popis, poznámku, jméno skladníka, který úlohu zpracoval, a datum a čas uzavření. Seznam úloh mohou uživatelé libovolně filtrovat či řadit dle zobrazených údajů.

Klikne-li uživatel na některou ze zobrazených úloh, je přesměrován na stránku, kde může nalézt podrobnější informace o úloze.

1.1.9.5 UC48 Zobrazit informace o úloze

Uživatelé, kteří mají roli vedoucí nebo skladník, mohou v systému zobrazit podrobné informace o úloze. Zatímco vedoucí může zobrazit informace o každé úloze, skladník může zobrazit podrobné informace pouze o jím zpracované uzavřené úloze. Detailní náhled úlohy je skladníkovi zobrazen také v průběhu plnění úlohy, avšak zde jsou zobrazeny pouze podstatné informace, které potřebuje znát pro úspěšné splnění úlohy.

Informace zobrazené v náhledu nejsou pro každou úlohu stejné, nýbrž se liší dle typu úlohy. Nejčastěji zde uživatel může nalézt čeho se úloha týká, s jakým zbožím bylo manipulováno, prioritu úlohy a případně nějaké další doplňující informace.

U úloh naskladnění a vyskladnění může uživatel stáhnout příjemku nebo výdejku ve formátu PDF. Případně může stáhnout souhrn zboží, se kterým bylo v úloze manipulováno, a to ve strojově čitelném formátu CSV a XLSX.

1.1.9.6 UC49 Zobrazit seznam volných úloh

Uživatelé s rolí skladník vidí na své úvodní stránce seznam úloh, které si mohou vzít ke zpracování. Mezi zobrazenými údaji mohou nalézt: číslo úlohy, typ úlohy, popis a prioritu.

Klikne-li uživatel na některou úlohu v seznamu, dochází k přiřazení úlohy uživateli a uživatel je přesměrován na stránku, kde může úlohu zpracovat.

1.1.9.7 UC50 Zobrazit seznam svých rozpracovaných úloh

Uživatelé s rolí skladník vidí na své úvodní stránce seznam jimi rozpracovaných úloh. Mezi zobrazenými údaji mohou nalézt číslo úlohy, typ úlohy, popis a prioritu.

Po kliknutí na některou z rozpracovaných úloh dochází k přesměrování uživatele na stránku, kde může v úloze pokračovat.

1.1.9.8 UC51 Zobrazit seznam úloh zpracovaných skladníkem čekajících na schválení vedoucím

Každý skladník vidí na své hlavní stránce seznam jím zpracovávaných úloh čekajících na schválení vedoucím. U každé úlohy vidí její identifikační číslo, typ, popis a prioritu. U úloh naskladnění a vyskladnění navíc vidí odkaz pro stažení příjemky či výdejky ve formátu PDF.

1.1.9.9 UC52 Zobrazit seznam uzavřených úloh zpracovaných skladníkem

Uživatelé s rolí skladník mohou zobrazit seznam uzavřených úloh, které sami zpracovali, Mezi zobrazenými údaji mohou nalézt: číslo a typ úlohy, popis, poznámku a datum a čas uzavření. Seznam úloh mohou libovolně filtrovat či řadit dle zobrazených údajů.

Klikne-li uživatel na některou ze zobrazených úloh, je přesměrován na stránku, kde může nalézt podrobnější informace o úloze.

1.1.9.10 UC53 Zobrazit výkony skladníků

Každý uživatel s rolí vedoucí může v systému zobrazit přehledovou tabulku, ve které může nalézt výkony skladníků za poslední tři dny. U každého skladníka vidí, kolik úloh daný skladník zpracoval, kolik druhů zboží zpracoval a kolik kusů zboží celkem mu prošlo rukama.

Klikne-li vedoucí na jméno skladníka, zobrazí se podrobný přehled, ve kterém jsou zachyceny shodné informace, avšak seskupené po jedné hodině.

1.1.10 Tvorba a řešení úloh

Tato podkapitola se věnuje analýze chování programu při tvorbě a řešení úloh. Diagram případů užití je zachycen na obrázku 1.8.

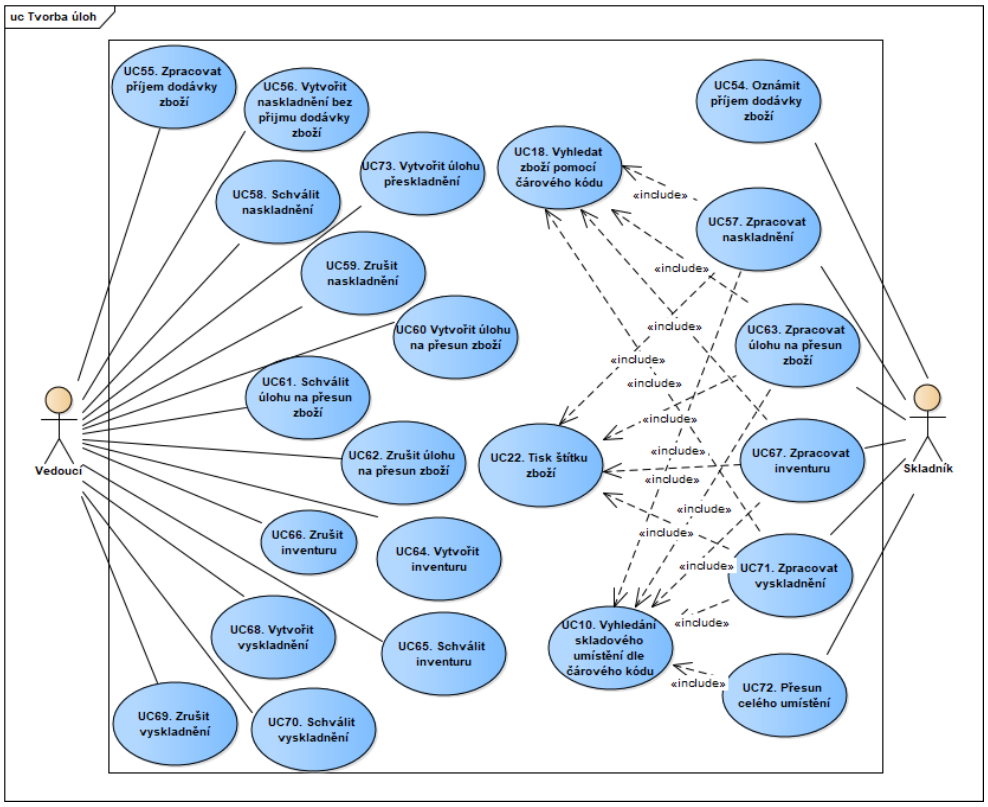
1.1.10.1 UC54 Oznámit příjem dodávky zboží

Každý uživatel s rolí skladník může vytvořit úlohu příjem dodávky. Aby mohl úlohu vytvořit, musí nejprve vyplnit formulář, který mu systém zobrazí. V tomto formuláři může vyplnit následující údaje (vše je volitelné):

- číslo dodacího listu
- číslo faktury
- dodavatele (vybrat může ze seznamu zákazníků)
- poznámku

K formuláři může připojit také fotografii dodacího listu.

Uživatel může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, zobrazí se hláška informující o úspěšném vytvoření úlohy a dochází k přesměrování na úvodní stránku. V opačném případě dochází k přesměrování zpět na předchozí stránku.



Obrázek 1.8: Diagram případu užití pro tvorbu a řešení úloh

1.1.10.2 UC55 Zpracovat příjem dodávky zboží

Vedoucí může úlohu příjem dodávky buď zrušit, nebo schválit.

Rozhodne-li se úlohu zrušit, systém označí úlohu jako zrušenou a uživatele přesměruje na úvodní stránku.

Bude-li chtít úlohu schválit, musí nejprve vyplnit formulář, na základě kterého vznikne nová úloha naskladnění. Ve zobrazeném formuláři může vyplnit následující údaje:

- sklad, pro který bude úloha naskladnění vytvořena
- poznámky k řešení
- preferenci k volbě skladových umístění
- prioritu nové úlohy

Případně může opravit údaje, které do systému vložil skladník během tvorby úlohy. Konkrétně může opravit číslo dodacího listu, číslo faktury nebo dodavatele.

Má-li uživatel formulář vyplněn, může jej potvrdit. Systém následně změní stav úlohy na uzavřený, vytvoří novou volnou úlohu naskladnění a přesměruje uživatele na hlavní stránku.

1.1.10.3 UC56 Vytvořit naskladnění bez přijmu dodávky zboží

Každý uživatel s rolí vedoucí může vytvořit novou úlohu naskladnění, nezávisle na úloze příjem dodávky zboží. Systém zobrazí uživateli formulář, kde může vyplnit následující údaje:

- sklad, pro který bude vytvořena úloha naskladnění
- poznámky k řešení
- preferenci k volbě skladových umístění
- prioritu nové úlohy

Uživatel může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, systém vytvoří novou volnou úlohu naskladnění a přesměruje uživatele na úvodní stránku.

1.1.10.4 UC57 Zpracovat naskladnění

Uživatel s rolí skladník může zpracovat jakékoli volné nebo jemu přiřazené naskladnění. Proces naskladnění je následující:

1. Uživatel zadá, nebo načte čtečkou čárových kódů (UC18), kód zboží. Zadá-li neznámý kód, systém nabídne uživateli vytvoření nového zboží k čárovému kódu (UC24)
2. U načteného zboží vybere skladové umístění z nabízeného seznamu, nebo jej načte čtečkou čárových kódů (UC10).
3. Nastaví celkový počet naskladněných kusů. Opakovaným načítáním zboží se zvyšuje počet kusů o 1.
4. Volitelně může vytisknout štítek pro polep zboží.
5. Chce-li naskenovat další zboží, poté se vrací zpět do bodu 1.
6. Chce-li upravit již naskenované zboží, klikne na zpracované zboží v tabulce a vrací se do bodu 2.
7. Chce-li uživatel úlohu dokončit, klikne na příslušné tlačítko. Dokončením úlohy se změní stav úlohy na čekající na schválení.

Uživatel může úlohu kdykoli přerušit a později se k ní vrátit. Během plnění úlohy vidí uživatel popis naskladnění a poznámky od vedoucího.

1.1.10.5 UC58 Schválit naskladnění

Uživatel s rolí vedoucího může úlohu naskladnění buď schválit, nebo vrátit k přepracováni.

Rozhodne-li se úlohu vrátit k přepracování, musí nejprve určit, zda bude úloha vrácena původnímu skladníkovi, nebo z ní udělá volnou úlohu, kterou může zpracovat jiný skladník. Jakmile zvolí jednu z možností, systém nastaví úloze stav v řešení a přesměruje jej na úvodní stránku.

Dříve než úlohu schválí, může u naskladněného zboží nastavit nákupní cenu. Při schválení označí systém úlohu jako uzavřenou a přesměruje jej na úvodní stránku.

1.1.10.6 UC59 Zrušit naskladnění

Uživatel s rolí vedoucí může zrušit již vytvořenou úlohu naskladnění, avšak pouze za předpokladu, že žádný skladník na úloze ještě nezačal pracovat. Systém takovou úlohu označí jako zrušenou.

1.1.10.7 UC60 Vytvořit úlohu na přesun zboží

V tomto případu užití je zaznamenána tvorba tří typů úloh: přeskladnění na umístění, rozmístění a sestěhování. Tyto úlohy se liší pouze ve specifikaci zdrojového a cílového umístění, a proto není vhodné rozepisovat je samostatně.

Uživatelé s rolí vedoucí mohou vytvářet úlohy typu přeskladění na umístění, rozmístění a sestěhování. Systém uživateli zobrazí formulář, kde musí vyplnit jaké zboží a kolik kusů zboží chce přesunout, prioritu úlohy a skladové umístění, jehož význam se liší dle typu úlohy:

- V případě úlohy přesun na umístění má význam cílového umístění. Vzhledem k tomu, že se tato úloha vytváří na stránce s náhledem umístění, je zdrojové umístění zvoleno automaticky dle otevřené stránky.
- V případě úlohy rozmístění má význam zdrojového umístění.
- V případě úlohy sestěhování má význam cílového umístění.

Uživatel může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, systém vytvoří novou volnou úlohu.

1.1.10.8 UC61 Schválit úlohu na přesun zboží

Uživatel s rolí vedoucí může schválit úlohu typu přeskladnění na umístění, rozmístění a sestěhování. Systém zobrazí uživateli seznam zboží, které skladník během úkolu zpracoval. Je-li s řešením skladníka spokojen, může úlohu schválit. V opačném případě může úlohu vrátit k přepracování.

Rozhodne-li se úlohu vrátit k přepracování, musí nejprve určit, zda bude úloha vrácena původnímu skladníkovi, nebo z ní udělá volnou úlohu, kterou

může zpracovat jiný skladník. Jakmile zvolí jednu z možností, systém nastaví úloze stav v řešení a přesměruje vedoucího na úvodní stránku.

Rozhodne-li se úlohu schválit, systém označí úlohu jako uzavřenou a odkáže jej na úvodní stránku.

1.1.10.9 UC62 Zrušit úlohu na přesun zboží

Uživatel s rolí vedoucí může zrušit již vytvořenou úlohu týkající se přesunu zboží po skladě, avšak pouze za předpokladu, že žádný skladník na úloze ještě nezačal pracovat. Systém takovou úlohu označí jako zrušenou.

1.1.10.10 UC63 Zpracovat úlohu na přesun zboží

V tomto případu užití je zaznamenán proces zpracování tří různých typů úloh. Konkrétně jde o sestěhování, přeskladnění na umístění a rozmístění. Tyto úlohy se liší pouze ve specifikaci zdrojového a cílového umístění, a proto není vhodné rozepisovat je samostatně.

Uživatel s rolí skladník může zpracovat jakoukoli jemu přiřazenou nebo volnou úlohu rozmístění, přeskladnění na umístění či sestěhování. Proces zpracování úlohy je následující:

1. Uživatel zadá, nebo načte čtečkou čárových kódů (UC18), kód zboží.

2. Je-li úloha typu rozmístění, vybere uživatel cílové umístění. Je-li úloha typu sestěhování, vybere uživatel zdrojové umístění. Umístění může buď vybrat z nabízeného seznamu, nebo jej může načíst čtečkou čárových kódů (UC10).

3. Nastaví celkový počet přesouvaných kusů. Opakovaným načítáním zboží se zvyšuje počet kusů o 1.

4. Volitelně může vytisknout štítek pro polep zboží.

5. Chce-li naskenovat další zboží, poté se vrací zpět do bodu 1.

6. Chce-li upravit již naskenované zboží, klikne na zpracované zboží v tabulce a vrací se do bodu 2.

7. Chce-li uživatel úlohu dokončit, klikne na příslušné tlačítko. V okamžiku, kdy není úloha kompletně splněna, je uživatel vyzván k udání důvodu. Po dokončení úlohy se změní stav úlohy na čekající na schválení.

Uživatel může úlohu kdykoli přerušit a později se k ní vrátit. Během plnění úlohy vidí uživatel popis úlohy a poznámky od vedoucího.

1.1.10.11 UC64 Vytvořit inventuru

Systém umožňuje, v náhledu skladového umístění, uživatelům s rolí vedoucí vytvořit novou úlohu inventura. Na stránce mohou uživatelé nalézt formulář, kde vyplní popis a prioritu nové úlohy. Po odeslání formuláře systém vytvoří novou volnou úlohu inventura.

1.1.10.12 UC65 Schválit inventuru

Uživatel s rolí vedoucí může úlohu inventura buď schválit, nebo vrátit k přepracování.

Rozhodne-li se úlohu vrátit k přepracování, musí nejprve určit, zda bude úloha vrácena původnímu skladníkovi, nebo z ní udělá volnou úlohu, kterou může zpracovat jiný skladník. Jakmile zvolí jednu z možností, systém nastaví úloze stav v řešení a přesměruje jej na úvodní stránku.

Rozhodne-li se úlohu schválit, systém označí úlohu jako uzavřenou a odkáže jej na úvodní stránku.

1.1.10.13 UC66 Zrušit inventuru

Uživatel s rolí vedoucí může zrušit již vytvořenou úlohu inventura, avšak pouze za předpokladu, že žádný skladník na úloze ještě nezačal pracovat. Systém takovou úlohu označí jako zrušenou.

1.1.10.14 UC67 Zpracovat inventuru

Uživatel s rolí skladník může zpracovat jakoukoli volnou nebo jemu přiřazenou úlohu inventura. Proces inventury je následující:

1. Uživatel zadá, nebo načte čtečkou čárových kódů (UC18), kód zboží. Zadá-li neznámý kód, systém nabídne uživateli vytvoření nového zboží k čárovému kódu (UC24)
2. U načteného zboží nastaví celkový počet nalezených kusů. Opakovaným načítáním zboží se zvyšuje počet kusů o 1.
3. Volitelně může vytisknout štítek pro polep zboží.
4. Chce-li naskenovat další zboží, poté se vrací zpět do bodu 1.
5. Chce-li upravit již naskenované zboží, klikne na zpracované zboží v tabulce a vrací se do bodu 2.
6. Chce-li uživatel úlohu dokončit, klikne na příslušné tlačítko. Dokončením úlohy se změní stav úlohy na čekající na schválení.

Uživatel může úlohu kdykoli přerušit a později se k ní vrátit. Během plnění úlohy vidí uživatel poznámky od vedoucího.

1.1.10.15 UC68 Vytvořit vyskladnění

Systém umožňuje, v náhledu skladu, uživatelům s rolí vedoucí vytvořit novou úlohu vyskladnění. Na stránce mohou uživatelé nalézt formulář, kam musí vyplnit následující údaje:

- zákazníka a jeho adresu (pokud jich má více)
- cílové umístění, na které bude zboží vyskladněno
- poznámku v dodacím listu
- počet kusů zboží
- poznámky k řešení
- prioritu nové úlohy
- příznak úkolu, který může nabývat jedné z hodnot: k fakturaci, reklamace, nefakturovat a fakturovat mimo systém Sysel

Uživatelé mohou formulář buď potvrdit, nebo opustit. Rozhodnou-li se formulář potvrdit, systém vytvoří novou volnou úlohu vyskladnění.

1.1.10.16 UC69 Zrušit vyskladnění

Uživatel s rolí vedoucí může zrušit již vytvořenou úlohu vyskladnění, avšak pouze za předpokladu, že žádný skladník na úloze ještě nezačal pracovat. Systém takovou úlohu označí jako zrušenou.

1.1.10.17 UC70 Schválit vyskladnění

Každý uživatel s rolí vedoucí může schválit úlohu vyskladnění. Systém zobrazí uživateli seznam zboží, které skladník během úkolu zpracoval. Je-li s řešením skladníka spokojen, může úlohu schválit. V opačném případě může úlohu vrátit k přepracování.

Rozhodne-li se úlohu vrátit k přepracování, musí nejprve určit, zda bude úloha vrácena původnímu skladníkovi, nebo z ní udělá volnou úlohu, kterou může zpracovat jiný skladník. Jakmile zvolí jednu z možností, systém nastaví úloze stav v řešení a přesměruje vedoucího na úvodní stránku.

Dříve než úlohu schválí, může vyplnit poznámku pro zákazníka. Po schválení označí systém úlohu jako uzavřenou a odkáže jej na úvodní stránku.

1.1.10.18 UC71 Zpracovat vyskladnění

Uživatel s rolí skladník může zpracovat jakékoli volné nebo jemu přiřazené vyskladnění. Proces vyskladnění je následující:

1. Uživatel zadá, nebo načte čtečkou čárových kódů (UC18), kód zboží.

2. U načteného zboží zadá zdrojové umístění, které může buď vybrat z nabízeného seznamu, nebo jej může načíst čtečkou čárových kódů (UC10).
3. Nastaví celkový počet kusů. Opakovaným načítáním zboží se zvyšuje počet kusů o 1.
4. Volitelně může vytisknout štítek pro polep zboží.
5. Chce-li naskenovat další zboží, poté se vrací zpět do bodu 1.
6. Chce-li upravit již naskenované zboží, klikne na zpracované zboží v tabulce a vrací se do bodu 2.
7. Chce-li uživatel úlohu dokončit, klikne na příslušné tlačítko. V okamžiku, kdy není úloha kompletně splněna, je uživatel vyzván k udání důvodu. Před dokončením je uživatel vyzván k zadání počtu balíků, ve kterých je vyskladněné zboží zabaleno. Po zadání příslušné hodnoty změní systém stav úlohy na čekající na schválení.

Uživatel může úlohu kdykoli přerušit a později se k ní vrátit. Během plnění úlohy vidí uživatel popis úlohy a poznámky od vedoucího.

1.1.10.19 UC72 Přesun celého umístění

Uživatel s rolí skladník může vytvořit úlohu přesun celého umístění. Systém zobrazí uživateli formulář, kde musí zadat, či načíst čtečkou čárových kódů (UC10), zdrojové a cílové umístění. Po zadání posledního umístění systém zkontroluje, zda nejsou zadaná umístění shodná, zda obě umístění patří do stejného skladu a zda cílové umístění neobsahuje nějaké zboží. Jsou-li všechny podmínky splněny, systém okamžitě provede přesun zboží, úlohu označí jako uzavřenou a přesměruje uživatele na úvodní stránku. V opačném případě je uživateli zobrazeno chybové hlášení.

1.1.10.20 UC73 Vytvořit úlohu přeskladnění

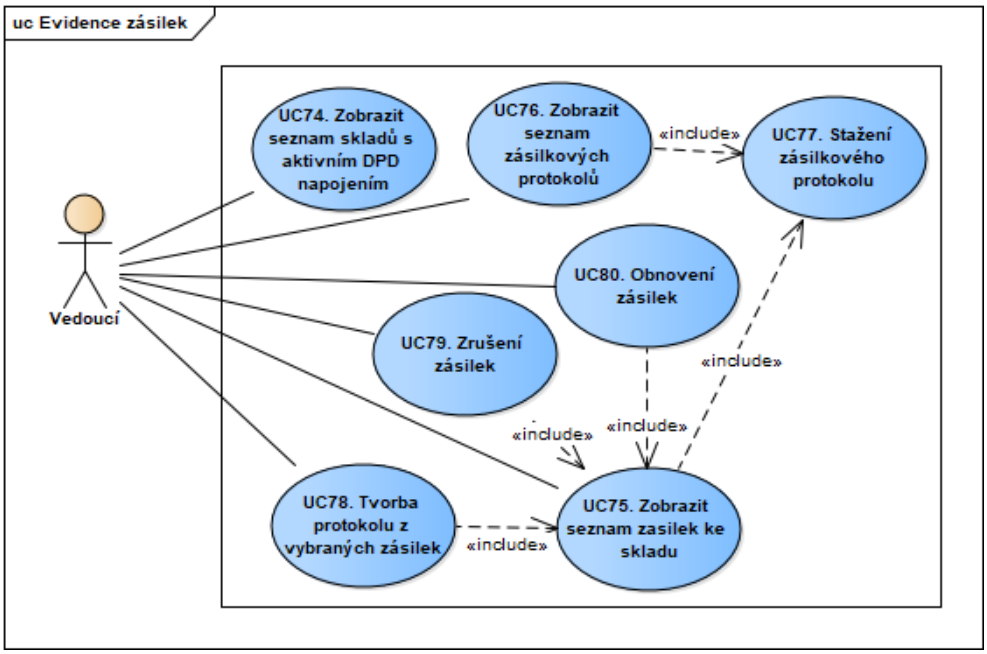
Systém umožňuje, v náhledu skladu, uživatelům s rolí vedoucí vytvořit novou úlohu přeskladnění. Na stránce mohou uživatelé nalézt formulář, kde musí vyplnit následující údaje:

- cílový sklad, kam bude zboží přeskladněno
- počet kusů přesouvaného zboží
- poznámky k řešení
- prioritu nové úlohy
- příznak úkolu, který může nabývat jedné z hodnot: k fakturaci, reklamace, nefakturovat a fakturovat mimo systém Sysel

Uživatel může formulář buď potvrdit, nebo opustit. Rozhodne-li se formulář potvrdit, systém vytvoří novou úlohu přeskladnění, kterou automaticky zpracuje a uzavře.

1.1.11 Evidence zásilek

Tato podkapitola se věnuje analýze funkcí spojených s evidencí zásilek. Nadcházející popis se týká pouze té části systému, která je přístupná uživateli přes webové rozhraní. Diagram případů užití je zachycen na obrázku 1.9.



Obrázek 1.9: Diagram případu užití pro evidenci zásilek

1.1.11.1 UC74 Zobrazit seznam skladů s aktivním DPD napojením

Uživatel s rolí vedoucí může zobrazit seznam skladů, které mají aktivní napojení na dopravce DPD. Mezi zobrazeným údaji může najít: název skladu, ulici, město a stát. Zobrazenou tabulku může dále filtrovat či řadit dle všech zobrazených údajů. Klikne-li na některý zobrazený záznam, je přesměrován na stránku, kde vidí seznam zásilek, jenž pochází z daného skladu.

1.1.11.2 UC75 Zobrazit seznam zásilek ke skladu

Uživatel s rolí vedoucí může zobrazit seznam zásilek, které byly a budou odeslány z konkrétního skladu. V zobrazeném přehledu může najít následující

informace: číslo zásilky přidělené dopravcem, číslo úlohy vyskladnění, jenž zásilku vytvořila, stav, adresáta, datum vytvoření a číslo protokolu. Stav zásilky může nabývat jedné z hodnot: vytvořeno, čeká na svoz, odesláno, doručeno, vráceno, nedoručeno nebo zrušeno. Zobrazenou tabulku může filtrovat či řadit dle všech zobrazených údajů.

Ve zobrazené tabulce vidí uživatel také odkazy, pomocí nichž se může rychle dostat k potřebným informacím. Například klikne-li na číslo zásilky, systém jej přesměruje na stránku dopravce, kde může zásilku sledovat. Podobně klikne-li na číslo úlohy vyskladnění, je přesměrován na stránku, kde může najít podrobné informace o úloze. Dále se zde nacházejí odkazy, které slouží pro stažení štítku zásilky a pro stažení zásilkového protokolu (viz UC77), v němž se zásilka nachází.

1.1.11.3 UC76 Zobrazit seznam zásilkových protokolů

Uživatel s rolí vedoucí může zobrazit seznam protokolů, jenž dříve vytvořil ze zásilek. U každého protokolu vidí jeho číslo, název skladu, ke kterému patří, a datum vytvoření. Zobrazenou tabulku může dle libosti řadit či filtrovat dle všech zobrazených údajů. Klikne-li na některý zobrazený záznam, poté dochází ke stažení příslušného protokolu – viz UC77.

1.1.11.4 UC77 Stažení zásilkového protokolu

Systém umožňuje uživateli s rolí vedoucí stáhnout zásilkový protokol. Výstupem je datový soubor ve formátu PDF, který obsahuje informace o odesílateli a seznam zásilek, jenž se v protokolu nachází.

1.1.11.5 UC78 Tvorba protokolu z vybraných zásilek

Systém umožňuje uživateli s rolí vedoucí vytvořit nový zásilkový protokol z vybraných zásilek. Uživatel může na stránce s přehledem zásilek ke skladu (viz UC75) vybrat zásilky, které chce přiřadit k jednomu zásilkovému protokolu. Následně kliknutím na tlačítko *Vytvořit protokol z vybraných* vytvoří nový zásilkový protokol. Po vytvoření protokolu jsou zásilky označeny jako odeslané.

1.1.11.6 UC79 Zrušení zásilek

Systém umožňuje uživateli s rolí vedoucí zrušit jednu či více zásilek. Uživatel může na stránce s přehledem zásilek ke skladu (viz UC75) vybrat zásilky, které chce zrušit. Následně kliknutím na tlačítko *Zrušit vybrané* tyto zásilky zruší (systém je označí jako zrušené). Zrušena může být pouze zásilka, která se nachází ve stavu *vytvořeno*.

1.1.11.7 UC80 Obnovení zásilek

Systém umožňuje uživateli s rolí vedoucí obnovit již zrušené zásilky. Uživatel může na stránce s přehledem zásilek ke skladu (viz UC75) vybrat zásilky, které chce obnovit. Následně kliknutím na tlačítko *Obnovit vybrané* tyto zásilky obnoví (systém je označí jako vytvořené).

1.2 Konkurenční řešení

V minulé podkapitole jsem se zabýval současným řešením od společnosti Jagu s. r. o. Nyní se zaměřím na vybraná konkurenční řešení, která se zabývají podobnou problémovou doménou, a pokusím se zhodnotit v čem jsou tato řešení lepší. Případně se pokusím identifikovat funkce, jimiž by bylo možné funkčnost nového skladového systému obohatit. Vzhledem k tomu, že jedním z cílů nově vznikajícího skladového systému je univerzálnost, tzn. aby bylo možné systém použit s jakoukoli aplikací, zařadil jsem do porovnání také aplikace, které se nezabývají čistě skladovou funkčností.

1.2.1 Ekonomický a informační systém POHODA

Prvním řešením, kterým se budu zabývat, je ekonomický a informační systém POHODA, vyvíjený českou společností STORMWARE s. r. o. Dle [1] se jedná o *„komplexní systém pro vedení účetních, skladových, majetkových, mzdových a dalších agend a řízení firmy“*. Systém je dle [2] dostupný v sedmi různých variantách, které se liší v nabízené funkčnosti. Díky tomuto rozdělení necílí systém pouze na velké společnosti, které využijí veškeré nabízené funkce, ale také na menší podnikatele, kteří si vystačí s menším množstvím funkcí.

Během analýzy se zaměřím pouze funkce spojené se skladovou agendou, ostatní funkčnost bude z analýzy vynechána.

1.2.1.1 Evidence skladů

Systém umožňuje evidovat větší množství skladů. Tyto sklady dále umožňuje členit do stromové struktury až do hloubky osmi úrovní, čímž ve skladu vznikají podsklady, které mohou přispět k lepší organizaci skladu. U každé fyzické i logické úrovně eviduje systém identifikátor a název skladu.

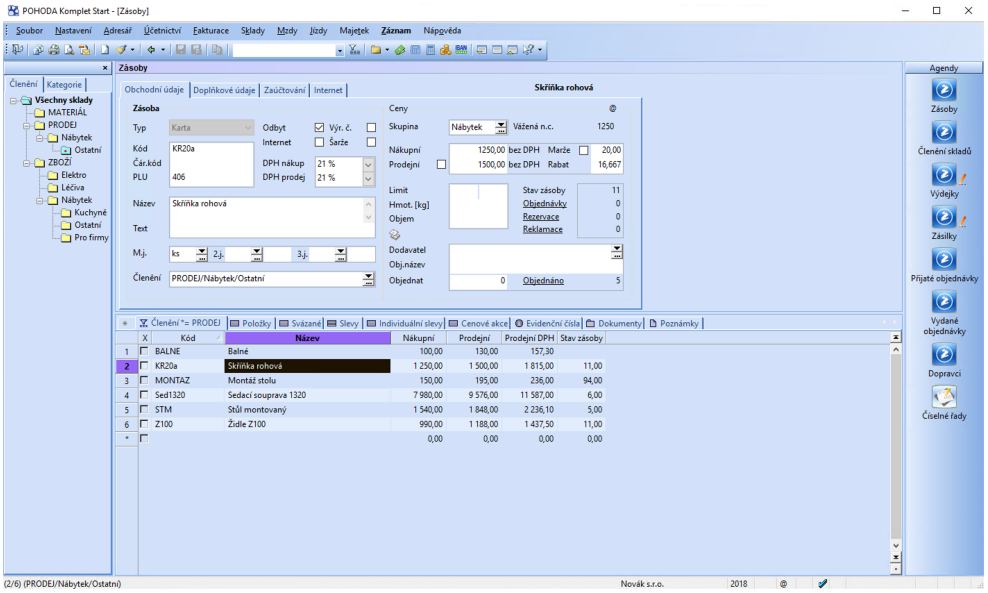
Na rozdíl od stávajícího řešení, tento systém neumožňuje evidovat fyzická skladová umístění, jenž se ve skladu nacházejí. Systém tedy dokáže uživateli říct, jaké zboží ve skladu má, ale už mu neřekne, kde má dané zboží hledat. Absence této informace ovšem vyplývá ze zaměření tohoto systému, jelikož z hlediska účetnictví nás nemusí zajímat, kde máme zboží fyzicky uloženo.

1.2.1.2 Evidence zboží

Na rozdíl od současného řešení, umožňuje systém POHODA rozlišovat zboží s výrobními čísly nebo šaržemi. Díky tomu systém dokáže zobrazit skladové operace, které se týkají pouze konkrétního zboží s daným výrobním číslem nebo šarží. U produktových šarží systém navíc umožňuje evidovat datum expirace, podle kterého se může uživatel rozhodovat během vyskladnění zboží.

Dále systém umožňuje u každého zboží evidovat dodatečné informace, mezi které patří například hmotnost a objem zboží, měrná jednotka, informace o záruce a další. Uživatel může zboží zařadit také do cenové skupiny, podle které systém předpočítá konečnou prodejní cenu. Pro lepší představu čtenáře je formulář pro správu zboží zachycen na obrázku 1.10.

Oproti současnému řešení, neumožňuje účetní systém POHODA evidovat více čárových kódů k jednomu druhu zboží.



Obrázek 1.10: Ukázka ekonomického systému Pohoda: Evidence zásob

1.2.1.3 Evidence dodavatelů a odběratelů

V systému POHODA je tato evidence realizována formou společného adresáře, ve kterém se nacházejí informace o všech firmách či osobách. U každého subjektu umožňuje systém evidovat jeho jméno, fakturační adresu, IČO, DIČ, kontaktní informace, seznam dodacích adres, seznam bankovních účtů, individuální slevy a další informace, které s modulem skladů již tolik nesouvisí. Dále může uživatel u každého subjektu určit, zda se jedná o dodavatele či odběratele, nebo zda jde o kombinaci obou možností.

1.2.1.4 Evidence objednávek

Evidence objednávek je v systému POHODA řešena standardním způsobem. To znamená, že systém u každé objednávky eviduje odběratele, čas vytvoření, požadovaný čas vyřízení, cenovou úroveň, formu platby a seznam objednaného zboží včetně prodejních cen. Dále nabízí uživateli podporu pro částečné uspokojení objednávky a nechybí ani podpora pro tisk souvisejících dokumentů, například potvrzení o přijetí objednávky.

Podobně jako skladový systém Sysel, umožňuje i tento systém tvorbu objednávek přes aplikační programové rozhraní. Avšak oproti současnému řešení je konfigurace takového přístupu značně komplikovanější, což ovšem vyplývá z faktu, že systém POHODA není primárně určen pro webové použití.

1.2.1.5 Evidence úloh

Systém POHODA nabízí následující typy skladových operací: naskladnění, vyskladnění, přeskladnění a inventura. Zásadní rozdíl, oproti současnému řešení, spočívá ve způsobu řešení těchto úloh. Zatímco skladový systém Sysel dělí zpracování úlohy mezi zadavatele a řešitele (skladníka), systém POHODA předpokládá, že celou úlohu zpracuje jedna osoba. Toto chování ovšem opět vyplývá ze zaměření systému POHODA na trochu jinou oblast. Účetní, který se systémem nejčastěji pracuje, pouze zaznamená obdržené údaje do formuláře. O samotné provedení skladové operace se již postarala jiná osoba.

Další zásadní rozdíl, který můžeme mezi systémy zaznamenat, je způsob řešení inventury. Zatímco v systému Sysel má inventura spíše informativní charakter, systém POHODA dokáže objevené rozdíly promítnout (zaúčtovat) do současného stavu skladu. Tuto funkci naneštěstí skladový systém Sysel nenabízí a případné změny je nutné provést manuálně přes úlohu naskladnění či vyskladnění.

Posledního rozdílu, kterého si můžeme při práci se systémem POHODA všimnout, je postup vyplňování informací u jednotlivých položek dokladu. Jakmile vyplníme kód zboží, nebo jej načteme čtečkou čárových kódů, systém nám ihned nabídne pole pro vyplnění nákupní či prodejní ceny. Ve skladovém systému Sysel jsou tato pole zobrazena až v průběhu schválení úlohy.

1.2.1.6 Evidence reklamací

U každé reklamace systém POHODA eviduje: typ (zákaznická / skladová), identifikační číslo, datum přijetí, datum předpokládaného vyřízení, datum skutečného vyřízení, stav, popis, odběratele, kterého se reklamace týká, kontaktní údaje na osobu, která zboží reklamuje, způsob přijetí a převzetí, jméno odpovědné osoby, číslo souvisejícího dokladu (typicky výdejky) a seznam reklamovaného zboží. Každá reklamace se může nacházet v jednom z patnácti stavů, jejichž význam lze shrnout do čtyř následujících možností: *přijetí reklamace*, *předáno do servisu*, *přijato ze servisu* a *předáno zákazníkovi*.

Systém mimo jiné umožňuje také tisk souvisejících dokumentů jako je například potvrzení o příjmu reklamace nebo reklamační protokol.

1.2.1.7 Evidence zásilek

V této podkapitole se podíváme jak sytém POHODA řeší evidenci zásilek – tzn. evidenci vyskladněného zboží čekajícího na vyzvednutí přepravcem.

Předtím než můžeme v systému POHODA vytvořit zásilku, musíme nejprve v agendě dopravců nakonfigurovat jaké způsoby přepravy zboží chceme povolit. Systém umožňuje volbu ze čtyř skutečných dopravců. Konkrétně jde o Geis, PPL, Českou poštu a GLS. Budeme-li chtít zboží přepravit jiným dopravcem, potom musíme zvolit dopravce s názvem *Ostatní*. Jakmile budeme mít nakonfigurované způsoby přepravy, můžeme začít používat agendu zásilek.

Nové zásilky umožňuje systém vytvářet pouze z přijatých objednávek a vydaných faktur. To znamená, že vytvoříme-li pouze výdejku ze skladu, systém nám tvorbu zásilky nepovolí. Ovšem způsob tvorby zásilek pro objednávky mi připadá poněkud zvláštní. Přestože u přijaté objednávky nastavím dopravce a objednávku označím jako vyřešenou, systém zásilku nevytvoří. Abych zásilku vytvořil, musím nejprve kliknout na tlačítko *Vygenerovat zásilky*. Teprve v tu chvíli dochází k vytvoření zásilky, kterou mohu později nalézt v agendě zásilek.

Podíváme-li se do této agendy, zjistíme, že každá zásilka se může nacházet buď ve stavu *k vyřízení*, nebo ve stavu *vyřešeno*. Do stavu vyřešeno se zásilka dostává v okamžiku exportu dat pro dopravce.

1.2.1.8 Shrnutí

Jak jsme viděli, skladový systém, který je součástí ekonomického systému POHODA, nabízí několik zajímavých funkcí, jimiž by bylo možné nový skladový systém obohatit. Dovolím si zde tyto funkce vyjmenovat:

- Evidence skladových čísel a šarží
- Zaúčtování inventury
- Logické členění skladu na podsklady
- Evidence reklamací

Naneštěstí podpora pro evidenci zásilek není, z mého pohledu, v tomto systému příliš dobře vyřešena, jelikož ze systému není možné zjistit, jaké zásilky již přepravce odvezl a jaké zásilky ještě máme na skladě.

1.2.2 CÉZAR G3

„Ekonomický informační systém CÉZAR je specialistou pro firmy zabývající se velkoobchodem, maloobchodem, výrobní, importní i exportní společnosti.

Je výborným pomocníkem pro všechny, kdo chtějí mít přehledně evidované všechny skladové operace.“ Jedná se o český produkt vyvíjený společností Breaker Software. [3]

1.2.2.1 Evidence skladů

Skladový systém CÉZAR podobně jako obě předchozí řešení dokáže současně evidovat více skladů. Ovšem na rozdíl od předchozích řešení může uživatel v jednu chvíli pracovat nejvýše s jedním skladem. Důvodem tohoto omezení je rozdělení skladů do samostatných databází. To znamená, že bude-li chtít uživatel pracovat s jiným skladem, musí do něj nejprve v aplikaci přepnout. Velikou nevýhodou tohoto řešení je skutečnost, že mezi jednotlivými sklady není sdílena ani evidence zboží ani evidence dodavatelů a odběratelů. Uživatel tedy musí tyto informace duplicitně nadefinovat ve všech skladech.

Podobně jako u systému POHODA ani tento systém nedokáže evidovat fyzická skladová umístění. Ovšem na rozdíl od skladového systému Sysel umožňuje logické členění skladu formou podskladů.

1.2.2.2 Evidence zboží

Evidence zboží se oproti skladovému systému Sysel příliš neliší. Systém umožňuje uživateli zaevidovat více doplňkových informací jako je například měrná jednotka, hmotnost, objem, počet kusů v balení a volné údaje. Volný údaj představuje údaj, který uživatel ke zboží zadá a který nepatří do žádného předpřipraveného formulářového pole. Dále může uživatel u každého zboží určit, do kterého podskladu se bude naskladňovat.

1.2.2.3 Evidence dodavatelů a odběratelů

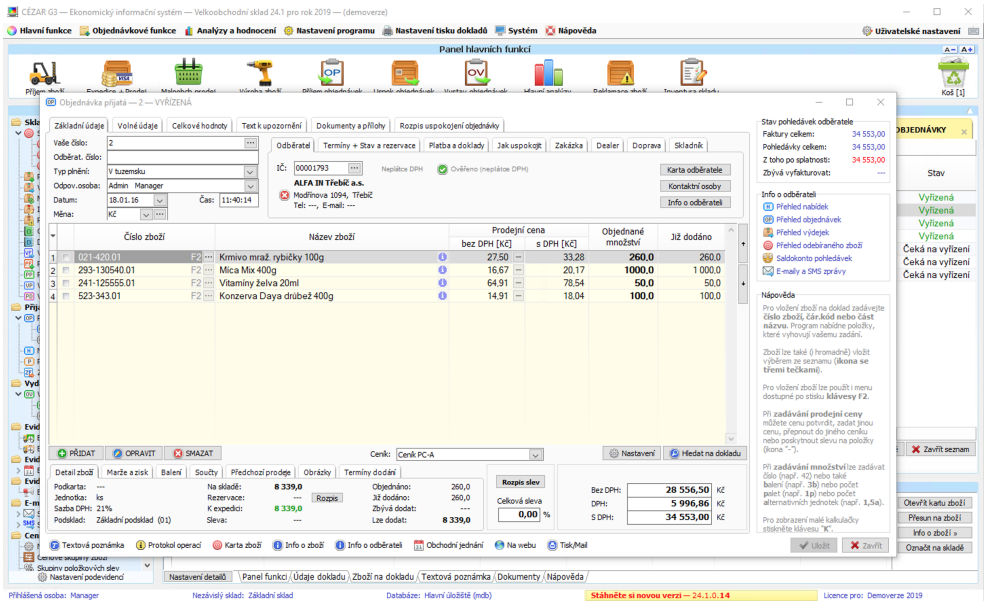
Evidence dodavatelů a odběratelů je víceméně shodná s evidencí, kterou obsahuje skladový systém Sysel. Navíc se zde eviduje bankovní spojení, kontaktní informace (email, telefon, webová stránka atp.) a čárový kód. Zajímavou funkcí, kterou tento systém disponuje, je ověření údajů subjektu vůči registrům veřejné správy.

1.2.2.4 Evidence objednávek

Evidence objednávek je řešena obdobným způsobem jako u předcházejících řešení. U každé objednávky je evidován její odběratel, měna, datum vytvoření, požadovaný termín dodání, předpokládaný termín dodání, stav objednávky, způsob platby, způsob uspokojení objednávky (celá nebo po částech), dopravce a seznam objednaného zboží včetně prodejní ceny. Pro lepší představu čtenáře je příslušný formulář zachycen na obrázku 1.11.

Na rozdíl od předchozích řešení, tento systém nedisponuje programovým aplikačním rozhraním. Z toho plyne, že zákazníci nemohou odeslat objednávku

přímo na rozhraní informačního systému, ale musí ji zaslat společnosti, která ji do systému zadá manuálně.



Obrázek 1.11: Ukázka skladového systému CÉZAR G3: Evidence přijatých objednávek

1.2.2.5 Evidence úloh

Vzhledem k tomu, že analyzovaný systém neumožňuje evidenci fyzických skladových umístění, nabízí pouze typické skladové operace. Mezi tyto operace patří *vyskladnění*, *naskladnění*, *přeskladnění mezi podsklady* a *inventura*.

I u tohoto skladového systému můžeme pozorovat shodné rozdíly jako u ekonomického systému POHODA. Najdeme zde shodný rozdíl ve způsobu řešení úlohy i ve způsobu vyplňování informací o jednotlivých položkách dokladu. Dokonce i tento systém umožňuje zaúčtování inventury.

Další rozdíl, který v tomto systému nalézt, je v úloze přeskladnění. Zatímco skladový systém Sysel umožňuje přesouvat zboží mezi sklady, tento systém umožňuje přesouvat zboží pouze mezi podsklady. Bude-li uživatel chtít provést přesun zboží mezi sklady, musí nejprve zboží vyskladnit ze zdrojového skladu a následně zboží naskladnit do cílového skladu.

1.2.2.6 Evidence reklamací

U každé reklamace dovoluje systém evidovat následující údaje: číslo reklamace, datum vytvoření, číslo výdejového dokladu, ke kterému se reklamace vztahuje, referenci na předchozí reklamaci (pokud existuje), stav reklamace,

informace o zákazníkovi a informace o reklamovaném zboží. Dále může uživatel k reklamaci připojit seznam zboží, jenž bylo spotřebováno v průběhu řešení reklamace.

Stav reklamace může nabývat jedné ze čtyř hodnot, kterými jsou *přijatá*, *čekání na dodavatele*, *čekání na zákazníka* a *vyřízená*. Systém samozřejmě nabízí také tisk všech potřebných dokumentů.

1.2.2.7 Evidence zásilek

Poslední oblastí, na kterou se zaměřím, je evidence zásilek čekajících na vyzvednutí dopravcem.

Systém řeší evidenci zásilek pomocí takzvaných zásilkových listů, jenž obsahují seznam vyskladnění (zásilek), které budou následně hromadně předány dopravci. U každé zásilky může uživatel nastavit jejího adresáta, číslo zásilky, hodnotu, hmotnost, objem, poznámku a počet balíků, ve kterých je zásilka zabalena.

U zásilkového listu systém eviduje identifikační číslo, které přidělil dopravce, druh dopravce, datum přepravy a stav. Stav může nabývat jedné z hodnot: *otevřená*, *připravená k odeslání*, *probíhá odeslání* a *odesláno*. Dokud uživatel nezmění stav zásilkového listu do stavu *probíhá odesílání*, může do zásilkového listu přidávat další zásilky. Uživatel také může vytvořit více zásilkových listů pro jednoho dopravce, které jsou na sobě vzájemně nezávislé.

1.2.2.8 Shrnutí

I u tohoto skladového systému jsem objevil několik funkcí, u kterých stojí za zvážení, zda by je nebylo vhodné začlenit do nově vznikajícího skladového systému. Mezi tyto funkce patří například:

- Validace údajů subjektů vůči rejstříkům veřejné správy
- Způsob členění skladů na podsklady
- Evidence reklamací
- Čárový kód u každého objektu (zboží, odběratel, dodavatel atp.)

1.2.3 Pokladní systém Dotykačka

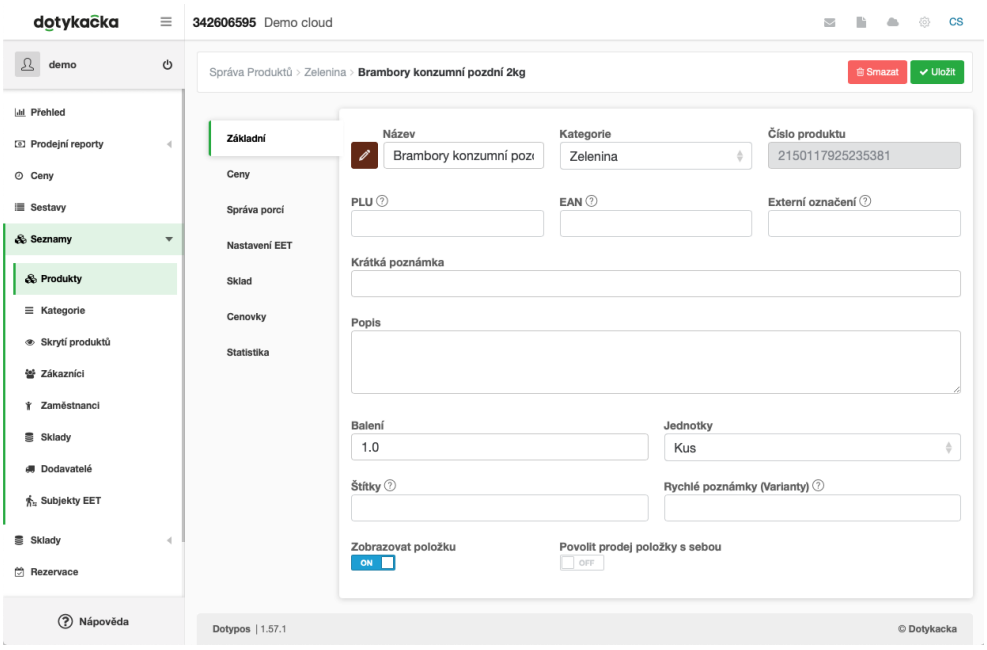
„Pokladní systémy Dotykačka zvládnou tisíce účtů, položek, neomezený počet uživatelů a transakcí. Naše pokladní systémy jsou připraveny na EET a mají množství praktických funkcí jako mapa stolů, happy hours, take away a například odtěžování ingrediencí. Vzdálenou správu a skladovou aplikaci uvítají podnikatelé s větším počtem skladových položek, pokladen a uživatelů. Pokladní systémy Dotykačka jsou vhodné jako pokladní systémy pro restaurace, pokladní systémy pro obchod, kavárny, bary a další gastro provozovny, stánkový prodej,

služby, řemeslníky.“ [4] Jedná se o české řešení od stejnojmenné společnosti Dotykačka.

Ačkoli toto řešení cílí primárně na jinou oblast, než je čistě skladová evidence, myslím si, že i přesto je vhodné se v rámci analýzy tímto systémem zaobírat, jelikož může nabídnout trochu jiný pohled na věc. Během analýzy se budu zabývat výhradně funkčností související se skladovou evidencí.

1.2.3.1 Evidence skladů

„Ve Vzdálené správě můžete vytvořit několik skladů a ty přiřadit určitým pokladnám. Každá pokladna může mít vlastní sklad, ze kterého odčerpává zboží nebo naopak jeden sklad může sloužit pro více pokladen. Můžete dokonce nastavit rozdílný sklad v závislosti na prodávaných produktech.“ [5] U každého skladu eviduje jeho název, barvu (pro snazší orientaci v rámci pokladního systému) a seznam pokladen, které mohou s daným skladem pracovat. Na rozdíl od předchozích řešení neumožňuje tento systém logické členění skladů na menší podsklady.



Obrázek 1.12: Ukázka pokladního systému Dotykačka: Evidence zboží

1.2.3.2 Evidence zboží

V rámci evidence zboží umožňuje tento systém evidovat více informací o konkrétních produktech. Uživatel může u každého produktu určit do jaké patří

kategorie, dále může definovat měrnou jednotku, štítek a další věci, které se však již netýkají skladové evidence. *„Kategorie umožňují přehledně třídit produkty a především v případě velkého množství produktů usnadňují jejich hledání a účtování přímo na pokladnách.“* [6] U každé kategorie systém eviduje její název, externí označení, marži a výchozí sazbu DPH.

Na obrázku 1.12 je zachycena ukázka z evidence zboží pokladního systému Dotykačka.

1.2.3.3 Evidence dodavatelů a odběratelů

Evidenci dodavatelů a odběratelů řeší pokladní systém Dotykačka podobným způsobem jako skladový systém Sysel. O subjektech eviduje základní informace jako je jméno, fakturační a dodací adresu, kontaktní informace, IČO a DIČ. U odběratelů navíc umožňuje evidovat čárový kód, který slouží pro rychlé nalezení odběratele během odbavení nákupu u poklady, a slevovou skupinu, podle které je vypočítána prodejní cena zboží. Podobně jako skladový systém CÉZAR i tento systém nabízí validaci údajů vůči rejstříkům veřejné správy.

1.2.3.4 Evidence úloh

Zpracování úloh se oproti předchozím systémům značně liší. Pokladní systém Dotykačka podporuje následující skladové operace: *naskladnění*, *přesun mezi sklady* a *inventuru*. Vyskladnění je realizováno formou prodeje zboží.

Pro naskladnění zboží nabízí systém uživateli formulář, kde vyplní sklad, ke kterému se naskladnění vztahuje, dodavatele, číslo dodacího listu, poznámku, a seznam zboží. U každého zboží může uživatel nastavit nákupní cenu, prodejní cenu a naskladňované množství. Pokud uživatel zadá záporné množství, systém to pochopí jako korekci skladu a příslušné množství zboží ze skladu odečte. Uživatel může provést naskladnění také skrze import datového souboru ve formátu CSV.

Přesun zboží mezi sklady lze realizovat skrze jiný formulář, který do jisté míry připomíná formulář naskladnění. Uživatel vybere zdrojový a cílový sklad, dále může vyplnit číslo skladového přesunu, poznámku a seznam zboží, které má být přesunuto. Uložením formuláře dochází k okamžitému přesunu zboží.

Inventuru může uživatel provést buď pro celý sklad, nebo může provést inventuru zboží, které patří do určité kategorie nebo má určitý štítek. Po zvolení požadováného omezení se uživateli zobrazí formulář, kam musí zadat skutečný stav zboží na skladě. Po ukončení inventury provede systém automaticky korekci skladu dle zjištěných rozdílů.

Poslední skladovou operací je vyskladnění. Tato operace, jak již bylo dříve řečeno, je automaticky realizována během prodeje zboží. To znamená, že jakmile obsluha pokladny prodá nějaké zboží zákazníkovi, systém toto zboží automaticky odečte z příslušného skladu. Nevýhodou tohoto řešení je skuteč-

nost, že uživatel nemůže provést vyskladnění nezávisle na prodeji nějakého zboží. Toto chování ovšem plyne ze specifického zaměření systému.

1.2.3.5 Evidence objednávek, reklamací a zásilek

Naneštěstí tyto funkce pokladní systém Dotykačka nenabízí. Ovšem je to pochopitelné z hlediska zaměření systému na jinou oblast zájmu.

1.2.3.6 Shrnutí

Přestože je pokladní systém Dotykačka zaměřen především na jinou problémovou doménu, i tak se našlo několik zajímavých funkcí, které bychom mohli do nově vznikajícího skladového systému začlenit. Pro rekapitulaci si tyto funkce dovolím připomenout:

- Zaúčtování inventury a skladové korekce
- Validace údajů subjektů vůči rejstříku veřejné správy
- Čárový kód pro rychlé vyhledání odběratele

1.3 Formulace požadavků

Nyní, po analýze současného a konkurenčních řešení, přichází ten správný čas na formulaci požadavků na nový skladový systém. Cílem této podkapitoly bude jasně a přesně specifikovat jakou funkčnost bude nový systém uživatelům nabízet. Tuto skutečnost se pokusím zachytit pomocí množiny funkčních a nefunkčních požadavků. Ovšem, než se do toho pustím, řekněme si nejprve něco o tom, co jsou to funkční a nefunkční požadavky.

Specifikace požadavků, někdy můžeme nalézt pod zkratkou SRS, slouží k jasnému vymezení hranic nově vyvíjeného systému. To znamená, že po přečtení dokumentu by mělo být každému jasné, jaká funkčnost bude v rámci vývoje pokryta. Samotné požadavky dělíme na *funkční* a *nefunkční*. První skupina požadavků popisuje, jak se bude daný systém chovat a jaké budou jeho vlastnosti a funkce. Naproti tomu nefunkční požadavky popisují obecné charakteristiky systému. Konkrétně může jít například o požadavky na bezpečnost, výkonnost, dostupnost či použitelnost systému. [7]

1.3.1 Funkční požadavky

1.3.1.1 FR1 Evidence uživatelů

Systém bude uživatelům s rolí vedoucího umožňovat správu uživatelů, jenž se budou moci přihlásit do skladového systému a dále s ním pracovat. Umožněna bude tvorba nových a úprava již vytvořených uživatelů. Dále bude umožněno uživatele aktivovat či deaktivovat. U každého uživatele bude systém evidovat

jeho jméno, příjmení, uživatelské jméno, email, telefon a uživatelské role. Vedoucí bude moci uživateli přiřadit některé ze tří nabízených uživatelských rolí, podle kterých bude řízen přístup k jednotlivým funkcím skladového systému. Konkrétně půjde o uživatelské role *vedoucí*, *skladník* a *zákazník*. U uživatelů s rolí vedoucí nebo skladník bude systém navíc udržovat jejich inventář, do nějž bude v průběhu řešení úloh ukládáno zboží, s kterým právě manipulují.

1.3.1.2 FR2 Evidence odběratelů

Systém bude umožňovat tvorbu, úpravu a zobrazení informací o jednotlivých odběratelích. V případě, že nebude mít odběratel žádné reference na jiné objekty, umožní systém uživateli odběratele odstranit. U každého odběratele bude systém evidovat jeho jméno, IČO, DIČ, telefon, email, webovou stránku, fakturační a dodací adresu. Vyplní-li uživatel během tvorby IČO, systém jednak zkontroluje jeho unikátnost napříč odběrateli a dále provede automatickou validaci zadaných údajů vůči rejstříkům veřejné správy. Systém umožní také vyhledávání odběratelů pomocí jejich interního identifikačního čísla a IČO, pokud bude vyplněno. Tvorba a úprava odběratelů bude přístupná pouze uživatelům s rolí vedoucí.

1.3.1.3 FR3 Evidence dodavatelů

Systém bude umožňovat tvorbu, úpravu a zobrazení informací o dodavatelích. V případě, že nebude mít dodavatel žádné reference na jiné objekty, umožní systém uživateli dodavatele odstranit. U každého dodavatele bude systém evidovat jeho jméno, IČO, DIČ, telefon, email, webovou stránku a fakturační adresu. Vyplní-li uživatel během tvorby IČO, systém jednak zkontroluje jeho unikátnost napříč dodavateli, ale také provede automatickou validaci zadaných údajů vůči rejstříkům veřejné správy. Systém umožní také vyhledávání dodavatelů pomocí jejich interního identifikačního čísla a IČO, pokud bude vyplněno. Tvorba a úprava dodavatelů bude omezena pouze na uživatele s rolí vedoucí.

1.3.1.4 FR4 Evidence majitelů podskladů

Uživatelům s rolí vedoucí bude umožněna správa majitelů podskladů. U každého majitele bude evidováno jeho jméno, IČO, DIČ, telefon, email, webová stránka, fakturační adresa a bankovní účet. Uživatelé budou moci vytvářet či editovat majitele, dále jim bude umožněno zobrazit detailní náhled. Odstranění majitele ze skladového systému bude umožněno pouze v případě, že na něj nebudou v rámci systému existovat žádné reference. Vyplní-li uživatel v průběhu tvorby majitele IČO, systém nejprve zkontroluje jeho unikátnost napříč majiteli, a následně provede automatickou validaci zadaných údajů vůči rejstříkům veřejné správy. Systém umožní také vyhledávání majitelů dle interního identifikačního čísla a IČO, pokud bylo vyplněno.

1.3.1.5 FR5 Evidence skladů

Nový skladový systém bude umožňovat evidenci více skladů, ke kterým umožní souběžný přístup. U každého skladu bude evidován jeho název, adresa, vedoucí a velikost (malý nebo velký). Bude-li sklad označen jako malý, poté systém nedovolí uživateli vytvořit ve skladu více než jedno skladové umístění a jeden podsklad. V opačném případě bude možné v každém skladu vytvořit neomezené množství skladových umístění a podskladů. Uživatel bude moci kdykoli změnit velikost skladu, avšak pouze za předpokladu, že budou splněny podmínky na počet skladových umístění a počet podskladů.

Tvorba a úprava skladů bude umožněna uživatelům s rolí vedoucí. Odstranění již existujících skladů bude povoleno všem uživatelům s rolí vedoucí, avšak pouze v případě, že ve skladovém systému nebudou existovat žádné reference na daný sklad. Zobrazení informací o skladech bude přístupné všem uživatelům s rolí vedoucí nebo skladník.

1.3.1.6 FR6 Evidence skladových umístění

U každého skladu bude umožněno vytvořit neomezené množství skladových umístění. Systém bude u těchto umístění evidovat jejich název a čárový kód, který bude unikátní napříč celým skladovým systémem. Tvorba a úprava skladových umístěné bude umožněna pouze uživatelům s rolí vedoucí. Odstranění skladových umístění bude umožněno pouze v případě, že uživatel bude mít uživatelskou roli vedoucí a že na dané skladové umístění nebudou existovat žádné reference. Zobrazení skladových umístění bude umožněno uživatelů s rolí vedoucí a skladník.

1.3.1.7 FR7 Evidence podskladů

V rámci každého skladu bude systém umožňovat evidovat neomezené množství podskladů, které budou představovat logické členění skladu. U každého podskladu bude evidován jeho název, majitel, příznak, který bude říkat, zda budou objednávky přicházející do tohoto podskladu automaticky uspokojovány, a seznam skladových umístění, jenž bude podsklad používat. Tvorba a úprava podskladů bude umožněna všem uživatelům s rolí vedoucího. Odstranění podskladů bude umožněno pouze uživatelům s uživatelskou rolí vedoucí, avšak pouze za předpokladu, že na daný podsklad nebudou v rámci skladového systému existovat žádné reference. Zobrazení informací o podskladech bude umožněno všem uživatelům s rolí vedoucí nebo skladník.

1.3.1.8 FR8 Evidence výrobců

Systém bude podporovat evidenci výrobců zboží. U každého výrobce bude evidováno jeho jméno a zkratka. Tvorba nových a úprava stávajících výrobců bude umožněna všem uživatelům s rolí vedoucí. Odstranění výrobce bude

umožněno uživatelům s uživatelskou rolí vedoucí, avšak pouze za předpokladu, že na daného výrobce nebudou v rámci skladového systému existovat žádné reference. Zobrazení informací o výrobcích bude umožněno všem přihlášeným uživatelům.

1.3.1.9 FR9 Evidence skladových karet zboží

Systém bude nabízet evidenci zboží. Každé produkt bude mít vlastní skladovou kartu, na kterou budou navázány další informace jako je například seznam výrobních čísel nebo seznam šarží. Dále zde budou uloženy následující informace: název zboží, popis, model, výrobce, prodejní cena, doporučená nákupní cena, která bude sloužit pro předvyplnění formuláře v naskladnění, sazba DPH, fotografie, množina čárových kódů a další atributy, které budou moci uživatelé svobodně vytvářet. Systém bude rozlišovat dva druhy fotografií, a to lokální a externí. Lokální fotografie budou uloženy fyzicky v datových souborech skladového systému, zatímco u externích fotografií bude systém evidovat pouze odkaz na určitou webovou stránku.

Tvorba nových a úprava existujících skladových karet bude umožněna všem uživatelům s rolí vedoucí nebo skladník. Odstranění skladových karet nebude umožněno. Místo toho budou moci uživatelé, s rolí vedoucí nebo skladník, zboží aktivovat či deaktivovat. Deaktivované zboží nebude zobrazeno v přehledových seznamech, ale bude přístupné při znalosti čárového kódu nebo interního identifikačního čísla. Zobrazení informací o produktech bude umožněno všem přihlášeným uživatelům. Zobrazený seznam bude stránkovaný a uživatelé jej budou moci řadit či filtrovat dle libosti.

1.3.1.10 FR10 Evidence produktových šarží

U každé produktové šarže bude systém evidovat její kód, datum výroby a datum expirace. Tvorba a úprava šarží bude umožněna uživatelům s rolí vedoucí a skladník. Odstranit šarži bude možné pouze v případě, že uživatel bude mít uživatelskou roli vedoucí nebo skladník a že na produktovou šarži nebudou odkazovat jiné objekty. Zobrazení informací o šaržích bude umožněno všem přihlášeným uživatelům.

1.3.1.11 FR11 Evidence výrobních čísel

Systém umožní u skladové karty zboží evidovat seznam výrobních čísel. Tato výrobní čísla budou vždy spojená pouze s jednou konkrétní skladovou kartou zboží a v rámci dané množiny budou vždy unikátní. Tvorba a úprava výrobních čísel bude umožněna uživatelům s rolí vedoucí nebo skladník. Odstranění výrobního čísla bude možné pouze v případě, že uživatel bude mít uživatelskou roli vedoucí nebo skladník a že na výrobní číslo nebudou odkazovat žádné jiné objekty. Zobrazení informací o výrobních číslech bude umožněno všem přihlášeným uživatelům.

1.3.1.12 FR12 Cenové úrovně

Systém umožní správu cenových úrovní. U každé cenové úrovně bude uloženo její jméno, seznam skladových karet zboží, kterých se týká, a koeficient, kterým bude upravena konečná prodejní cena. Pro každé zboží bude systém umožňovat vytvořit více cenových úrovní. Již vytvořené cenové úrovně bude možné přiřadit k více odběratelům, avšak pouze za předpokladu, že se cenové úrovně nebudou překrývat. Přístup k tvorbě, úpravě a odstranění cenových úrovní bude omezen pouze na uživatele s rolí vedoucí.

1.3.1.13 FR13 Vyhledávání zboží podle čárových kódů

Přihlášeným uživatelům bude umožněno rychlé vyhledávání zboží podle čárového kódu. U každého čárového kódu zboží bude mimo reference na konkrétní zboží uložen také počet kusů, které má systém po načtení kódu zpracovat.

1.3.1.14 FR14 Vyhledávání skladových umístění podle čárových kódů

Přihlášeným uživatelům bude umožněno rychlé vyhledávání skladových umístění dle čárových kódů.

1.3.1.15 FR15 Vyhledávání subjektů v rejstříku veřejné správy

Systém umožní vyhledávat informace o subjektech v registrech veřejné správy České republiky. Uživatel zadá na vstup IČO subjektu a systém vrátí informace o subjektu ve strojově čitelném formátu.

1.3.1.16 FR16 Interní čárové kódy

Systém bude pro každý objekt ve skladovém systému evidovat takzvaný interní čárový kód. Tyto čárové kódy budou složené z pevně definovaného prefixu a interního identifikačního čísla objektu. Například může jít o čárový kód *X07-000874*, který bude říkat, že se jedná o skladové umístění (dle prefixu X07) s identifikátorem 874. Systém bude tyto čárové kódy uživateli nabízet u objektů, které nemají žádné jiné uživatelem definované čárové kódy.

1.3.1.17 FR17 Tisk čárových kódů

Systém umožní tvorbu datových souborů pro tisk čárových kódů. Konkrétně půjde o datové formáty PDF, ZPL a EPL. Uživatel bude moci určit kolik čárových kódů se má v rámci jednoho požadavku vytvořit.

1.3.1.18 FR18 Evidence nákupních cen

Systém bude u zboží v podskladech evidovat nákupní ceny. Tyto ceny se budou odvíjet od nákupních cen zadaných během naskladnění příslušného zboží do

podskladu. Vyskladnění zboží bude realizováno metodou FIFO. To znamená, že ze skladu bude vždy odstraněno to nejstarší zboží. V podskladu tedy budou udržovány pouze nejnovější nákupní ceny.

1.3.1.19 FR19 Evidence skladových pohybů

Systém umožní uživatelům s rolí vedoucí zobrazit u každého podskladu přehled skladových pohybů. Zobrazený přehled bude stránkovaný a bude jej možné filtrovat dle data uskutečnění operace a dle interního identifikačního čísla skladové karty zboží. U každého pohybu bude systém uživateli zobrazovat následující informace: typ operace (příjem nebo výdej), podsklad a skladové umístění, kterého se pohyb týká, zboží a počet kusů, se kterými bylo manipulováno, úlohu, v rámci které byl pohyb proveden, datum a čas vytvoření, osobu, která pohyb provedla, a počet kusů zboží, který na umístění v daném podskladu zůstal po uskutečnění pohybu.

1.3.1.20 FR20 Zobrazení stavu skladu

Systém umožní uživatelům, s uživatelskou rolí vedoucí nebo skladník, zobrazit přehled zboží v podskladu, který bude možné dále filtrovat buď dle skladového umístění, nebo dle skladové karty zboží. Přehled bude možné zobrazit jednak k současnému datu, ale také k datu, který určí uživatel systému. V přehledu budou zobrazeny následující informace: skladové umístění, na kterém se produkt nachází, produkt, který je na skladovém umístění uložen, počet kusů konkrétního produktu na skladovém umístění a datum a čas poslední změny.

1.3.1.21 FR21 Historie změn

Systém bude u všech objektů, mimo úloh, udržovat historii změn, jež uživatelé během používání systému provedli. V systému tedy bude dohledatelné, kdo změnu provedl a co bylo v rámci této změny upraveno.

1.3.1.22 FR22 Časové značky

U všech evidovaných objektů bude systém udržovat časové značky, z kterých bude možné zjistit, kdy byl objekt vytvořen a kdo jej vytvořil. Dále bude možné zjistit kdo jej změnil a kdy změnu provedl. Jaké změny uživatel provedl budou dohledatelné v historii změn – viz FR24.

1.3.1.23 FR23 Zobrazení úloh

Systém umožní vybraným uživatelům zobrazit informace o právě probíhajících, nebo již uzavřených úlohách. Zatímco vedoucímu bude umožněno zobrazit jakoukoli úlohu, skladníkovi bude umožněno zobrazit pouze takové úlohy, které

jsou buď jemu přiřazené, nebo jsou volné (tzn. není nikomu přiřazena). Zobrazený přehled bude stránkován a bude jej možné filtrovat či řadit dle vybraných údajů.

1.3.1.24 FR24 Přiřazení úloh

Systém umožní uživatelům s rolí skladník vzít si libovolné množství volných úloh ke zpracování. Jakmile si uživatel vezme úlohu ke zpracování, úloha zmizí ze seznamu volných úloh a systém uživateli zpřístupní rozhraní nutné pro vyřešení dané úlohy. Struktura tohoto rozhraní se bude lišit v závislosti na právě řešeném typu úlohy.

1.3.1.25 FR25 Tvorba poznámek u úloh

Systém bude umožňovat řešitelům úloh a vedoucím vkládat k úlohám libovolné množství poznámek (komentářů). Uživatelé budou moci vytvářet nové poznámky a dále jim bude umožněno upravovat a mazat jimi vytvořené poznámky. Vedoucím bude navíc povolen přístup k editaci poznámek jiných uživatelů. U každé poznámky bude systém evidovat její obsah, datum a čas vytvoření, datum a čas úpravy a uživatele, jenž poznámku vytvořil.

1.3.1.26 FR26 Připojení příloh k úloze

Systém bude umožňovat vedoucím a řešitelům úloh připojit libovolné množství příloh ke každé úloze. Uživatelům bude povoleno vytvářet nové přílohy, stahovat již existující přílohy a odstraňovat jimi vytvořené přílohy. Vedoucím bude navíc umožněno smazat jakoukoli přílohu. O každé příloze bude systém navíc evidovat následující informace: jméno souboru, čas a datum vytvoření a uživatele, jenž přílohu vytvořil.

1.3.1.27 FR27 Časové výkazy u úloh

Systém umožní řešitelům úloh a vedoucím připojovat k úlohám časové výkazy. Uživatelům bude umožněno vytvářet nové časové výkazy. Dále jim bude umožněno upravovat a mazat jimi vytvořené záznamy. Vedoucím bude navíc povoleno upravovat a mazat záznamy jiných uživatelů. U každého výkazu bude systém evidovat uživatele, datum a čas vytvoření a počet minut.

1.3.1.28 FR28 Přehled časových výkazů

Systém umožní vedoucím zobrazit přehled časových výkazů, kde budou moci zjistit, kolik času bylo úlohám věnováno. Zobrazený přehled bude stránkovaný a bude jej možné filtrovat dle data.

1.3.1.29 FR29 Příjem dodávky zboží

Systém umožní skladníkovi vytvořit novou úlohu *příjem dodávky zboží*, jejímž cílem je informovat vedoucího o příjmu zboží od dodavatele. V rámci úlohy budou evidovány následující údaje: číslo faktury, číslo dodacího listu, dodavatel a fotografie dodacího listu. Úlohu bude vždy zpracovávat vedoucí, který příjem dodávky buď odmítne, nebo schválí. Schválení úlohy vytvoří novou úlohu naskladnění a ukončí úlohu na příjem dodávky zboží. Odmítnutí úlohy bude mít za následek pouze její ukončení.

1.3.1.30 FR30 Naskladnění

Systém bude obsahovat podporu pro naskladnění zboží do podskladu. Uživateli s rolí vedoucí bude umožněno vytvořit novou úlohu *naskladnění*, která může buď vycházet z výstupu úlohy *příjem dodávky zboží*, nebo může být zcela nezávislá. Bude-li úloha vycházet z příjmu dodávky zboží, budou uživatelům, v náhledu úlohy, zobrazeny informace týkající se předešlé úlohy. Během tvorby úlohy bude systém od uživatele vyžadovat následující údaje: podsklad, do kterého bude zboží naskladněno, prioritu úlohy, poznámky k řešení a určení preferencí na volbu skladových umístění. Jakmile uživatel úlohu vytvoří, systém ji bude zobrazovat v seznamu volných úloh, odkud si ji bude moci vzít kterýkoli uživatel s rolí skladník. Zadá-li uživatel úlohu omylem, bude ji moci zrušit, avšak pouze do chvíle, než si úlohu převezme některý uživatel s rolí skladník.

V okamžiku, kdy si uživatel s rolí skladník úlohu převezme k řešení, systém zobrazí rozhraní pro její vyřešení. V tomto rozhraní bude uživateli umožněno načítat zboží a umisťovat jej na příslušná skladová umístění, jenž vyhovují preferencím zadavatele. Jakmile uživatel načte veškeré dodané zboží, bude moci úlohu předat vedoucímu ke schválení. Poté, co bude úloha předána vedoucímu, se k ní již nebude moci vrátit.

Uživatel s rolí vedoucí bude moci vyřešenou úlohu buď schválit, nebo vrátit k přepracování. Rozhodne-li se vrátit úlohu k přepracování, bude muset nejprve určit, zda se úloha vrátí k původnímu řešiteli, nebo zda se z ní stane volná úloha, kterou si bude moci vzít jiný skladník. V opačném případě bude umožněno uživateli vyplnit nákupní ceny k dodanému zboží. Po vyplnění nákupních cen bude úloha označena jako uzavřená a změny se promítnou do skladového systému.

Systém bude mimo jiné umožňovat také tisk příjmového dokladu ve formátu PDF.

1.3.1.31 FR31 Přesun zboží po skladě

Systém bude podporovat tvorbu úloh pro přesun zboží mezi skladovými umístěními v podskladě. Podporovány budou tři varianty a to:

- přesun zboží z jednoho skladového umístění na mnoho různých skladových umístění,
- přesun zboží z více skladových umístění na jedno jiné skladové umístění,
- přesun zboží jednoho skladového umístění na jedno jiné skladové umístění.

Tato úloha zastupuje tři úlohy ze současného řešení. Konkrétně jde o zástupce za úlohy rozmístění, přeskladnění zboží a sestěhování.

Tvorba úlohy *přesun zboží po skladě* bude umožněna všem uživatelům s rolí vedoucí. Během tvorby budou od uživatele vyžadovány následující údaje: zdrojové a cílové umístění (alespoň jedno musí být vyplněno), podsklad, kterého se úloha bude týkat, priorita úlohy, popis úlohy a seznam zboží, které bude v rámci úlohy přesunuto. Podmínkou pro vytvoření úlohy bude skutečnost, že zadané zboží je dostupné v daném podskladu a na určeném zdrojovém umístění. Jakmile uživatel úlohu vytvoří, systém ji bude zobrazovat v seznamu volných úloh, odkud si ji bude moci vzít kterýkoli uživatel s rolí skladník. Zadáli uživatel úlohu omylem, bude ji moci zrušit, avšak pouze do chvíle, než si úlohu převezme některý uživatel s rolí skladník.

V okamžiku, kdy uživatel s rolí skladník převezme úlohu k řešení, systém zobrazí rozhraní pro její vyřešení. V tomto rozhraní bude uživateli umožněno přesouvat zboží ze zdrojového umístění do jeho inventáře a z jeho inventáře na cílové umístění. Cílové umístění bude moci uživatel kdykoli změnit, avšak nové umístění bude muset vždy splňovat podmínku na cílové umístění definované zadáním úlohy. Uživateli bude umožněno vrátit zboží na zdrojové umístění, avšak pouze na to, odkud jej vzal. Úlohu bude moci kdykoli předat vedoucímu ke schválení, ovšem pouze za předpokladu, že v tu chvíli nebude mít žádné zboží ve svém inventáři. Bude-li úloha zpracována jen zčásti, poté bude systém od uživatele vyžadovat komentář s odůvodněním, proč nebyla zpracována celá úloha. Poté, co bude úloha předána vedoucímu, se k ní již uživatel nebude moci vrátit.

Uživatel s rolí vedoucí bude moci vyřešenou úlohu buď schválit, nebo vrátit k přepracování. Rozhodne-li se úlohu vrátit k přepracování, bude muset nejprve určit, zda se úloha vrátí k původnímu řešiteli, nebo zda se z ní stane volná úloha, kterou si bude moci vzít jiný skladník. V opačném případě bude úloha označena jako uzavřená.

1.3.1.32 FR32 Přesun zboží mezi podsklady

Systém bude podporovat tvorbu úloh pro přesun zboží mezi podsklady stejného skladu. Tvorba úlohy *přesun zboží mezi podsklady* bude umožněna všem uživatelům s rolí vedoucí. Během tvorby úlohy bude systém od uživatele vyžadovat následující údaje: zdrojový a cílový podsklad, prioritu úlohy, popis úlohy a seznam zboží, které bude v rámci úlohy přesunuto. Dříve než systém úlohu

vytvoří, ověří, že zadané zboží je v daném zdrojovém podskladu fyzicky dostupné. Jakmile uživatel úlohu vytvoří, systém se ji pokusí automaticky zpracovat. Pokud se mu to podaří, úlohu ihned uzavře. Pokud se mu to nepodaří, například protože podsklady nebudou mít shodná skladová umístění, zobrazí úlohu v seznamu volných úloh, odkud si ji bude moci vzít kterýkoli uživatel s rolí skladník. Zadá-li uživatel úlohu omylem, bude ji moci zrušit, avšak pouze v případě, že úloha nebyla vyřešena automaticky a úlohu si ještě nepřevzal žádný uživatel s rolí skladník.

Jakmile uživatel s rolí skladník převezme úlohu k řešení, systém zobrazí rozhraní potřebné pro její vyřešení. V tomto rozhraní bude uživateli umožněno přesouvat zboží ze skladového umístění ve zdrojovém podskladu do jeho inventáře a z jeho inventáře na skladové umístění v cílovém podskladu. Cílové umístění bude moci uživatel kdykoli změnit, avšak nové zvolené skladové umístění se bude muset nacházet v cílovém podskladu. Dále bude uživateli umožněno zboží vrátit na zdrojové umístění, ovšem pouze na to, odkud jej vzal. Zboží, které již jednou bylo přesunuto do cílového podskladu, nebude možné, v rámci této úlohy, přesunout zpět na zdrojové umístění. Uživatel bude moci úlohu kdykoli předat vedoucímu ke schválení, ovšem pouze za předpokladu, že v tu chvíli nebude mít žádné zboží ve svém inventáři. Bude-li úloha zpracována jen zčásti, systém bude od uživatele vyžadovat komentář s odůvodněním, proč nebyla zpracována celá úloha. Poté, co bude úloha předána vedoucímu ke schválení, se k ní již uživatel nebude moci vrátit.

Uživatel s rolí vedoucí bude moci vyřešenou úlohu buď schválit, nebo vrátit k přepracování. Rozhodne-li se úlohu vrátit k přepracování, bude muset nejprve určit, zda se úloha vrátí k původnímu řešiteli, nebo zda se z ní stane volná úloha, kterou si bude moci vzít jiný skladník. V opačném případě bude úloha označena jako uzavřená.

1.3.1.33 FR33 Inventura

Systém bude podporovat tvorbu úlohy *inventura*. Ve výchozím stavu se bude inventura vztahovat k celému skladu, ovšem bude ji možné omezit podle podskladu, výrobce, skladových umístění, majitele nebo skladové karty zboží. Tvorba tohoto typu úlohy bude umožněna všem uživatelům s rolí vedoucí.

Během tvorby úlohy bude systém od uživatele vyžadovat následující údaje: sklad, kterého se inventura bude týkat, prioritu úlohy, popis úlohy a omezení na rozsah. Jakmile uživatel úlohu vytvoří, systém ji bude ihned zobrazovat v seznamu volných úloh, odkud si ji bude moci vzít kterýkoli uživatel s rolí skladník. Zadá-li uživatel úlohu omylem, bude ji moci zrušit, avšak pouze do chvíle, než si úlohu převezme některý uživatel s rolí skladník.

V okamžiku, kdy uživatel s rolí skladník převezme úlohu k řešení, systém zobrazí rozhraní potřebné pro její vyřešení. V tomto rozhraní bude uživateli umožněno zaznamenávat zboží, které nalezne na fyzických skladových umístěních. Jakmile uživatel zaznamená veškeré zboží, bude moci úlohu předat ve-

doucímu ke schválení. Poté, co bude úloha předána vedoucímu, se k ní již nebude moci vrátit.

Uživatel s rolí vedoucí bude moci vyřešenou úlohu buď schválit, nebo vrátit k přepracování. Rozhodne-li se úlohu vrátit k přepracování, bude muset nejprve určit, zda se úloha vrátí k původnímu řešiteli, nebo zda se z ní stane volná úloha, kterou si bude moci vzít jiný skladník. V opačném případě bude úloha označena jako uzavřená a uživateli bude umožněno provést zaúčtování inventury – viz FR34.

Systém bude mimo jiné umožňovat také tisk dokumentu, ve kterém bude zaznamenán výstup provedené inventury.

1.3.1.34 FR34 Zaúčtování inventury

Systém bude u každé uzavřené inventury podporovat její promítnutí do stávajícího stavu podskladu. Použití této funkce bude přístupné pouze uživatelům s rolí vedoucí a bude povoleno pouze u poslední provedené inventury. To znamená, že bude-li pro daný podsklad existovat nějaká novější inventura, systém nedovolí tuto operaci provést.

1.3.1.35 FR35 Vyskladnění

Systém bude umožňovat tvorbu úlohy na vyskladnění zboží z podskladu. Uživatelům s rolí vedoucí bude umožněno vytvořit novou úlohu *vyskladnění*, která skončí buď tím, že uživatel osobně předá zboží odběrateli, nebo tím, že se z vyskladnění vytvoří nová zásilka.

Během tvorby úlohy bude systém od uživatele vyžadovat následující údaje: podsklad, ze kterého bude zboží vyskladněno, prioritu úlohy, odběratele, poznámky k řešení, seznam zboží a způsob předání. Bude-li zboží předáno formou zásilky, bude systém od uživatele vyžadovat ještě informaci o tom, kam má řešitel zboží umístit. Jakmile uživatel úlohu vytvoří, systém ji bude ihned zobrazovat v seznamu volných úloh, odkud si ji bude moci vzít kterýkoli uživatel s rolí skladník. Zadá-li uživatel úlohu omylem, bude ji moci zrušit, avšak pouze do chvíle, než si úlohu převezme některý uživatel s rolí skladník.

V okamžiku, kdy uživatel s rolí skladník úlohu převezme k řešení, zobrazí systém rozhraní nutné pro její vyřešení. V tomto rozhraní bude uživateli umožněno přesouvat zboží z podskladu jeho inventáře. Bude-li zboží předáno formou zásilky, bude uživateli umožněno přesunout zboží z inventáře na cílové skladové umístění. Jakmile uživatel zpracuje veškeré zboží, bude moci úlohu úlohu označit jako vyřešenou. V případě, že se jedná o osobní předání, bude úloha ihned označena jako uzavřená. V opačném případě bude předána vedoucímu ke schválení. Poté, co bude úloha předána vedoucímu, se k ní již nebude moci vrátit.

Uživatel s rolí vedoucí bude moci vyřešenou úlohu buď schválit, nebo vrátit k přepracování. Rozhodne-li se uživatel vrátit úlohu k přepracování, bude

muset nejprve určit, zda se úloha vrátí k původnímu řešiteli, nebo zda se z ní stane volná úloha, kterou si bude moci vzít jiný skladník. V opačném případě bude uživateli umožněno buď zásilku připojit k již existujícímu zásilkovému listu, nebo pro zásilku vytvořit nový zásilkový list. Bude-li uživatel chtít vytvořit nový zásilkový list, poté bude systém od uživatele vyžadovat také informaci o tom, jaký dopravce bude zásilku zpracovávat. Dále bude uživateli umožněno nastavit prioritu a popis úlohy *příprava zásilky k odeslání* (viz FR39), jenž vznikne po uzavření úlohy.

Systém bude mimo jiné umožňovat také tisk výdejového dokladu ve formátu PDF.

1.3.1.36 FR36 Evidence dopravců

Systém bude evidovat seznam dopravců, jenž bude možné využít při volbě způsobu přepravy zásilky. Data v této evidenci nebude možné upravovat skrze aplikační rozhraní. To znamená, že jakákoli změna, týkající se této evidence, bude provedena výhradně vývojáři systému. Výjimku bude tvořit pouze konfigurace přístupových údajů k aplikačnímu rozhraní dopravce – viz FR37.

1.3.1.37 FR37 Integrace s reálnými dopravci

Systém bude podporovat integraci s přepravní společností DPD. Uživateli s rolí vedoucí bude umožněno zadat do systému přístupové údaje, jenž jsou nezbytné pro navázání komunikace s aplikačním rozhraním DPD GeoAPI[1].

1.3.1.38 FR38 Evidence zásilek

Systém bude podporovat evidenci zásilek – tj. zboží čekajícího na převzetí přepravní společností. Evidence bude řešena pomocí zásilkových listů, které budou obsahovat seznam zásilek, jež vznikly z úloh vyskladnění. U každého zásilkového listu bude systém evidovat dopravce, stav, datum a čas vytvoření a datum a čas odeslání. Stav zásilkového listu bude moci nabývat jedné z následujících hodnot: *vytvořeno*, *připraveno k vyzvednutí*, *čeká se na vyzvednutí* a *odesláno*. Přechod mezi stavy *vytvořeno* a *připraveno k vyzvednutí* bude řešit systém automaticky a to tak, že jakmile budou všechny zásilky uvnitř zásilkového listu připraveny k odeslání, systém změní stav na *připraveno k odeslání*. Přechod do dalších stavů budou uživatelé řešit manuálně. Přístup do evidence zásilek bude povolen pouze uživatelům s rolí vedoucí.

U každé zásilky bude systém evidovat její stav, hmotnost, hodnotu, která bude vypočtena z prodejních cen zboží, identifikační číslo přidělené dopravcem a počet balíků, ve kterých je zásilka zabalena. Zásilka může být buď

[1] Jedná se o webové rozhraní přepravní společnosti DPD, jehož cílem je umožnit programovou tvorbu zásilek. Více informací na: `https://www.dpd.com/cz/business_customers/vas_pruvodce_prepravou/aplikace_a_nastroje/dpd_geoapi`

připravená, nebo *nepřipravená*. Přechod mezi těmito stavy řeší úloha *příprava zásilky k odeslání* – viz FR39.

1.3.1.39 FR39 Příprava zásilky k odeslání

Systém bude nabízet funkčnost, která umožní vyřešení úlohy *příprava zásilky k odeslání*, jenž vznikne při uzavření úlohy vyskladnění. Uživatelé s rolí skladník budou moci tento typ úlohy nalézt v seznamu volných úloh.

Jakmile uživatel s rolí skladník převezme tuto úlohu k řešení, systém zobrazí rozhraní potřebné pro její vyřešení. V tomto rozhraní bude moci uživatel zadat počet balíků, do kterých zásilku zabalil, celkovou váhu zásilky a fotografii zabaleného zboží. V okamžiku, kdy uživatel zadá všechny potřebné údaje, bude moci úlohu předat vedoucímu ke schválení.

Uživatel s rolí vedoucí bude moci vyřešenou úlohu buď schválit, nebo vrátit k přepracování. Rozhodne-li se uživatel vrátit úlohu k přepracování, bude muset nejprve určit, zda se úloha vrátí k původnímu řešiteli, nebo zda se z ní stane volná úloha, kterou si bude moci vzít jiný skladník. V opačném případě bude úloha označena jako vyřešená.

1.3.1.40 FR40 Evidence objednávek

Systém umožní uživatelům s rolí vedoucí nebo zákazník tvorbu a následnou správu objednávek. Uživatelům bude umožněno objednávky vytvářet, zjišťovat jejich stav a případně je rušit (za předpokladu, že objednávka ještě nebyla zpracována). Vedoucím bude navíc umožněno objednávky uspokojovat. Každá objednávka bude moci být uspokojena buď celá, nebo jen zčásti, podle toho, zda se objednané zboží nachází fyzicky na skladě. V okamžiku naskladnění potřebného zboží bude možné částečně vyřešené objednávky uspokojit. Systém bude u každé objednávky evidovat její identifikační číslo a stav, odběratele, datum a čas vytvoření, plánovaný termín dodání, požadovaný termín dodání, seznam objednaného zboží včetně prodejních cen, dopravce a požadovaný způsob uspokojení objednávky. Stav objednávky bude nabývat jedné z následujících hodnot: *čeká na vyřízení*, *zpracovává se*, *vyřízená* a *zrušená*. Samotné zpracování objednávky bude řešeno pomocí úloh *vyskladnění* – viz FR35. Systém bude nabízet také podporu pro automatické uspokojování objednávek, avšak tato funkce bude konfigurovatelná u každého podskladu.

1.3.1.41 FR41 Správa objednávkových vstupů pro odběratele

Systém bude uživatelům s rolí vedoucí umožňovat správu objednávkových vstupů pro jednotlivé odběratele. Každý vstup bude vázán ke konkrétnímu odběrateli a bude obsahovat seznam podskladů, ve kterých odběratel může vytvářet nové objednávky. Odběratel bude disponovat nejvýše jedním objednávkovým vstupem. Správcům této evidence bude umožněno tvořit nové vstupy, upravovat a mazat již existující vstupy.

1.3.1.42 FR42 Evidence reklamací

Systém bude, mimo jinou funkčnost, nabízet také evidenci reklamací. U každé reklamace bude systém evidovat její identifikační číslo, stav, datum vytvoření, odpovědnou osobu, referenci na výdejový doklad, datum skutečného vyřešení, datum plánovaného vyřešení, informaci o reklamovaném zboží, popis reklamace, kontakt na zákazníka, požadovaný způsob vyřešení a referenci na předchozí reklamaci, pokud existuje. Každá reklamace se bude nacházet v jednom z pěti stavů: *reklamace přijata*, *zboží převzato od zákazníka*, *odesláno dodavateli*, *přijato od dodavatele* a *vyřešeno*. V průběhu celé reklamace bude umožněno uživatelům vkládat k reklamaci upřesňující komentáře. Systém bude, mimo samotné evidence, nabízet také tisk dokumentů souvisejících s řešením reklamace, například reklamační protokol. Přístup do správy reklamací bude umožněn pouze uživatelům s rolí vedoucí.

1.3.1.43 FR43 Mobilní aplikace

Nový skladový systém bude, podobně jako skladový systém Sysel, disponovat mobilní aplikací, která umožní jejím uživatelům snazší práci se skladovým systémem. Mobilní aplikace bude určena výhradně pro mobilní zařízení od společnosti Zebra Technologies s operačním systémem Android. Kompatibilita s jinými zařízeními nebude zajištěna.

1.3.2 Nefunkční požadavky

1.3.2.1 NFR1 Rozšířitelnost pomocí dalších API

Systém bude snadno rozšířitelný pomocí nezávislých aplikačních programových rozhraní. Těmto rozhraním bude poskytnut přístup k evidenci uživatelů a to v rozsahu nutném pro ověření pravomocí uživatelů, jenž chtějí s rozhraním pracovat. Pro přístup k ostatním funkcím skladového systému budou tato rozhraní využívat standardní funkce popsané v dokumentaci příslušného aplikačního rozhraní.

1.3.2.2 NFR2 Webová aplikace

Nový skladový systém bude realizován formou webové aplikace. To znamená, že uživateli bude k používání systému postačovat jakékoli zařízení disponující internetovým připojením a webovým prohlížečem.

1.3.2.3 NFR3 Lokalizace

Skladový systém bude dostupný v českém a anglickém jazyce.

1.3.2.4 NFR4 Autentizace a autorizace

Se skladovým systémem budou moci pracovat pouze přihlášení uživatelé, kteří budou mít omezen přístup pouze k vybraným částem systému. Tato omezení budou kladena v závislosti na tom, jakými uživatelskými rolemi uživatel disponuje.

1.3.2.5 NFR5 Volba technologií

Během návrhu budou zohledněny technologie, které již firma Jagu s. r. o. používá při vývoji jiných infromačních systémů. Důvodem tohoto omezení je minimalizace různorodosti používaných technologií.

1.3.2.6 NFR6 Architektura

Skladový systém bude rozdělen na klientskou část (frontend) a serverovou část (backend). Tento požadavek vyplývá ze zadání diplomové práce.

1.3.3 Omezení na vývoj systému

Vzhledem k počtu a časové náročnosti některých požadavků, jenž jsou kladeny na nový skladový systém, jsem se rozhodl, že některé požadavky z nadcházejícího návrhu a vývoje vyloučím a některé požadavky vyřeším jen zčásti. Do budoucna však s realizací těchto požadavků počítám, a proto budu během návrhu dbát na to, aby jejich následný vývoj nevyžadoval hlubší zásahy do již existující implementace. Mezi vyloučenými požadavky se nachází:

- FR12 Cenové úrovně
- FR35 Zaúčtování inventury
- FR37 Integrace s reálnými dopravci
- FR40 Evidence objednávek a poptávek
- FR41 Správa objednávkových přístupů pro odběratele
- FR42 Evidence reklamací

Mezi částečně řešené požadavky patří:

- FR16 Interní čárové kódy

  V prototypu budou interní čárové kódy dostupné pouze pro skladové karty zboží a skladová umístění. Podpora u ostatních objektů bude realizována až v dalších fázích vývoje systému.

- FR17 Tisk čárových kódů

  Prototyp bude podporovat tvorbu datových souborů pouze ve formátu PDF a ZPL.

- FR21 Historie změn

  Prototyp bude udržovat historii změn pouze u evidence výrobců. Podpora pro ostatní evidence bude doplněna v dalších fázích vývoje.

Kapitola 2

Návrh

V této kapitole se budu zabývat návrhem nového skladového systému podle množiny funkčních a nefunkčních požadavků, jejichž analýzou jsem se zabýval na konci minulé kapitoly. Jak si můžeme z předchozí kapitoly pamatovat, na nový systém bylo kladeno několik požadavků, které souvisely s architekturou systému. První věcí, kterou se tedy budu během návrhu zabývat, bude právě architektura systému.

V dalších částech kapitoly se budu zabývat konkrétními částmi systému, které vyplynou z návrhu architektury. Pořadí nebude náhodné, nýbrž bude odpovídat tomu, jak budou tyto části postupně vznikat v průběhu realizace systému.

2.1 Architektura

Vraťme se zpět k požadavkům a připomeňme si, jaké požadavky jsou kladeny na architekturu nového skladového systému. Identifikovat můžeme následující požadavky:

- **FR43 Mobilní aplikace**

  Tento požadavek říká, že systém bude disponovat mobilní aplikací. Tím vzniká první komponenta.

- **NFR1 Rozšířitelnost pomocí dalších API**

  Tento požadavek říká, že systém bude rozdělený do několika komponent, mezi nimiž bude jedna komponenta, která bude mít na starosti evidenci uživatelů.

- **NFR2 Webová aplikace**

  Tento požadavek říká, že systém bude realizován formou webové aplikace. Z toho vyplývá, že systém bude používat architekturu klient-server, kde server bude představovat skladový systém a klient bude jeho

uživatel. Předpokládejme, že půjde o standardní webové řešení, tedy jeden server, který obsluhuje mnoho klientů.

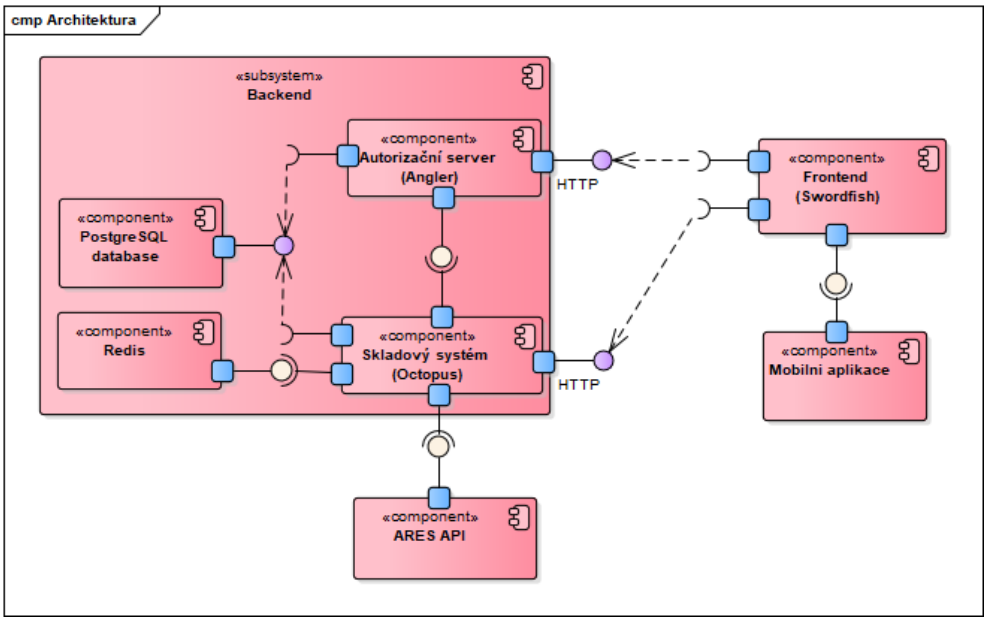
- **NFR4 Autentizace a autorizace**

  Tento požadavek se týká bezpečnosti a říká, že komponenta, která bude mít na starosti evidenci uživatelů, bude řešit také autentizaci a autorizaci.

- **NFR6 Architektura**

  Poslední požadavek říká, že systém bude rozdělen na dvě části, a to backend a frontend. Vzhledem k tomu, že frontend řeší paralelně diplomová práce Bc. Oldřicha Malce, budu jej dále v práci považovat za černou skříňku a budu jej modelovat jako jednu komponentu.

Když tato zjištění shrneme, zjistíme, že se systém bude skládat minimálně ze tří až čtyř komponent, podle toho, zda evidenci uživatelů vyčleníme do samostatné komponenty. Můj návrh s vyčleněním evidence uživatelů do samostatné komponenty počítá, a proto se systém bude skládat minimálně ze čtyř komponent. Návrh dále počítá s tím, že skladovou funkčnost bude, v tuto chvíli, řešit pouze jedna komponenta. Z toho vyplývá, že se systém bude skládat právě ze čtyř komponent. Konkrétně půjde o *mobilní aplikaci*, *frontend*, *autorizační server* a *skladový systém*. Pro lepší představu čtenáře je architektura celého systému zachycena na obrázku 2.1.



Obrázek 2.1: Architektura systému

2.2 Mobilní aplikace

První komponentou, kterou se budu v rámci práce zabývat, je mobilní aplikace. Pro připomenutí, cílem mobilní aplikace je umožnit jejím uživatelům snazší práci se skladovým systémem. Jak již může být zvídavému čtenáři zřejmé, pod tímto tvrzením si můžeme představit cokoli. Pojďme se tedy znovu zabývat sběrem požadavků a pokusme se co nejpřesněji zachytit, co bude cílem této aplikace.

2.2.1 Formulace požadavků

V této podkapitole budou uvedeny nejen požadavky, které vyplývají z chování původní aplikace, ale budou zde uvedeny také nové požadavky, které jsou na aplikaci kladeny nezávisle na požadavcích z kapitoly 1.3.

2.2.1.1 Funkční požadavky

- **MOB-FR1 Podpora pro interní čtečku čárových kódů**

  Aplikace bude podporovat práci s interní čtečkou čárových kódů, kterou disponují vybraná mobilní zařízení Zebra. Po načtení čárového kódu aplikace zajistí jeho zpracování.

- **MOB-FR2 Tisk dokumentů**

  Aplikace bude podporovat rychlý tisk dokumentů podobně jako současné řešení.

- **MOB-FR3 Podpora pro geolokační služby**

  Aplikace bude nabízet rozhraní pro zjištění aktuální polohy uživatele.

- **MOB-FR4 Viditelnost navigačního menu**

  Aplikace během práce schová navigační menu, které se nachází ve spodní části obrazovky a které zbytečně zabírá místo na obrazovce.

2.2.1.2 Nefunkční požadavky

- **MOB-NFR1 Omezení na mobilní zařízení Zebra**

  Z požadavku FR43 víme, že běh nové mobilní aplikace bude omezen pouze na mobilní zařízení společnosti Zebra Technologies s operačním systémem Android.

- **MOB-NFR2 Znovupoužitelnost**

  Aplikace nebude mít vlastní uživatelské rozhraní, nýbrž bude pouze rozšiřovat funkčnost komponenty frontend. Komponentu frontend bude možné kdykoli vyměnit, aniž by se odstranila nějaká funkčnost mobilní aplikace.

2.2.2 Návrh

Jak si můžeme z požadavků povšimnout, je zde kladen určitý důraz na znovupoužitelnost. Dále víme, že skladový systém, jenž bude aplikace používat, bude realizován formou webové aplikace. Logicky z toho tedy vyplývá, že by aplikace měla používat webový prohlížeč, ve kterém zobrazí uživatelské rozhraní systému.

Předpokládejme tedy, že základním stavebním kamenem aplikace bude webový prohlížeč (komponenta *WebView* [8]), ve kterém se při otevření aplikace otevře frontend skladového systému. Při hlubší analýze této komponenty můžeme zjistit, že již obsahuje podporu pro geolokační služby, čímž máme vyřešen požadavek MOB-FR3.

Nyní se zaměřme na požadavek týkající se viditelnosti UI prvků operačního systému. Dle dokumentace Androidu [9] můžeme v aplikaci vynutit schování jak stavové lišty v horní části obrazovky, tak i navigačního menu v dolní části obrazovky. Tím máme zajištěnou proveditelnost požadavku MOB-FR4.

Dále se zaměřme na požadavek MOB-FR2, který se zabývá tiskem dokumentů. Při hlubší analýze již používané aplikace můžeme zjistit, že tuto funkčnost neposkytuje přímo tato aplikace, nýbrž jiná aplikace, na kterou je tisk pouze delegován. Zajistíme-li tedy v nové aplikaci delegaci tisku na tuto pomocnou aplikaci, budeme mít problém vyřešen. Delegaci můžeme vyřešit pomocí standardního rozhraní operačního systému, jenž můžeme v literatuře nalézt pod pojmem *Intent* [10].

Poslední požadavek, který nám zbývá vyřešit, se týká podpory čtečky čárových kódů. Podporu pro tento požadavek můžeme nalézt v programovém rozhraní EMDK [11], jímž disponují mobilní zařízení od společnosti Zebra Technologies. Toto rozhraní umožňuje vývojářům přistupovat k různým funkcím mobilního zařízení jako je například Wi-Fi, správa napájení, fotoaparát, nastavení hodin atp. Jednou z těchto funkcí je i čtečka čárových kódů, ke které lze přistupovat skrze třídu *BarcodeManager* [12]. Skrze tuto třídu může vývojář získat právě načtený čárový kód, který může následně v aplikaci zpracovat. V kontextu naší mobilní aplikace jej zpracujeme tak, že jej předáme aplikaci, jenž se nachází ve webovém prohlížeči. Předání proběhne skrze zavolání javascriptové funkce *onBarcodeRead* v okně prohlížeče, kde prvním argumentem bude získaný čárový kód. Aplikace, která bude chtít čárový kód získat, bude muset tuto funkci implementovat.

2.2.3 Shrnutí

Mobilní aplikace bude realizována formou webového prohlížeče, jenž bude obohacen o vybrané funkce pocházející z programových rozhraní operačního systému. Výhodou tohoto řešení bude úplná nezávislost na aplikaci zobrazené ve webovém prohlížeči, jelikož veškerá komunikace bude probíhat přes javascriptové rozhraní.

2.3 Autorizační server

Další komponentou, kterou se v rámci této práce budu zabývat, je autorizační server. Cílem autorizačního serveru bude uchovávat evidenci uživatelů a řešit jejich autentizaci a autorizaci. Pro jeho realizaci jsem se rozhodl použít OAuth 2.0 framework s některými rozšířeními. Pojďme si na hned úvod říct něco o tom, co je to vlastně OAuth 2.0 framework.

2.3.1 OAuth 2.0

OAuth 2.0 je autorizační framework (respektive protokol) umožňující aplikacím třetích stran získat omezený přístup k nějakému webovému rozhraní. Hlavní výhoda tohoto frameworku je v tom, že uživatel může poskytnout aplikaci třetí strany přístup k jeho datům v nějaké službě, aniž by té aplikaci musel vyzradit své přihlašovací údaje. [13]

2.3.1.1 Popis rolí

Framework OAuth 2.0 definuje čtyři druhy rolí [13]:

- *resource owner* (uživatel)

  Entita schopná přidělit nebo odebrat přístup ke chráněnému zdroji. Nejčastěji se jedná o koncového uživatele systému.

- *resource server* (služba)

  Služba poskytující chráněné zdroje. Přijímá a obsluhuje požadavky obsahující přístupový token (access token).

- *client* (klient)

  Aplikace, která přistupuje ke chráněným zdrojům služby (resource server) s oprávněními uživatele (resource owner).

- *authorization server* (autorizační server)

  Server, který klientům vydává přístupové tokeny (access token) po úspěšné autentizaci uživatele (resource owner) a autorizaci požadavku.

2.3.1.2 Základní princip

V minulé podkapitole jsem se zabýval rolemi, jenž OAuth 2.0 framework definuje. Nyní se podívejme, jak tyto role spolu interagují. Průběh celé interakce je zachycen na obrázku 2.2 a obsahuje následující kroky [13]:

1. Klientská aplikace vyžádá přihlášení uživatele.

2. Uživatel se přihlásí a tím potvrdí klientovu žádost. Způsob přihlášení definují takzvané autorizační granty, jimiž se budeme zabývat v podkapitole 2.3.1.4.

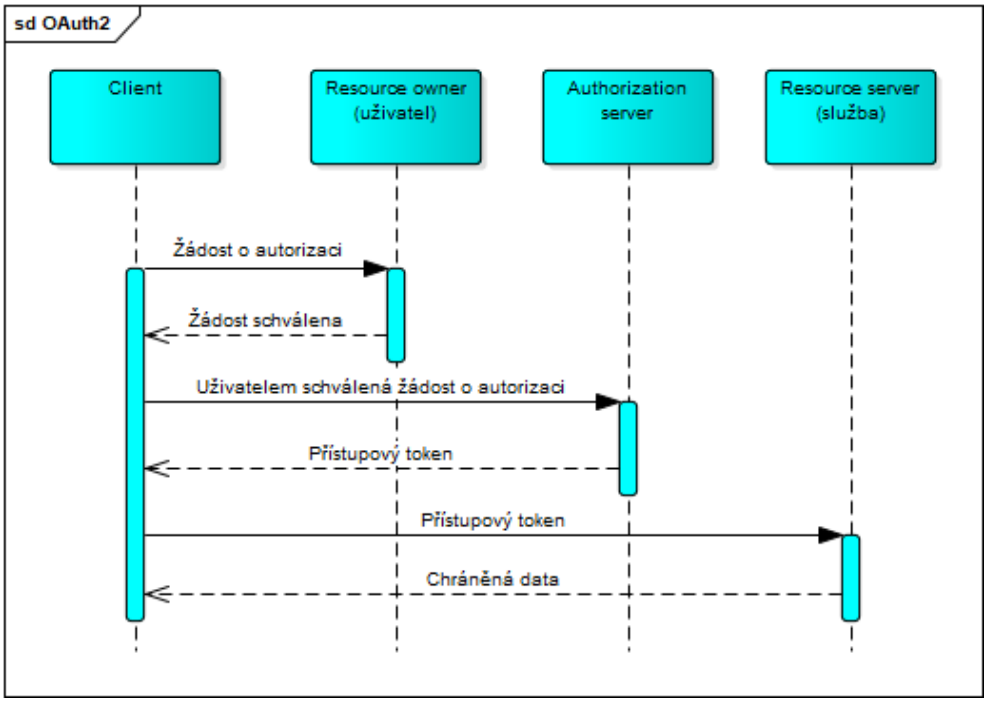
3. Klient zašle autorizačnímu serveru žádost o nový přístupový token. K žádosti připojí potvrzení získané od uživatele.

4. Autorizační server ověří příchozí žádost, a to včetně potvrzení získaného od uživatele. Pokud je vše platné, vygeneruje nový přístupový token a zašle jej klientovi.

5. Klient zašle dotaz na chráněný zdroj služby a připojí přístupový token.

6. Služba ověří platnost přístupového tokenu. Pokud je platný, vrátí uživateli chráněná data.

Platí tu však jedno omezení a to, že klient musí být na autorizačním serveru zaregistrován dříve, než se pokusí o navázání komunikace. V opačném případě bude navázání komunikace neúspěšné. Během registrace klient získá přístupový údaj *client_id* (někdy i *client_secret*), kterým bude při komunikaci s autorizačním serverem prokazovat svojí identitu – viz podkapitola 2.3.1.4.



Obrázek 2.2: Obecný průběh autorizace s využitím OAuth 2.0

2.3.1.3 Druhy tokenů

Specifikace OAuth 2.0 protokolu rozlišuje dva druhy tokenů. Konkrétně jde o *přístupový token (access token)* a *obnovovací token (refresh token)*.

Přístupový token představuje tajný klíč, který jednoznačně identifikuje relaci konkrétního uživatele. Obvykle se jedná o náhodně vygenerovaný textový řetězec, avšak v některých případech může jít o JWT[2] řetězec, ve kterém mohou být zakódované různé informace (například seznam privilegií či datum expirace). Používá ho především klient, který jej využívá k získání přístupu k chráněnému obsahu. Platnost tohoto tokenu je omezená, typicky na jednu hodinu.

Obnovovací token je tajný klíč, jenž se používá k získání nového přístupového tokenu v okamžiku, kdy aktuálnímu přístupovému tokenu vyprší jeho platnost. Na rozdíl od přístupového tokenu, se tento token používá pouze při komunikaci s autorizačním serverem. Jakékoli vyzrazení obnovovacího tokenu třetí straně má významný dopad na bezpečnost systému, jelikož pomocí toho tokenu může útočník generovat nové přístupové tokeny, které může zneužít ve svůj prospěch.

2.3.1.4 Autorizační granty

Autorizační grant stanovuje mechanismus, jakým způsobem bude probíhat potvrzení autorizační žádosti klienta a následné získání přístupového tokenu. Specifikace OAuth 2.0 frameworku definuje celkem čtyři typy autorizačních grantů [13]:

- Autorizační kód (*authorization code grant*)

  Autorizační kód je druh potvrzení autorizační žádosti, jenž klient získá v okamžiku úspěšného přihlášení uživatele na autorizačním serveru. Níže uvádím, pro lepší představu, postup získání a použití tohoto kódu:

  1. Klient přesměruje uživatele na autorizační server. Součástí URL adresy, na níž je uživatel přesměrován, jsou dva parametry *client_id* a *redirect_uri*. V parametru *client_id* je zaznamenán unikátní identifikátor klienta, pod kterým je veden v autorizačním serveru, a v parametru *redirect_uri* je uvedena adresa, na kterou bude uživatel přesměrován po úspěšném přihlášení.
  2. Autorizační server ověří příchozí žádost a zobrazí uživateli formulář, ve kterém se může přihlásit.
  3. Po úspěšném přihlášení přesměruje autorizační server uživatele zpět na klienta (dle parametru *redirect_uri*). V URL adrese, na níž je uživatel přesměrován, může klient nalézt parametr *code*, v němž je uložen autorizační kód.

[2] JSON Web Token – více informací na `https://jwt.io`

4. Klient odešle na autorizační server POST požadavek, ve kterém budou uvedeny následující parametry:
   - *grant_type* – vždy řetězec *authorization_code*
   - *client_id* – identifikátor klienta
   - *client_secret* – tajný klíč, který získal klient při registraci
   - *code* – autorizační kód
   - *redirect_uri* – adresa shodná s adresou odeslanou v bodu 1
5. Autorizační server ověří příchozí požadavek a pokud jsou všechny údaje platné, vrátí JSON strukturu, ve které bude uložen přístupový a obnovovací token.

Jak si můžeme povšimnout, výhodou tohoto mechanismu je skutečnost, že klientská aplikace vůbec nepotřebuje znát přihlašovací údaje uživatele, jelikož celý proces přihlášení proběhne na autorizačním serveru. Naopak slabinou tohoto mechanismu je přístupový údaj *client_secret*, jehož vyzrazení by mělo fatální dopad na bezpečnost systému.

- Implicitní (*implicit grant*)

  Implicitní grant je zjednodušením mechanismu autorizačního kódu. Místo toho, aby klient získal, po přihlášení uživatele, autorizační kód, obdrží přímo přístupový token. Na rozdíl od předchozího autorizačního grantu, u implicitního grantu klient neobdrží obnovovací token.

- Klientský přístup (*client credentials*)

  Tento mechanismus, v literatuře jej někdy můžeme nalézt pod termínem *service account*, neslouží pro autorizaci uživatele, nýbrž klienta. Klient během registrace obdrží přístupové údaje *client_id* a *client_sercet*, díky nímž bude moci od autorizačního serveru získat přístupový token. Tento mechanismus se hodí v případech, kdy potřebujeme, aby nějaká služba mohla, bez vědomí uživatele, pracovat s chráněnými zdroji jiné služby.

- Přihlašovací údaje uživatele (*resource owner password credentials grant*)

  Tento mechanismus ověření uživatele je shodný s tím, který známe z běžných aplikací. Uživatel zadává své přihlašovací údaje přímo do klientské aplikace, která je následně odešle autorizačnímu serveru. Jak může být z uvedeného patrné, tento způsob ověření je nejméně bezpečný a není doporučeno jej používat.

2.3.1.5 Podpora introspekce

V základní specifikaci OAuth 2.0 frameworku není určeno, jakým způsobem budou služby (resource server) ověřovat platnost obdržených přístupových tokenů. U menších systémů, kde typicky obě aplikace sdílí jednu databázi, je

tento problém vcelku snadno řešitelný. Ovšem u větších systémů, kde se tyto aplikace často nacházejí na jiných serverech, se řešení tohoto problému začíná značně komplikovat. Naštěstí tímto problémem se již několik lidí zabývalo a díky tomu vzniklo rozšíření původního protokolu (nazvané *OAuth 2.0 Token Introspection*), které definuje rozhraní, pomocí nějž může služba získat informace o obdrženém přístupovém tokenu.

Specifikace tohoto rozšíření definuje nový chráněný datový zdroj, jenž očekává POST požadavek, jehož obsahem bude parametr *token*. V parametru *token* uvede služba přístupový token, o kterém chce získat informace. Jakmile autorizační server příchozí dotaz zpracuje, odešle odpověď ve formátu JSON, ve které můžeme nalézt následující informace (atributy):

- *active* – udává platnost přístupového tokenu
- *client_id* – identifikátor klienta, kterému byl token přidělen
- *scope* – seznam privilegií, kde jsou jednotlivá privilegia oddělena mezerou
- *username* – identifikátor uživatele, pro kterého byl token vytvořen
- *exp* – časová značka indikující datum a čas expirace tokenu

Mimo parametru *active* jsou všechny atributy volitelné. Autorizační server může do odpovědi přidat také další atributy, které jsou specifické pro jeho doménu, avšak neměly by se překrývat s těmi, jenž definuje specifikace. [14]

2.3.1.6 Rozšíření pro veřejné klienty

Veřejní klienti[3], kteří používají mechanismus přidělení autorizačního kódu, jsou vysoce náchylní na útoky spočívající v odposlechu komunikace mezi nimi a autorizačním serverem. Podaří-li se potencionálnímu útočníkovi narušit komunikaci v okamžiku, kdy dochází k přesměrování uživatele z autorizačního serveru zpět do klientské aplikace (například přepsáním handleru pro URI schéma), získává útočník autorizační kód. Naneštěstí díky tomu, že aplikace má veřejně přístupný zdrojový kód, může útočník snadno získat také klientovi přihlašovací údaje (tj. *client_id* a *client_secret*). V tu chvíli má útočník vše potřebné pro získání přístupového tokenu. Naštěstí i tento problém je již řešitelný a to pomocí rozšíření *Proof Key for Code Exchange by OAuth Public Clients* [15].

Toto rozšíření upravuje původní mechanismus pro získání autorizačního kódu a zavádí dva nové parametry *code_verifier* a *code_challenge*, které jsou náhodně generovány klientem při zahájení komunikace a které plně nahrazují nutnost parametru *client_secret*. Popišme si, pro lepší představu, průběh celé komunikace:

[3]Veřejný klient je aplikace s veřejně přístupným zdrojovým kódem – například moderní SPA (Single Page Application)

1. Klient vytvoří náhodný řetězec, složený z písmen (bez diakritiky), čísel a vybraných interpunkčních znaků (- . _ ~) o délce 43 až 128 znaků, a nazve jej *code_verifier*. Nyní, pokud to daný klient podporuje, vytvoří jeho SHA256 otisk a zakóduje jej pomocí funkce BASE64-URL-encode. Výsledný řetězec nazve *code_challenge*.

2. Klient přesměruje uživatele na autorizační server a do URL adresy připojí parametry *client_id*, *redirect_uri*, *code_challenge* a *code_challenge_method*. Poslední parametr říká, jakým způsobem klient parametr *code_verifier* zakódoval. Dle specifikace může nabývat hodnot *S256* a *plain*.

3. Autorizační server, podobně jako v předcházejícím případě, vrátí klientovi autorizační kód.

4. Klient odešle na server shodný požadavek pro získání přístupového tokenu, avšak místo parametru *code_secret* uvede parametr *code_verifier*.

5. Autorizační server ověří obdržené údaje a vrátí klientovi přístupový a obnovovací token.

Jak si můžeme povšimnout, v tuto chvíli již útočníkovi nestačí odposlechnout autorizační kód, jelikož bez parametru *code_verifier* jej nemůže použít k získání přístupového tokenu.

2.3.2 Formulace požadavků

Nyní, vybaveni znalostmi o technologii OAuth 2.0, se můžeme pustit do návrhu autorizačního serveru. Nejdříve se ale vraťme k analýze požadavků a podívejme se, jaká funkčnost se od autorizačního serveru očekává. V následujícím seznamu budou uvedeny i takové požadavky, které přímo nesouvisí s požadavky z kapitoly 1.3, ale vyplývají ze specifikace OAuth 2.0, nebo jsou kladeny zcela nezávisle.

2.3.2.1 Funkční požadavky

- **AUTH-FR1 Evidence klientů**

  Autorizační server bude evidovat seznam klientů, kteří jej budou moci využívat. U každého klienta bude určen mechanismus autorizace (tj. autorizační grant). U klientů, kteří budou používat mechanismus klientský přístup, bude server evidovat uživatelskou identitu, podle které získá klient potřebná privilegia.

- **AUTH-FR2 Evidence privilegií**

  Autorizační systém bude evidovat seznam privilegií, které budou sloužit k řízení přístupu k různým zdrojům v rámci domény. Privilegia mohou

být sdílená mezi několika službami, nebo mohou být určena výhradně pro řízení přístupu v jedné službě.

- **AUTH-FR3 Evidence uživatelů**

  Autorizační server bude evidovat informace o uživatelích. Konkrétně půjde o jméno, příjmení, uživatelské jméno, heslo, privilegia a další volitelné parametry, které bude moci nastavit oprávněná osoba.

- **AUTH-FR4 Validace přístupových tokenů**

  Autorizační server bude poskytovat rozhraní, které umožní službám získat informace o přístupových přístupových tokenech a které budou moci služby používat pro jejich ověření – viz podkapitola 2.3.1.5.

- **AUTH-FR5 Podpora veřejných klientů**

  Autorizační server bude nabízet podporu pro veřejné klienty – viz podkapitola 2.3.1.6.

- **AUTH-FR6 Programové rozhraní pro vzdálenou správu**

  Autorizační server bude poskytovat programové rozhraní, které umožní oprávněným uživatelům vzdálenou správu evidence uživatelů, privilegií a klientů.

- **AUTH-FR7 Webové rozhraní pro správu**

  Autorizační server bude poskytovat webové rozhraní, přes které bude možné spravovat všechny dostupné evidence. Toto rozhraní bude přístupné pouze administrátorům systému.

2.3.2.2 Nefunkční požadavky

- **AUTH-NFR1 Omezení autorizačních grantů**

  Autorizační server bude podporovat pouze autorizační granty klientský přístup a autorizační kód. Ostatní nebudou podporovány.

- **AUTH-NFR2 Volba technologií**

  Během tvorby webového rozhraní budou použity shodné technologie jako při vývoji komponenty frontend, jenž má na starosti kolega Bc. Oldřich Malec. Programové rozhraní bude používat shodné technologie jako bude používat komponenta skladový systém – viz podkapitola 2.4.

- **AUTH-NFR3 Oprávněný uživatel**

  Oprávněný uživatel představuje druh uživatele, který disponuje privilegii umožňující mu správu autorizačního serveru. Tyto privilegia budou vázána pouze na autorizační server, tudíž nebudou sdílena s jinou službou.

2.3.3 Návrh architektury

Jak si můžeme ze seznamu funkčních a nefunkčních požadavků povšimnout, autorizační server bude rozdělen do dvou částí: *programové rozhraní* a *webová aplikace*.

Webová aplikace bude představovat uživatelské rozhraní, které budou moci používat administrátoři systému pro správu autorizačního serveru. Aplikace bude, podobně jako komponenta frontend, používat architekturu SPA (Single Page Application) a bude vytvořena pomocí frameworku Vue.js [16]. Dále bude tato aplikace používat programové rozhraní autorizačního serveru.

Programové rozhraní bude rozhraní, pomocí kterého budou moci služby získávat či validovat přístupové a obnovovací tokeny a pomocí kterého budou moci administrátoři autorizační server spravovat. Rozhraní bude používat koncept REST API a bude vytvořeno pomocí frameworku Symfony [17]. K uložení dat bude používat databázi PostgreSQL [18].

Podrobným popisem zvolených technologií se budu zabývat v podkapitole 2.6.

2.3.4 Návrh uživatelského rozhraní

Dalším logickým krokem návrhu je návrh uživatelského rozhraní webové aplikace. Ovšem vzhledem k tomu, že se tato aplikace bude skládat z několika málo formulářů a bude ji používat pouze omezené množství uživatelů, dovolím si podrobný návrh uživatelského rozhraní vynechat. Rád bych zde ale poznamenal, že aplikace bude používat Material Design [19] vycházející z knihovny Vuetify [20].

2.3.5 Datový model

Návrh datového modelu je zachycen na obrázku B.2 v příloze. Vzhledem k tomu, že většina entit vychází ze specifikace OAuth 2.0 frameworku, dovolím si jeho podrobný popis vynechat a zaměřím se pouze na jeho specifika. V datovém modelu můžeme najít:

- Entitu *user_parameter*, ve které jsou uloženy dodatečné informace o uživateli.
- Entitu *client* a její parametr *protected*, který říká, zda je klient chráněný. Chráněný klient je takový klient, který nemůže být upraven či smazán uživateli s nižšími privilegii. Tento parametr se bude používat pouze u klíčových klientů, jejichž odstranění by mělo za následek nefunkčnost celého systému. Příkladem je klient, jenž bude představovat frontend skladového systému.
- Entity *service_account* a *web_application* reprezentující autorizační granty *klientský přístup* a *autorizační kód*.

2.3.6 Návrh programového rozhraní

V této podkapitole se budu zabývat samotným návrhem programového rozhraní. Na úvod představím privilegia, kterými se klienti budou prokazovat při přístupu ke chráněným zdrojům, a následně se budu zaobírat návrhem těchto zdrojů. V rámci této části práce uvedu pouze jejich výčet. Podrobná dokumentace rozhraní se nachází na přiloženém médiu.

2.3.6.1 Specifikace privilegií

Autorizační server definuje celkem čtyři typy privilegií:

- *auth:manage:scopes* – umožňuje přístup ke zdrojům nabízející správu privilegií
- *auth:manage:users* – umožňuje přístup ke zdrojům pro správu uživatelů
- *auth:manage:clients* – umožňuje přístup ke zdrojům pro správu klientů (mimo chráněných)
- *auth:manage:clients:protected* – umožňuje přístup ke zdrojům pro správu klientů (včetně chráněných)

2.3.6.2 Specifikace API

Tabulka 2.1: Specifikace API autorizačního serveru – OAuth 2.0

URI	**Metoda**	**Popis**
Prefix: oauth		
/token	**POST**	Získání přístupového tokenu
/authorize	**GET**	Žádost o autorizaci klienta
/introspection	**POST**	Zobrazí informace o tokenu
/logout	**GET**	Odhlásí uživatele

Tabulka 2.2: Specifikace API autorizačního serveru – privilegia

URI	**Metoda**	**Popis**
Privilegia: auth:manage:scopes		
Prefix: scopes		
/	**GET**	Zobrazí všechny privilegia
/	**POST**	Vytvoří privilegium
/{name}	**GET**	Zobrazí informace o privilegiu
/{name}	**PUT**	Upraví informace o privilegiu
/{name}	**DELETE**	Odstraní privilegium

Tabulka 2.3: Specifikace API autorizačního serveru – uživatelé

URI	Metoda	Popis
Privilegia: auth:manage:users		
Prefix: users		
/	**GET**	Zobrazí všechny uživatele
/	**POST**	Vytvoří uživatele
/{userId}	**GET**	Zobrazí informace o uživateli
/{userId}	**PATCH**	Upraví uživatele
/{userId}	**DELETE**	Odstraní uživatele
Prefix: users/{userId}		
/parameters	**GET**	Zobrazí volitelné parametry
/parameters	**POST**	Vytvoří parametr
/parameters/{name}	**GET**	Zobrazí parametr
/parameters/{name}	**PUT**	Upraví parametr
/parameters/{name}	**DELETE**	Odstraní parametr

Tabulka 2.4: Specifikace API autorizačního serveru – klienti

URI	Metoda	Popis
Privilegia: auth:manage:clients, auth:manage:clients:protected		
Prefix: clients		
/	**GET**	Zobrazí všechny klienty
/	**POST**	Vytvoří klienta
Prefix: clients/{clientId}		
/	**GET**	Zobrazí informace o klientu
/	**PATCH**	Upraví klienta
/	**DELETE**	Odstraní klienta
/refresh-secret	**POST**	Vytvoří nový tajný klíč

2.4 Skladový systém

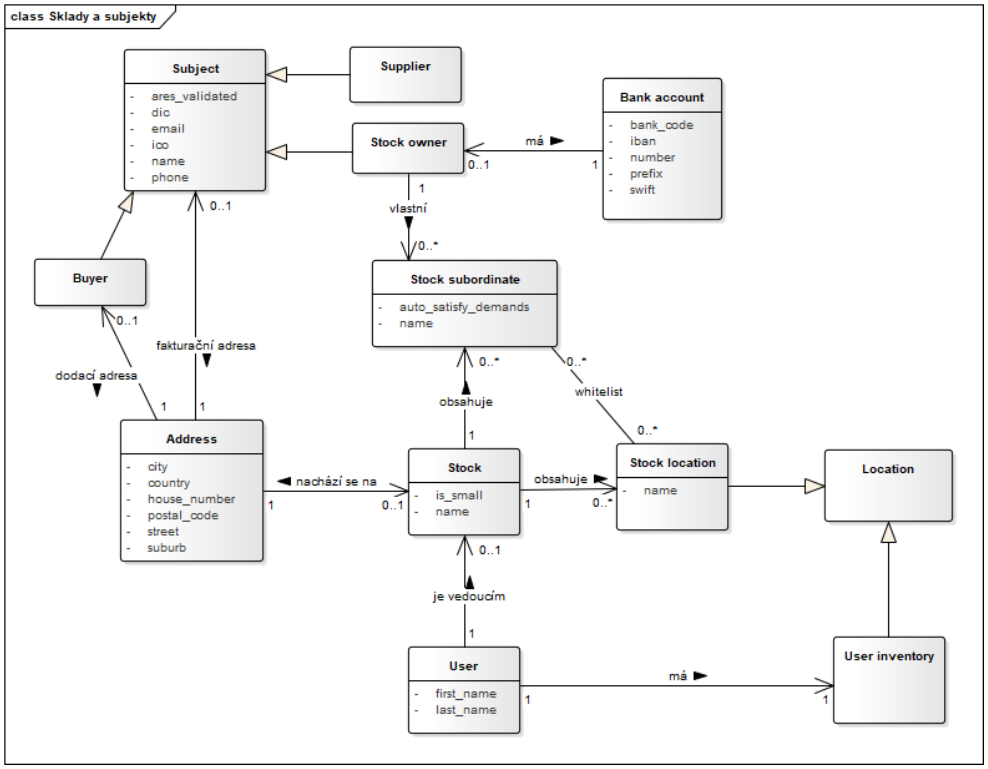
Poslední komponentou, kterou se budu v rámci této práce zabývat, je skladový systém. Cílem této komponenty bude realizovat skladovou funkčnost popsanou v podkapitole 1.3. Pojďme se tedy pustit do návrhu a začněme s doménovým modelem.

2.4.1 Doménový model

V této podkapitole se budu zabývat popisem entit (tříd), jenž souvisí s analyzovanou doménou. Tyto entity budou seskupeny do několika částí podle toho, jak spolu souvisí. Jednotlivé entity budou podrobně popsány tak, aby bylo zřejmé, jaké informace budeme ve skladovém systému uchovávat.

2.4.1.1 Evidence skladů a subjektů

V této podkapitole se budu zabývat popisem tříd souvisejících se evidencí skladů, dodavatelů, odběratelů a majitelů podskladů. Doménový model této oblasti je zachycen na obrázku 2.3.



Obrázek 2.3: Doménový model evidence skladů a subjektů

Address
Adresa, na které se bude nacházet nějaký objekt. Adresa nebude sdílená mezi vícero objekty, nýbrž bude unikátní pro každý objekt.

Název atributu	Popis
city	Město
country	Název státu
house_number	Číslo popisné
postal_code	Poštovní směrovací číslo
street	Název ulice bez čísla popisného
suburb	Předměstí (druhý adresní řádek)

Tabulka 2.5: Popis atributů třídy Address

Bank account
Bankovní účet majitele podskladu.

Název atributu	Popis
bank_code	Kód banky
iban	Mezinárodní číslo účtu (IBAN)
number	Číslo bankovního účtu
prefix	Předčíslí
swift	SWIFT kód

Tabulka 2.6: Popis atributů třídy Bank account

Subject
Abstraktní entita představující fyzickou nebo právnickou osobu.

Název atributu	Popis
ares_validated	Indikuje ověření subjektu v registru veřejné správy
dic	Daňové identifikační číslo
email	Kontaktní emailová adresa
ico	Identifikační číslo osoby
name	Jméno subjektu
phone	Mobilní spojení
website	Webová stránka subjektu

Tabulka 2.7: Popis atributů třídy Subject

Buyer
Tato entita představuje odběratele, pro kterého můžeme vyskladňovat zboží.

Supplier
Tato entita představuje subjekt, který je dodavatelem.

Stock owner
Majitel skladu, který může vlastnit nějaké podsklady.

Stock
Sklad, ve kterém se nachází skladová umístění a který je členěn na podsklady.

Název atributu	Popis
is_small	Indikuje velikost skladu (malý / velký)
name	Název skladu

Tabulka 2.8: Popis atributů třídy Stock

Malé sklady nemohou obsahovat více než jedno skladové umístění a více než jeden podsklad.

Stock location
Umístění, které se nachází právě v jednom skladu.

Název atributu	Popis
name	Název skladového umístění

Tabulka 2.9: Popis atributů třídy Stock locationn

Stock subordinate
Podsklad, který představuje logické členění skladu (budovy).

Název atributu	Popis
auto_satisfy_demands	Indikuje, zda bude systém automaticky zpracovávat příchozí objednávky
name	Název podskladu

Tabulka 2.10: Popis atributů třídy Stock subordinate

User
Osoba, která používá skladový systém. Tato osoba nebude přímo uložená ve skladovém systému, nýbrž bude získána z autorizačního serveru.

Název atributu	Popis
first_name	Jméno uživatele
last_name	Příjmení uživatele

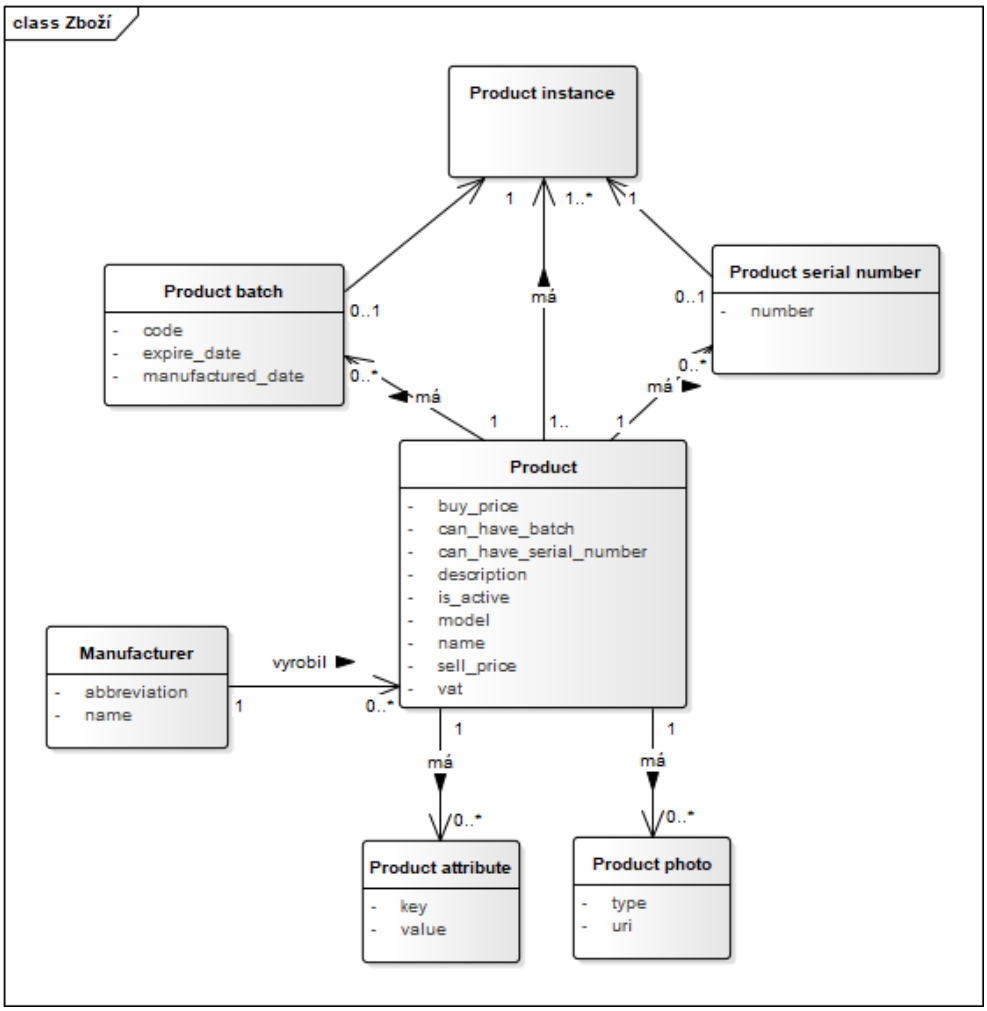
Tabulka 2.11: Popis atributů třídy User

Location
Tato entita představuje jakékoli umístění evidované skladovým systémem.

User inventory
Tato entita představuje uživatelův inventář, do kterého bude během plnění úloh vkládáno zboží. Systém bude evidovat inventář pouze pro osoby s uživatelskou rolí vedoucí nebo skladník.

2.4.1.2 Evidence zboží

V této podkapitole se budu zabývat popisem entit souvisejících s evidencí zboží. Na obrázku 2.4 je zachycen doménový model této oblasti.



Obrázek 2.4: Doménový model evidence zboží

Manufacturer
Výrobce, který vyrábí zboží.

Název atributu	Popis
abbreviation	Zkratka výrobce
name	Název výrobce

Tabulka 2.12: Popis atributů třídy Manufacturer

Product
Skladová karta zboží, která obsahuje obecné informace o zboží.

Název atributu	Popis
buy_price	Nákupní cena bez DPH, která bude předvyplňována při naskladnění
can_have_batch	Indikuje, zda musí mít každý kus zboží určenou šarži
can_have_serial_number	Indikuje, zda musí mít každý kus zboží vlastní výrobní číslo
description	Popis zboží
is_active	Stav (aktivní nebo neaktivní)
model	Model zboží (nebo vlastní identifikátor)
name	Název zboží
sell_price	Prodejní cena bez DPH
vat	Sazba DPH

Tabulka 2.13: Popis atributů třídy Product

Product attribute
Tato třída představuje parametr zboží, který může uživatel libovolně vytvářet, upravovat či mazat.

Název atributu	Popis
key	Název parametru
value	Hodnota parametru

Tabulka 2.14: Popis atributů třídy Product attribute

Product photo
Fotografie, která patří ke zboží.

Název atributu	Popis
type	Typ fotografie (lokální nebo externí)
uri	Adresa, kde je fotografie uložena

Tabulka 2.15: Popis atributů třídy Product photo

Product batch
Tato entita představuje konkrétní šarži zboží.

Název atributu	Popis
code	Kód šarže
expire_date	Datum a čas expirace šarže
manufactured_date	Datum a čas výroby šarže

Tabulka 2.16: Popis atributů třídy Product batch

Product serial number
Tato entita představuje konkrétní výrobní číslo zboží.

Název atributu	Popis
number	Výrobní číslo

Tabulka 2.17: Popis atributů třídy Product serial number

Product instance
Konkrétní instance zboží, která může mít přiřazené výrobní číslo nebo šarži. Tato třída nepředstavuje konkrétní kus zboží, nýbrž kombinaci skladové karty zboží a výrobního čísla nebo šarže. To znamená, že:

- nebude-li mít skladová karta ani výrobní čísla ani šarže, bude existovat právě jedna instance zboží
- bude-li skladová karta obsahovat výrobní čísla, bude existovat shodný počet instancí zboží s počtem výrobních čísel
- bude-li skladová karta obsahovat šarže, bude existovat shodný počet instancí zboží s počtem šarží

2.4.1.3 Evidence čárových kódů

V této podkapitole se budu zabývat popisem entit souvisejících s evidencí čárových kódů. Doménový model je zachycen na obrázku 2.5.

Barcode
Tato entita představuje čárový kód nějakého objektu.

Název atributu	Popis
code	Čárový kód

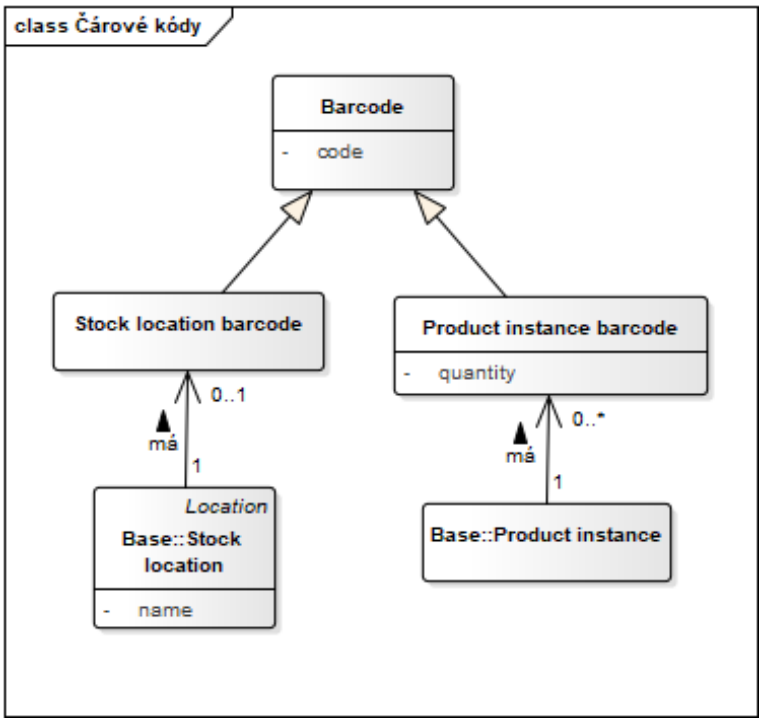
Tabulka 2.18: Popis atributů třídy Barcode

Product instance barcode
Čárový kód konkrétní instance zboží.

Název atributu	Popis
quantity	Počet kusů zboží, který bude zpracován po načtení kódu

Tabulka 2.19: Popis atributů třídy Product instance barcode

Stock location barcode
Čárový kód konkrétního skladového umístění.



Obrázek 2.5: Doménový model evidence čárových kódů

2.4.1.4 Evidence úloh

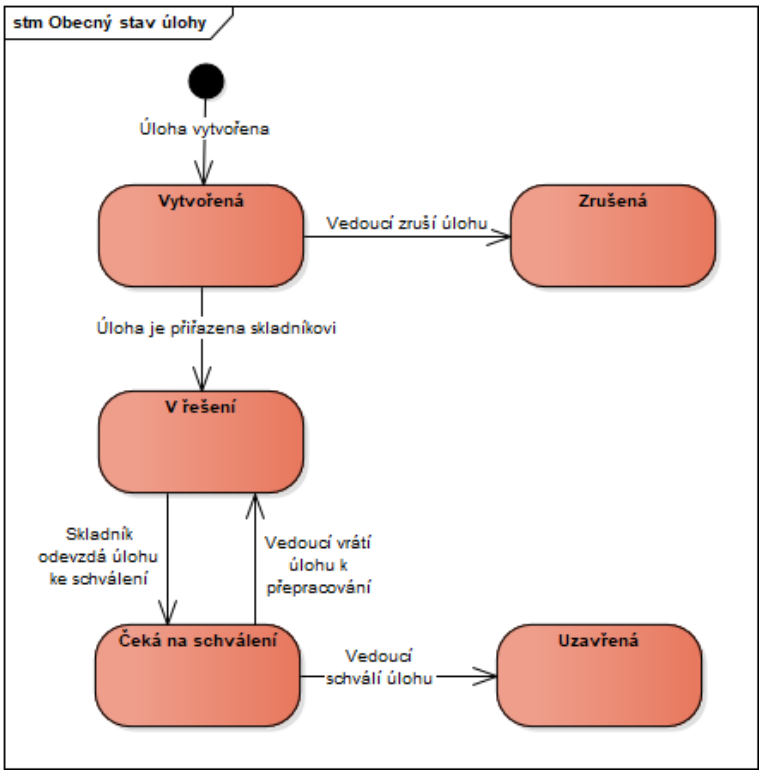
V této kapitole se budu zabývat popisem entit spojených s evidencí úloh. Na obrázku 2.7 se nachází doménový model této oblasti.

Task
Abstraktní třída představující instanci jedné úlohy.

Název atributu	Popis
description	Popis úlohy
state	Stav úlohy
priority	Priorita úlohy (nízká, normální nebo vysoká)

Tabulka 2.20: Popis atributů třídy Task

Každá úloha se může nacházet v různých stavech, které definuje konkrétní implementace úlohy. Obecně se bude jednat o následující stavy, které jsou popsané také na obrázku 2.6.



Obrázek 2.6: Stavový diagram pro obecnou úlohu

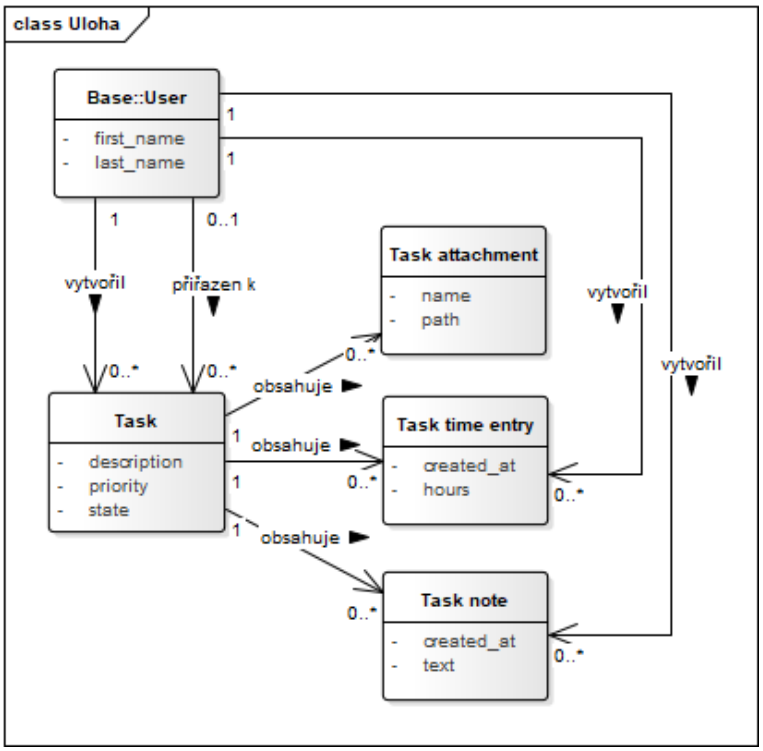
- *Vytvořená* – úloha, která byla právě vytvořena a může si ji vzít kterýkoli uživatel s rolí skladník ke zpracování.

- *Zrušená* – úloha, která byla ihned po jejím vytvoření uživatelem s rolí vedoucí zrušena.

- *V řešení* – úloha, která je rozpracovaná. To znamená, že je přiřazena některému uživateli s rolí skladník, nebo byla vrácena k přepracování kterémukoli uživateli s rolí skladník, jenž si ji může vzít ke zpracování.

- *Čeká na schválení* – úloha, která čeká na schválení uživatelem s rolí vedoucí.

- *Uzavřená* – úloha, která již byla vyřešena a schválena.



Obrázek 2.7: Doménový model evidence úloh

Task note
Komentář vytvořený uživatelem k jedné úloze.

Název atributu	Popis
created_at	Datum vytvoření
text	Text komentáře

Tabulka 2.21: Popis atributů třídy Task note

Task time entry
Časový výkaz vytvořený uživatelem k jedné úloze.

Název atributu	Popis
created_at	Datum vytvoření
hours	Počet hodin strávených řešením úlohy

Tabulka 2.22: Popis atributů třídy Task time entry

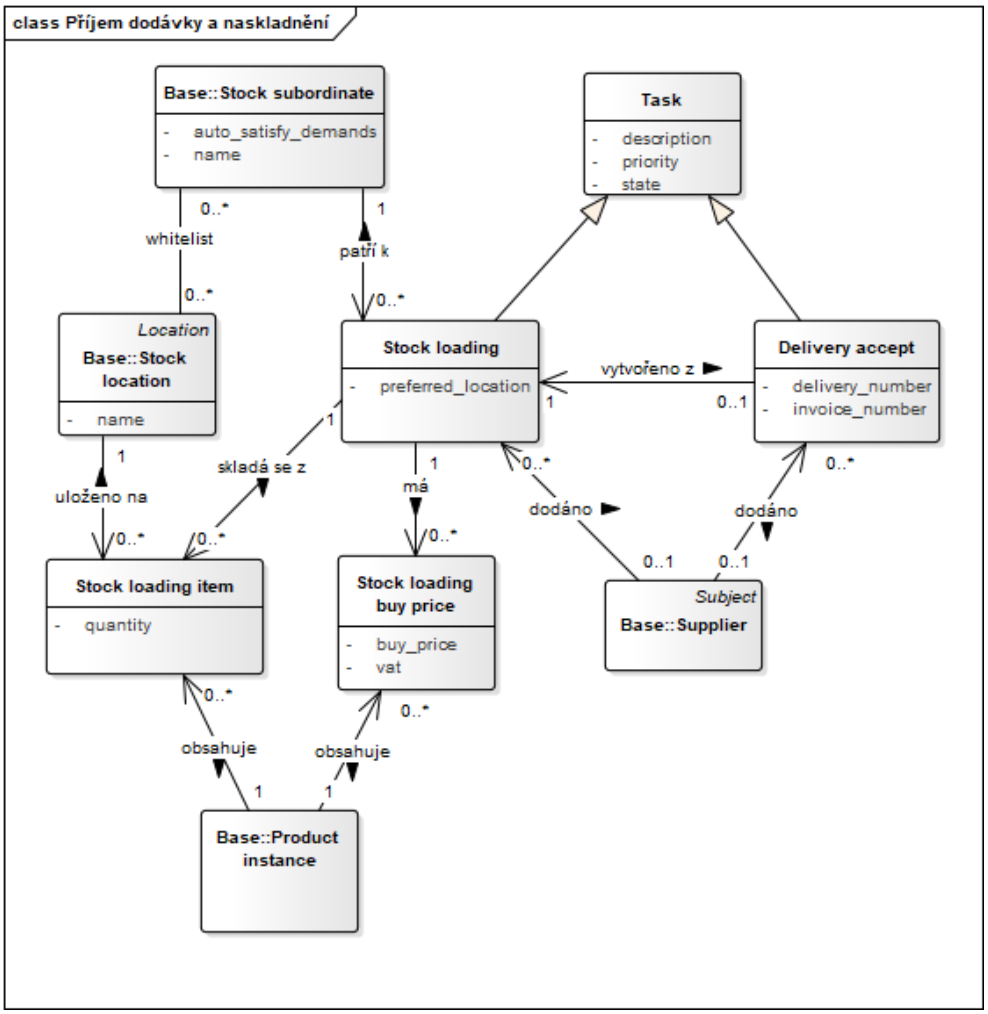
Task attachment
Příloha, která byla uživatelem připojena k jedné úloze.

Název atributu	Popis
name	Jméno přílohy
path	Cesta k datovému souboru na souborovém systému

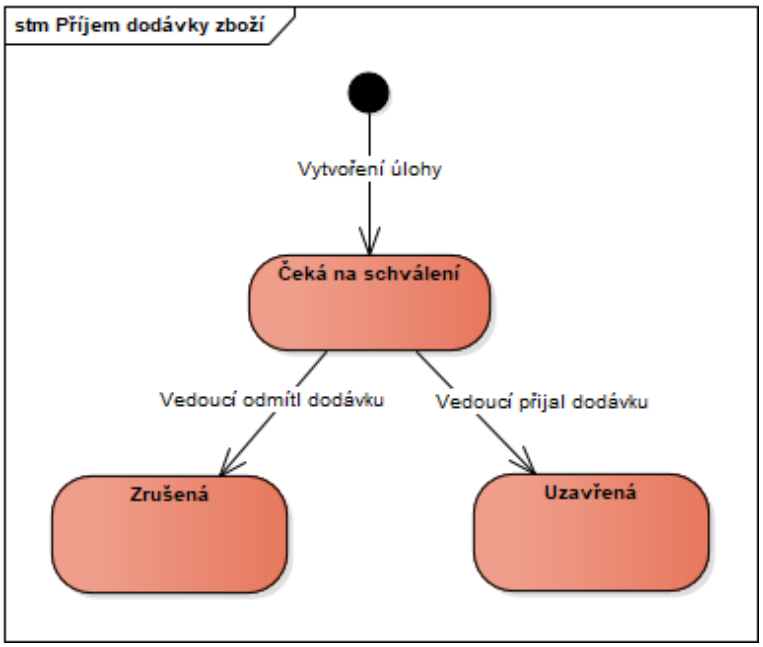
Tabulka 2.23: Popis atributů třídy Task attachment

2.4.1.5 Naskladnění a příjem dodávky zboží

V této podkapitole se budu zabývat popisem entit spojených s úlohou naskladnění a příjem dodávky zboží. Doménový model této oblasti je zachycen na obrázku 2.8.



Obrázek 2.8: Doménový model úlohy naskladnění



Obrázek 2.9: Stavový diagram úlohy příjem dodávky zboží

Delivery accept
Tato třída představuje úlohu příjem dodávky zboží.

Název atributu	Popis
delivery_number	Číslo dodacího listu obdrženého při příjmu dodávky
invoice_number	Číslo faktury obdržené při příjmu dodávky

Tabulka 2.24: Popis atributů třídy Delivery accept

Úloha se může nacházet ve třech různých stavech, které jsou popsány na obrázku 2.9. Vzhledem k tomu, že stavy této úlohy vychází z množiny stavů obecné úlohy, dovolím si jejich popis vynechat.

Stock loading
Tato třída představuje úlohu naskladnění.

Název atributu	Popis
preferred_location	Indikuje, jaká skladová umístění mají být během plnění úlohy volena

Tabulka 2.25: Popis atributů třídy Stock loading

Úloha se může nacházet v různých stavech, které se řídí dle stavů obecné úlohy zachycené na obrázku 2.6.

Stock loading item
Záznam, který uchovává informaci o tom, jaké zboží a kam bylo naskladněno.

Název atributu	Popis
quantity	Množství naskladněného zboží

Tabulka 2.26: Popis atributů třídy Stock loading item

Stock loading buy price
Záznam, který uchovává nákupní cenu k naskladněnému zboží.

Název atributu	Popis
buy_price	Nákupní cena zboží bez DPH
vat	Sazba DPH

Tabulka 2.27: Popis atributů třídy Stock loading buy price

2.4.1.6 Přesun zboží v podskladu

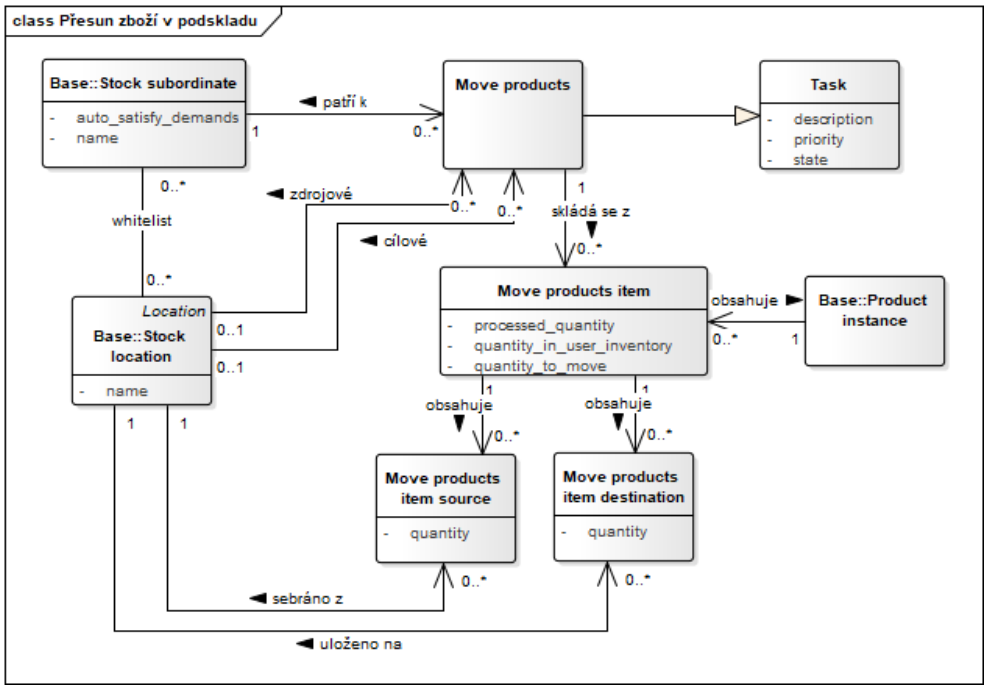
V této podkapitole se budu zabývat popisem tříd spojených s úlohou přesun zboží v podskladu. Doménový model týkající se této úlohy je zachycen na obrázku 2.10.

Move products
Tato třída představuje konkrétní instanci úlohy přesun zboží v podskladu. Úloha může nabývat několika stavů, avšak celý proces vychází ze stavového diagramu pro obecnou úlohu (viz obrázek 2.6), a proto si jej dovolím z popisu vynechat.

Move products item
Záznam, který uchovává informaci o tom, jaké zboží bude přesunuto, a dále uchovává průběžný stav přesouvaného zboží.

Název atributu	Popis
processed_quantity	Počet kusů zboží, který byl již zpracován
quantity_in_user_inventory	Počet kusů zboží, který se právě nachází v inventáři řešitele
quantity_to_move	Počet kusů zboží, který má být během úlohy zpracován

Tabulka 2.28: Popis atributů třídy Move products item



Obrázek 2.10: Doménový model úlohy přesun zboží

Move products item source

Záznam, ve kterém je uchováno zdrojové umístění, ze kterého uživatel zboží sebral.

Název atributu	Popis
quantity	Počet odebraných kusů zboží

Tabulka 2.29: Popis atributů třídy Move products item source

Move products item destination

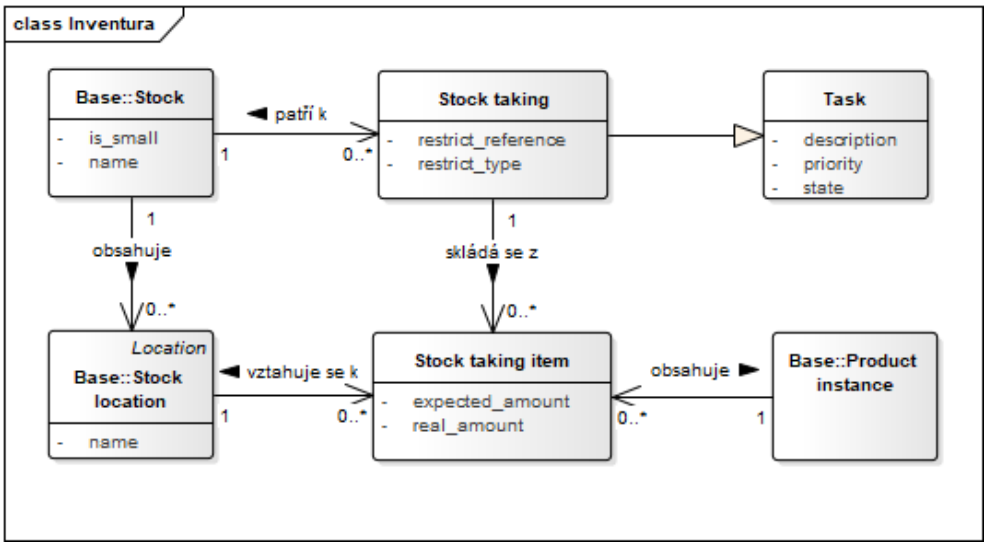
Záznam, ve kterém je uchováno cílové umístění, na které uživatel zboží uložil.

Název atributu	Popis
quantity	Počet uložených kusů zboží

Tabulka 2.30: Popis atributů třídy Move products item destination

2.4.1.7 Inventura

V této podkapitole se budu zabývat popisem entit spojených s úlohou inventura. Na obrázku 2.11 je zachycen doménový model této úlohy.



Obrázek 2.11: Doménový model úlohy inventura

Stock taking

Tato třída představuje konkrétní instanci úlohy inventura.

Název atributu	Popis
restrict_reference	Reference na objekt, podle kterého byl rozsah úlohy omezen
restrict_type	Typ omezení úlohy

Tabulka 2.31: Popis atributů třídy Stock taking

Úloha se může nacházet v různých stavech, které se řídí dle stavů obecné úlohy zachycené na obrázku 2.6. Rozsah úlohy může být omezen podle podskladu, výrobce, skladových umístění, majitele nebo skladové karty zboží.

Stock taking item

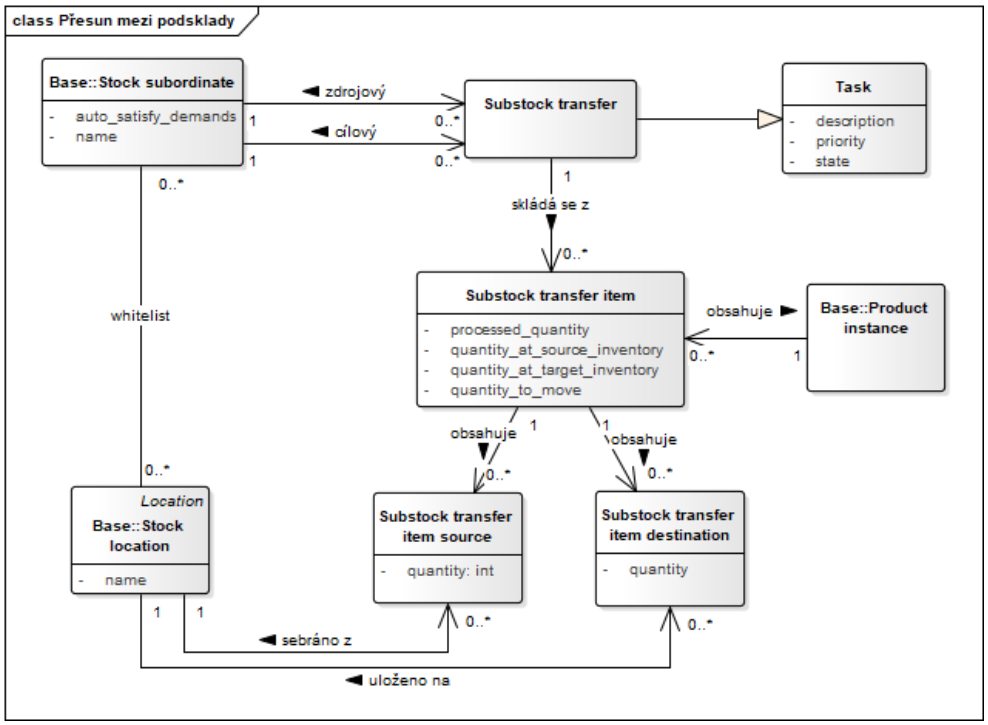
Tento záznam uchovává informaci o tom, jaké zboží se dle systému ve skladu nachází a kolik jej ve skutečnosti řešitel objevil.

Název atributu	Popis
expected_amount	Počet kusů zboží evidovaných ve skladovém systému
real_amount	Počet kusů zboží, které řešitel ve skladu nalezl

Tabulka 2.32: Popis atributů třídy Stock taking item

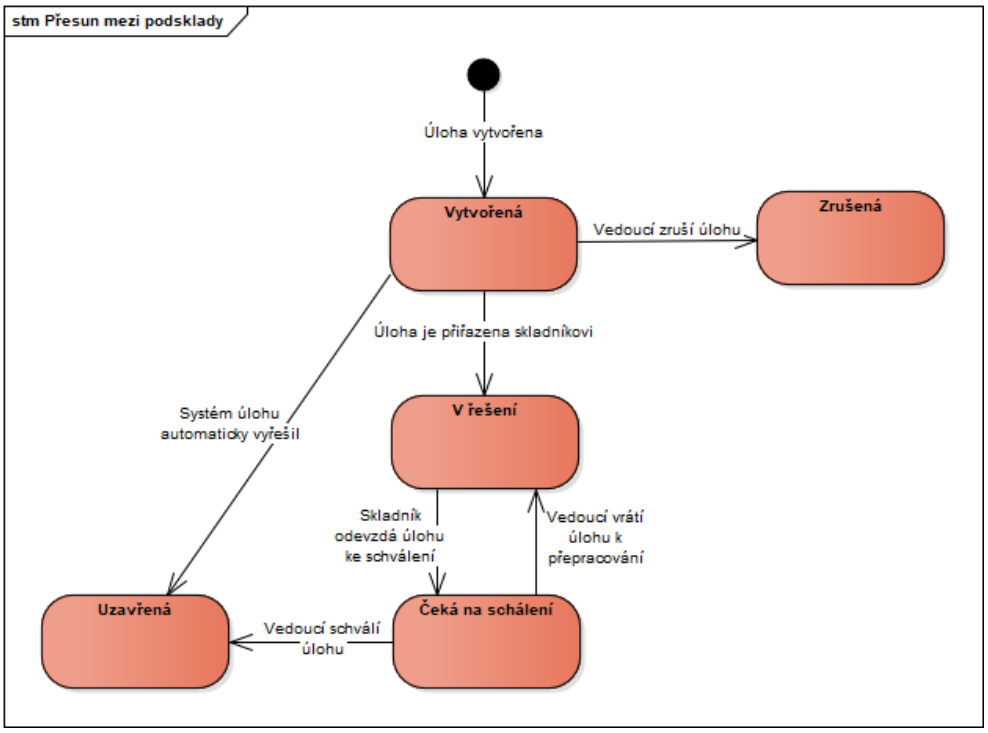
2.4.1.8 Přesun zboží mezi podsklady

V této podkapitole se budu zabývat popisem tříd spojených s úlohou přesun zboží mezi podsklady. Doménový model této oblasti je zachycen na obrázku 2.12.



Obrázek 2.12: Doménový model úlohy přesun mezi podsklady

Substock transfer
Tato třída představuje konkrétní instanci úlohy přesun mezi podsklady. Úloha se může nacházet v různých stavech, jenž jsou popsány na obrázku 2.13. Vzhledem k tomu, že tyto stavy vycházejí ze stavového diagramu obecné úlohy (viz 2.6), dovolím si jeho popis vynechat.



Obrázek 2.13: Stavový diagram úlohy přesun mezi podsklady

Substock transfer item
Záznam, který uchovává informace o tom, jaké zboží se bude v rámci úlohy přesouvat. Dále uchovává průběžný stav řešení úlohy.

Název atributu	Popis
processed_quantity	Počet již zpracovaných kusů zboží
quantity_at_source_inventory	Počet kusů zboží, které má řešitel v inventáři a které ještě nebyly přesunuty do cílového podskladu
quantity_at_target_inventory	Počet kusů zboží, které má řešitel v inventáři a které již byly přesunuty do cílového podskladu
quantity_to_move	Počet kusů zboží, který bude v rámci úlohy přesunut

Tabulka 2.33: Popis atributů třídy Substock transfer item

Substock transfer item source
Záznam, ve kterém je uchováno umístění ve zdrojovém podskladu, ze kterého řešitel zboží sebral.

Název atributu	Popis
quantity	Počet odebraných kusů zboží

Tabulka 2.34: Popis atributů třídy Substock transfer item source

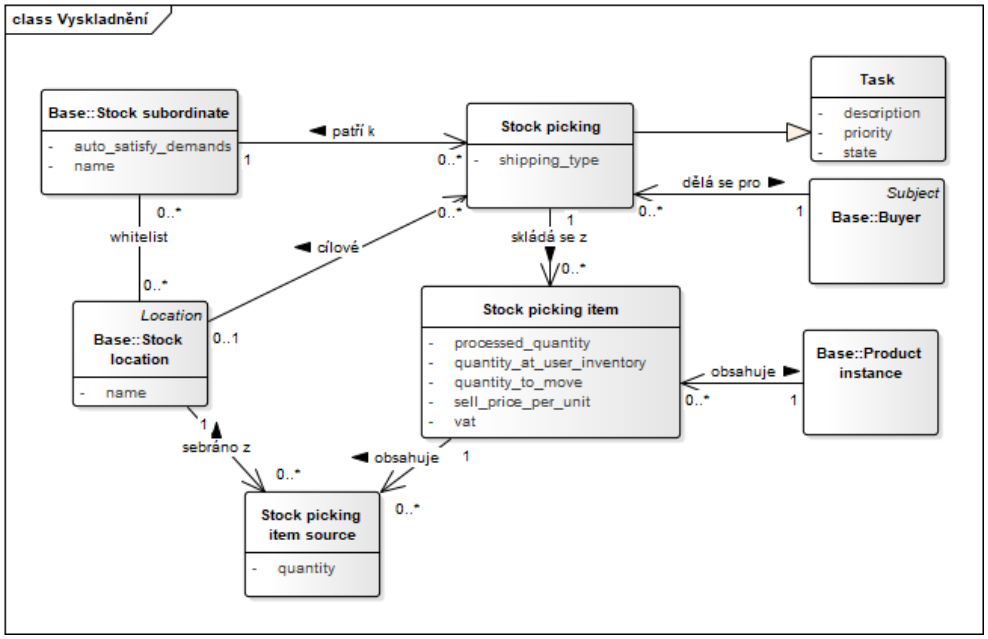
Substock transfer item destination
Záznam, ve kterém je uchováno umístění v cílovém podskladu, na které řešitel zboží umístil.

Název atributu	Popis
quantity	Počet uložených kusů zboží

Tabulka 2.35: Popis atributů třídy Substock transfer item destination

2.4.1.9 Vyskladnění

V této podkapitole se budu zabývat popisem entit spojených s úlohou vyskladnění. Na obrázku 2.14 je zaznamenán příslušný doménový model.



Obrázek 2.14: Doménový model úlohy vyskladnění

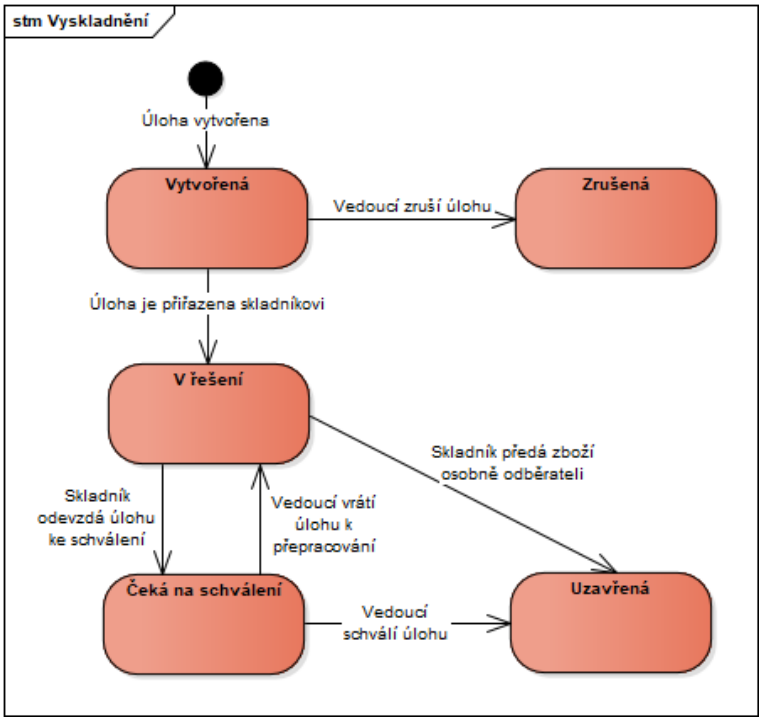
Stock picking

Tato třída představuje konkrétní instanci úlohy vyskladnění.

Název atributu	Popis
shipping_type	Způsob předání (osobní předání nebo zásilka)

Tabulka 2.36: Popis atributů třídy Stock picking

Úloha se může nacházet v různých stavech, které jsou zachyceny na obrázku 2.15. Vzhledem k tomu, že jsou tyto stavy shodné se stavy u obecné úlohy (viz 2.6), dovolím si jejich popis vynechat. Rozdíl je zde pouze v přechodu mezi stavy *v řešení* a *uzavřená*, který symbolizuje osobní předání zboží odběrateli.



Obrázek 2.15: Stavový diagram úlohy vyskladnění

Stock picking item

Záznam, který uchovává informaci o tom, jaké zboží bude vyskladněno. Dále uchovává informace o průběžném řešení úlohy.

Název atributu	Popis
processed_quantity	Počet již zpracovaných kusů zboží
quantity_at_user_inventory	Počet kusů zboží, které má řešitel ve svém inventáři
quantity_to_move	Počet kusů zboží, který bude v rámci úlohy vyskladněn
sell_price_per_unit	Prodejní cena bez DPH
vat	Sazba DPH

Tabulka 2.37: Popis atributů třídy Stock picking item

Stock picking item source
Záznam, ve kterém je uchováváno zdrojové umístění sebraného zboží.

Název atributu	Popis
quantity	Počet odebraných kusů zboží

Tabulka 2.38: Popis atributů třídy Stock picking item source

2.4.1.10 Evidence zásilek a úloha příprava zásilky k odeslání

V této podkapitole se budu zabývat popisem tříd souvisejících s evidencí zásilek a úlohy příprava zásilky k odeslání. Doménový model je zachycen na obrázku 2.17.

Carrier
Dopravce, který převáží zásilky.

Název atributu	Popis
active	Stav (aktivní nebo neaktivní)
name	Jméno dopravce

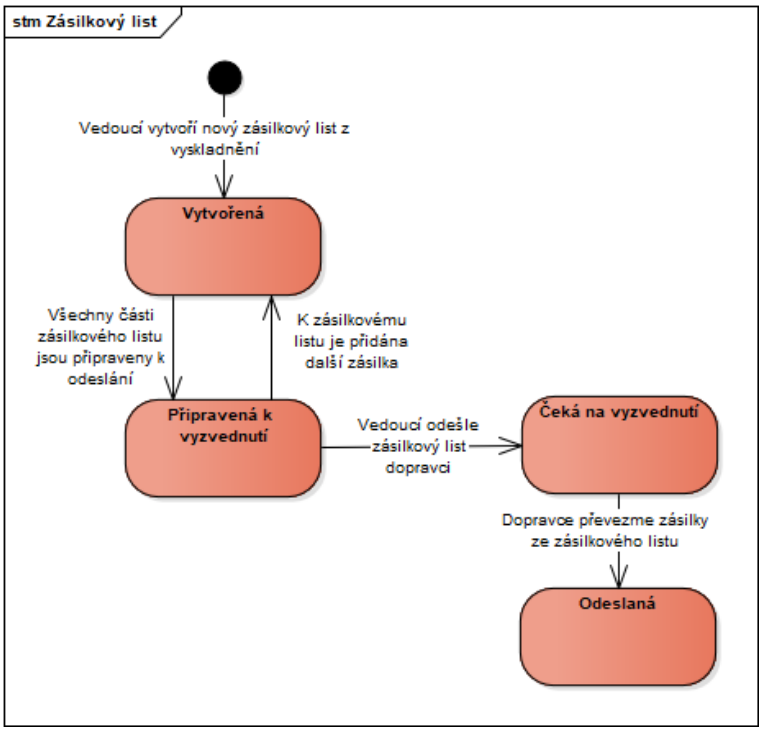
Tabulka 2.39: Popis atributů třídy Carrier

Shipment
Zásilkový list obsahující seznam zásilek, které budou předány dopravci.

Název atributu	Popis
sent_date	Datum a čas vyzvednutí přepravcem
state	Stav zásilkového listu

Tabulka 2.40: Popis atributů třídy Shipment

Každý zásilkový list se může nacházet v jednom z následujících stavů. Pro lepší představu je přechod mezi těmito stavy zachycen na obrázku 2.16.



Obrázek 2.16: Stavový diagram zásilkového listu

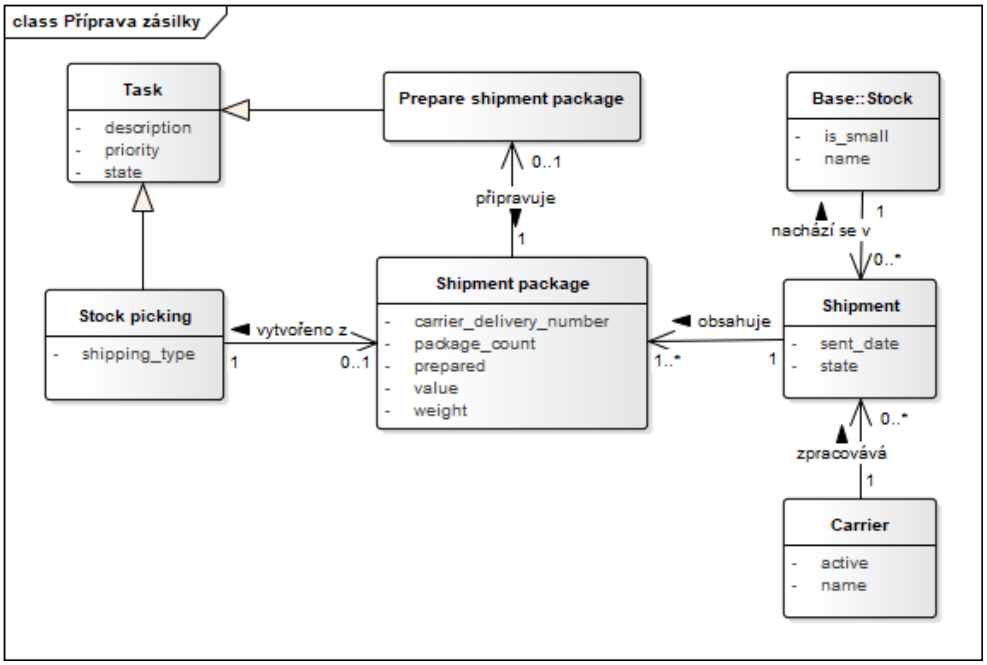
- *Vytvořeno* – zásilkový list, jenž obsahuje alespoň jednu zásilku, která není připravená k odeslání.
- *Připraveno k vyzvednutí* – zásilkový list, jenž obsahuje pouze zásilky, které jsou připravené k odeslání.
- *Čeká na vyzvednutí* – zásilkový list, u kterého již proběhlo předání informací dopravci.
- *Odeslaná* – zásilkový list, jehož zásilky již byly zpracovány (tj. odvezeny) dopravcem.

Do budoucna se počítá s rozšířením množiny těchto stavů, avšak to již bude záviset na konkrétních implementacích napojení na rozhraní příslušných dopravců.

Shipment package
Zásilka, která se nachází na zásilkovém listu a který byla vytvořena z úlohy vyskladnění.

Název atributu	Popis
carrier_delivery_number	Číslo zásilky přidělené dopravcem
package_count	Množství balíků, do kterých bylo zboží zabaleno
prepared	Indikuje, zda je zásilka připravena k odeslání
value	Hodnota zásilky vypočítána z prodejních cen
weight	Celková hmotnost zásilky

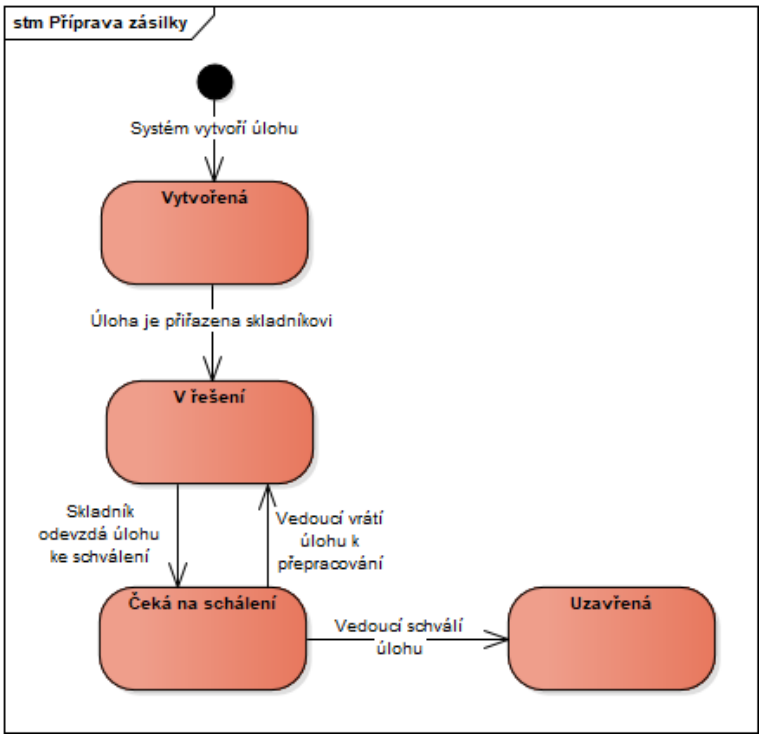
Tabulka 2.41: Popis atributů třídy Shipment package



Obrázek 2.17: Doménový model evidence zásilek a úlohy příprava zásilky k odeslání

Prepare shipment package

Konkrétní instance úlohy příprava zásilky k odeslání. Tato úloha se může nacházet v různých stavech, které jsou zachyceny na obrázku 2.18. Vzhledem k tomu, že tento diagram vychází z toho, který jsem definoval pro obecnou úlohu (viz 2.6), dovolím si jeho popis vynechat.



Obrázek 2.18: Stavový diagram úlohy příprava zásilky k odeslání

2.4.1.11 Skladové pohyby

Poslední oblastí, jejíž popisem se budu zabývat, tvoří skladové pohyby. Doménový model je zachycen na obrázku 2.19.

Stock movement
Tato třída představuje jeden konkrétní pohyb zboží po skladě.

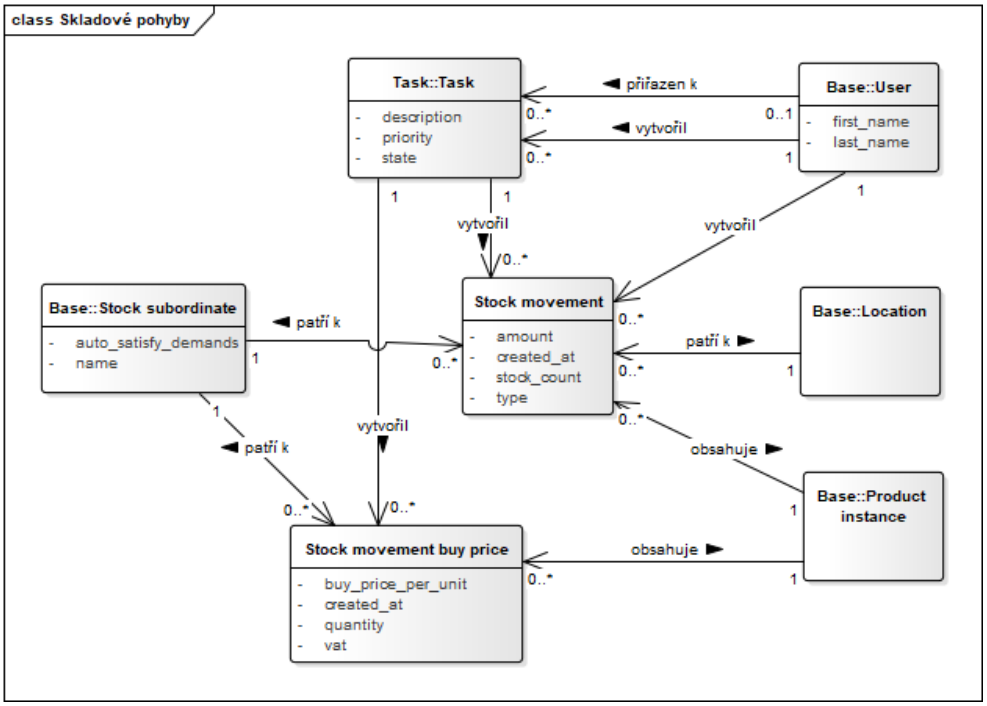
Název atributu	Popis
amount	Počet přesunutých kusů zboží
created_at	Datum a čas přesunu
stock_count	Počet kusů zboží, který na umístění zůstane po provedení přesunu
type	Typ pohybu (sebráno nebo uloženo)

Tabulka 2.42: Popis atributů třídy Stock movement

Stock movement buy price
Skladový pohyb, který slouží pro zaznamenání nákupních cen.

Název atributu	Popis
buy_price_per_unit	Nákupní cena za kus bez DPH
created_at	Datum a čas vytvoření záznamu
quantity	Počet kusů, pro které platí tato nákupní cena
vat	Sazba DPH

Tabulka 2.43: Popis atributů třídy Stock movement buy price



Obrázek 2.19: Doménový model evidence skladových pohybů

2.4.2 Návrh architektury

Komponenta bude realizována formou REST API pomocí frameworku Symfony [17]. Tato komponenta bude dále využívat služeb autorizačního serveru a služeb registru veřejné správy České republiky (ARES) [21]. Pro uchování dat bude používat databáze PostgreSQL [18], pro trvalá data, a Redis [22], pro často používaná předpočítaná data (cache). Přístup k trvalým datům bude zajištěn pomocí objektově relačního mapování (ORM), k čemuž využiji framework Doctrine [23]. Popisu zvolených technologií se budu věnovat v podkapitole 2.6.

2.4.3 Datový model

Návrh datového modelu je zachycen na obrázku B.1 v příloze. Vzhledem k tomu, že tento model vychází z doménového modelu, jenž jsem popisoval v podkapitole 2.4.1, dovolím si jeho podrobný popis vynechat a zaměřím se pouze na věci, které jsou v tomto modelu nové. Konkrétně se budeme zabývat časovými značkami a uchováním historie změn.

2.4.3.1 Časové značky

Jak si můžeme z datového modelu povšimnout, u mnoha entit přibyly atributy *created_at*, *created_by*, *updated_at* a *updated_by*. Tato změna není náhodná, nýbrž jde o řešení požadavku FR22, který byl definován v podkapitole 1.3. Do těchto atributů budou, během tvorby a úpravy příslušných objektů, ukládány informace o tom, kdy ke změně došlo a kdo změnu provedl. Bude-li změna provedena nepřihlášeným uživatelem, do atributů *updated_by* a *created_by* bude zapsána hodnota -1.

2.4.3.2 Historie změn

Historii změn můžeme uchovávat několika způsoby. Například můžeme pro každou tabulku vytvořit její kopii, kam budeme při úpravách ukládat původní data s časovou značkou, nebo můžeme vytvořit strukturu, do které budeme zaznamenávat pouze změny. Ačkoli se první řešení může zdát jako vhodnější, disponuje jistým problémem. Tento problém spočívá v rychlém růstu velikosti databáze, jelikož při každé změně musíme zduplikovat celý záznam. Naopak velkým přínosem tohoto řešení je, relativně, rychlý přístup k hodnotám záznamu, jimiž disponoval v nějaký časový okamžik.

Jak již asi tušíme, u druhé varianty je to přesně naopak. Sice omezíme růst databáze, ale znemožníme rychlý přístup k hodnotám záznamu, jimiž disponoval někdy v historii systému. Jedinou možností, jak se k těmto datům dostat, je rekonstruovat celou historii až do bodu, který potřebujeme.

Pro realizaci požadavku FR21 z podkapitoly 1.3 jsem se rozhodl vydat druhou cestou. V datovém modelu tedy můžeme najít dvě nové entity *Journal* a *Journal_details*, do nichž bude systém ukládat změny, které se chystá provést. V tabulkách níže je uvedeno, co konkrétně bude v těchto entitách zaznamenáno.

Název atributu	**Popis**
reference_id	Identifikátor změněného objektu
reference_type	Jméno změněného objektu
created_at	Datum a čas změny
created_by	Identifikátor uživatele, jenž změnu provedl

Tabulka 2.44: Popis atributů entity Journal

Název atributu	Popis
property	Jméno změněného atributu
old_value	Původní hodnota
new_value	Nová hodnota

Tabulka 2.45: Popis atributů entity Journal details

2.4.4 Návrh rozhraní

Posledním krokem, kterým se budu během návrhu komponenty zabývat, je návrh samotného programového rozhraní. Podobně jako v kapitole zabývající se návrhem autorizačního serveru, i zde začnu nejprve specifikací privilegií, jimiž se budou klienti při přístupu ke zdrojům prokazovat, a posléze se zaměřím na návrh těchto zdrojů.

2.4.4.1 Specifikace privilegií

Komponenta skladový systém definuje celkem tři privilegia:

- *ROLE_STOREKEEPER* – uživatelská role skladník
- *ROLE_CHIEF* – uživatelská role vedoucí
- *ROLE_CUSTOMER* – uživatelská role zákazník

2.4.4.2 Specifikace API

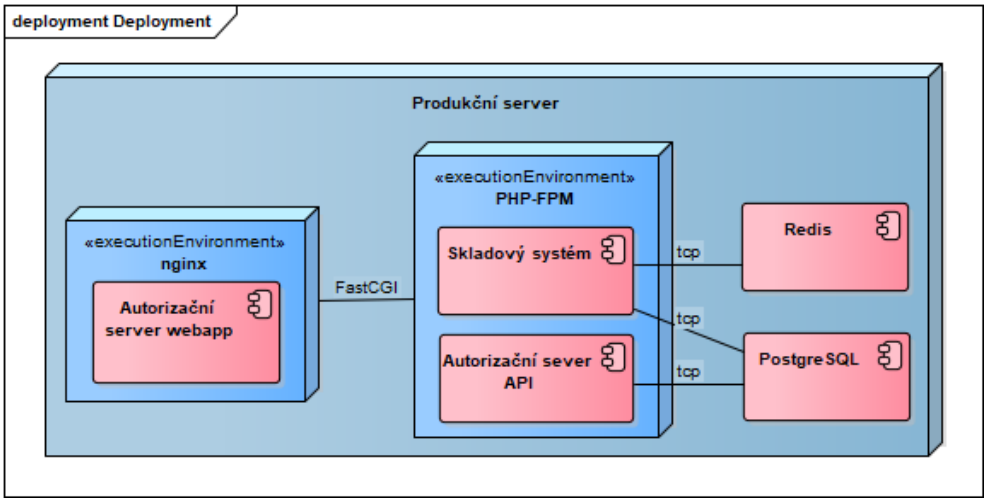
Vzhledem k tomu, že navržené rozhraní bude schopno rozeznat 203 různých API volání, dovolím si v textové části práce jejich výčet vynechat. Kompletní dokumentaci rozhraní je možné nalézt na přiloženém médiu.

2.5 Běhové prostředí

Posledním krokem návrhu je provést návrh běhového prostředí pro nově vytvářený systém. V několika minulých podkapitolách jsme se zabývali návrhem jednotlivých komponent a díky tomu nyní víme, co budou tyto komponenty pro svůj běh potřebovat. Pro rekapitulaci si dovolím připomenout, že se bude jednat o tři webové aplikace, kde dvě aplikace budou napsané v jazyce PHP [24] a jedna aplikace v jazyce JavaScript [25]. Tyto aplikace budou dále používat databázové stroje PostgreSQL [18] a Redis [22].

Nyní nám zbývá vyřešit poslední díl skládanky, a tím je volba webového serveru, jenž bude zprostředkovávat přístup k těmto aplikacím. Vzhledem k tomu, že s touto oblastí mám již jisté zkušenosti, dovolím si, na základě těchto zkušeností, bez další analýzy zvolit webový server nginx [26].

Návrh celého prostředí je, pro lepší představu čtenáře, zachycen na obrázku 2.20.



Obrázek 2.20: Diagram nasazení

2.6 Použité technologie

V této podkapitole se budu věnovat popisu zvolených technologií. Volba těchto technologií nebyla náhodná, nýbrž vycházela jednak z požadavků kladených na nový systém, mých zkušeností, ale také zohlednila technologie, které již společnost Jagu s. r. o. využívá.

2.6.1 PHP

PHP (neboli PHP: Hypertext Preprocessor) je univerzální skriptovací jazyk s otevřeným zdrojovým kódem, který je určen především pro vývoj webových aplikací. Jedná se o multiplatformní řešení, které je možné provozovat téměř na jakémkoli operačním systému, včetně operačních systémů Linux, Microsoft Windows a MacOS. První verze jazyka vznikla již v roce 1995 a od té doby se jazyk vyvíjí. Za tu dobu vznikla řada modulů, pomocí nichž lze funkčnosti jazyka dále rozšiřovat. Konkrétně se jedná například o moduly umožňující komunikaci s databázovými servery, moduly umožňující snazší práci s multimédii, souborovým systémem atp. [24]

2.6.2 Symfony

Symfony je framework s otevřeným zdrojovým kódem určený pro tvorbu webových aplikací v jazyce PHP. Předností tohoto framworku je nejen výbor-

ná dokumentace, ale také jeho architektura. Celý framework se skládá z řady malých znovupoužitelných komponent, které dohromady tvoří prostředí nejen pro vývoj webových aplikací. Tyto komponenty navíc můžeme často najít v téměř každé PHP aplikaci, protože právě díky své znovupoužitelnosti na nich často staví i jiné frameworky, které tyto aplikace používají. O vývoj celého řešení se stará společnost SensioLabs společně s více než třemi tisíci dobrovolných přispěvatelů. [17].

2.6.3 Doctrine ORM

Doctrine je knihovna nabízející podporu pro objektově relační mapování v jazyce PHP. Pro svůj běh využívá vlastní abstraktní databázovou vrstvu (DBAL), pomocí které přistupuje k fyzickým datům v databázi. Jedním z jejích klíčových rysů je možnost tvorby dotazů ve vlastním dialektu SQL nazvaném Doctrine Query Language (DQL). [23]

2.6.4 PostgreSQL

PostgreSQL je objektově-relační databázový systém s otevřeným zdrojovým kódem, který využívá a v některých ohledech i rozšiřuje jazyk SQL. Jeho vývoj probíhá již od roku 1986 a, na rozdíl od jiných řešení, pokrývá téměř celou specifikaci SQL:2011. Mezi jeho hlavní přednosti patří například podpora strukturovaných datových typů, které si dokonce může uživatel sám nadefinovat, podpora paralelního zpracování dotazů, JIT kompilace, podpora dělení tabulek do menších oddílů (Table partitioning), procedurální jazyk PL/PGSQL a mnoho dalších věcí. [18]

2.6.5 Redis

Redis je key-value databáze s otevřeným zdrojovým kódem. Výhodou tohoto řešení je skutečnost, že celá databáze běží v operační paměti. Díky tomu nabízí velice vysoký výkon, a proto bývá nejčastěji využívána jako vyrovnávací paměť. Ovšem díky podpoře perzistence ji lze použít i pro jiné účely. Na rozdíl od jednoduchých key-value databází, podporuje také operace s datovými strukturami, jako je například seznam, hešovací tabulka nebo množina, a nechybí ani podpora replikace. [22]

2.6.6 nginx

nginx je HTTP a reverzní proxy server s otevřeným zdrojovým kódem, který umožňuje směrovat jakýkoli provoz přicházející přes obecné protokoly TCP a UDP. Jeho původním autorem je Igor Sysoev, který jej vydal již v roce 2004. V současné době se o vývoj serveru stará společnost Nginx, Inc., avšak do projektu přispívá i spoustu dobrovolníků. V dubnu 2019 obsluhoval, dle společnosti Netcraft, 26,22 % nejvytíženějších stránek na internetu. [26]

2.6.7 Administrativní registr ekonomických subjektů

„*Administrativní registr ekonomických subjektů je informační systém, který umožňuje vyhledávání nad ekonomickými subjekty registrovanými v České republice. Zprostředkovává zobrazení údajů vedených v jednotlivých registrech státní správy, ze kterých čerpá data (tzv. zdrojové registry).*“ [21] Pro účely této práce budu využívat informace z veřejného obchodního rejstříku. Dále v práci bude na tuto službu odkazovat zkratka ARES.

2.6.8 JavaScript

JavaScript je skriptovací jazyk, využívaný především při tvorbě interaktivních webových stránek. Na rozdíl od jiných programovacích jazyků, které zpracovávají kód na serveru, tento jazyk je zpracováván až u klienta v okamžiku načtení webové stránky. Existují však i implementace, které jej umožňují použit i na straně serveru – typicky jde o projekty založené na technologii Node.js. První verze jazyka byla vydána již v roce 1995. Později, v roce 1997, byl jazyk standardizován společnosti ECMA a díky tomu jej můžeme v některé literatuře nalézt také pod jménem ECMAScript. [25, 27]

2.6.9 Vue.js

Vue.js je framework pro tvorbu uživatelských rozhraní a SPA aplikací v jazyce JavaScript (a jeho derivátů). Na rozdíl od jiných monolitických frameworků je navržen tak, aby jej bylo možné do již existujících projektů zapracovávat postupně. Naneštěstí díky tomu obsahuje minimum funkcí, a proto je nutné při vývoji využívat i jiné knihovny, které potřebnou funkčnost doplňují. [16]

2.6.10 Vuetify

Vuetify je knihovna implementující koncept Material Design [19] pomocí frameworku Vue.js. Knihovna nabízí řadu předpřipravených komponent, pomocí nichž lze snadno a rychle vytvořit uživatelské rozhraní jakékoli webové aplikace. [20]

Kapitola 3

Realizace

Tato kapitola bude věnována implementaci návrhu, jímž jsem se zabýval v předcházející kapitole. Vzhledem k tomu, že cílem fáze realizace je provést především implementaci navržených komponent, budou v této kapitole popsány pouze vybrané implementační detaily, jimiž jsem se v průběhu realizace systému zabýval. V závěru kapitoly se budu zabývat také možnostmi pro automatizované nasazení do reálného prostředí.

3.1 Autorizační server

Jak si můžeme z pamatovat z podkapitoly 2.3, cílem bylo vytvořit autorizační server, který bude využívat protokol OAuth 2.0 a který bude disponovat webovým a programovým rozhraním pro správu. Nyní se zaměříme na implementaci vybraných částí.

3.1.1 Implementace OAuth 2.0 protokolu

Začínat vývoj od píky je vždy komplikované a výsledné řešení s sebou často přináší mnoho chyb, na které přicházíme až při reálném provozu. Obzvláště u věcí, které se týkají bezpečnosti, je tento přístup velice nebezpečný. Z toho důvodu jsem se rozhodl neimplementovat OAuth 2.0 protokol vlastními silami, nýbrž jsem se rozhodl využít knihovnu, která tento problém již řeší. Moje volba padla na knihovnu *league/oauth2-server* [28] a to jednak z důvodu, že nabízí vše, co jsme definovali v návrhu, ale také proto, že se dle [29] jedná o často používanou knihovnu.

Velikou výhodou této knihovny je skutečnost, že jediné, co musíme při její integraci provést, je implementovat rozhraní pro manipulaci s příslušnými objekty a nasměrovat naše rozhraní na tuto knihovnu. Vše ostatní již knihovna zařídí sama. Konkrétně, pro lepší představu čtenáře, se jedná o následující rozhraní:

- *AccessTokenRepositoryInterface* – pro práci s přístupovými tokeny
- *ClientRepositoryInterface* – pro práci s klienty
- *RefreshTokenRepositoryInterface* – pro práci s obnovovacími tokeny
- *ScopeRepositoryInterface* – pro práci s privilegii
- *UserRepositoryInterface* – pro práci s uživateli
- *AuthCodeRepositoryInterface* – pro práci s autorizačními kódy

3.1.2 Uživatelské rozhraní

Pro tvorbu uživatelského rozhraní webové aplikace jsem používal výhradně prvky z knihovny Vuetify. Výsledná podoba je zachycena na několika obrázcích v příloze C.

3.2 Skladový systém

Tato část kapitoly bude věnována implementačním detailům souvisejících s komponentou skladového systému.

3.2.1 Časové značky

Jak si můžeme pamatovat z datového modelu, téměř u každé entity se nacházejí časové značky. Vzhledem k tomu, že by bylo velice neefektivní řešit každou entitu zvlášť, rozhodl jsem se implementovat rozhraní pro framework Doctrine, které tento problém bude řešit automaticky. Vytvořené rozhraní se skládá ze dvou částí a to traity *BlameableTrait* a obsluhy událostí *BlameableListener*.

V traitě *BlameableTrait* je implementováno mapování příslušných údajů na strukturu v databázi a jejich následné získání.

Obsluha událostí *BlameableListener* odposlouchává události *onPersist* a *onFlush*, vyvolané frameworkem Doctrine, a pokud objeví objekt implementující traitu *BlameableTrait*, provede aktualizaci příslušných údajů.

3.2.2 Historie změn

Podobně jako by u časových značek bylo neefektivní řešit jejich zaznamenávání odděleně pro každou entitu, ani zde by nebylo příliš efektivní řešit ukládání historie samostatně. Z toho důvodu jsem se i pro tento problém rozhodl vytvořit nové rozšíření frameworku Doctrine, které bude řešit ukládání historie automaticky. V projektu tedy vznikly dvě nové třídy *TraceableInterface* a *TraceableListener*.

Třída *TraceableInterface* představuje rozhraní, které musí entita implementovat, pokud chce ukládat svojí historii. Definice tohoto rozhraní je následující:

```
interface TraceableInterface
{
    /** Gets object id. */
    public function getId(): int;

    /**  Gets traceable name. */
    public function getTraceableName(): string;

    /**  Trace entity association changes. */
    public function traceAssociationChanges(
        Journal $journal,
        UnitOfWork $unitOfWork,
        EntityManagerInterface $em
    ): void;
}
```

Funkce *getId* vrací identifikátor objektu, funkce *getTraceableName* vrací jméno objektu, které je unikátní napříč systémem, a nakonec se zde nachází funkce *traceAssociationChanges*, která umožňuje ukládat hodnoty, jež se změnily v přidružených entitách.

Třída *TraceableListener* představuje obsluhu událostí, která odposlouchává událost *onFlush*. Jakmile je tato událost vyvolána, třída zjistí z metadat framewroku Doctrine seznam všech změněných entit a na základě tohoto seznamu vytvoří rozdílové vektory, které později uloží do databáze (do tabulek *Journal* a *Journal_details*).

3.2.3 Lokalizace

Vytvořené rozhraní podporuje výpis chybových hlášek v českém a anglickém jazyce. Klienti, kteří rozhraní používají, mohou příslušný jazyk zvolit tím, že do zasílaného požadavku připojí hlavičku *Accept-Language*, kde hodnotou bude zvolený jazyk. Vybírat mohou z hodnot *en*, pro angličtinu, nebo *cs*, pro češtinu. Zadá-li uživatel jakoukoli jinou hodnotu, systém nastaví výchozí jazyk, kterým je čeština.

3.2.4 Zjištění stavu skladu

Jak si můžeme z návrhu databáze (viz B.1) povšimnout, v databází se nenachází žádná tabulka, která by uchovávala současný stav zboží na umístění v podskladě. Důvod je prostý, tato informace bude vypočítávána za běhu z tabulky skladových pohybů. V databázi se nachází nový pohled *stock_status_view*, který má následující definici:

```
SELECT
   parent_stock_id AS stock_id,
   substock_id, location_id, instance_id,
   stock_count, sub.created_at AS last_change
FROM (
   SELECT
      instance_id, location_id, substock_id, created_at,
      stock_count,
      ROW_NUMBER() OVER (
         PARTITION BY substock_id, location_id, instance_id
         ORDER BY created_at DESC
      ) AS rn
   FROM stock_movement
) sub
LEFT JOIN stock_subordinate USING(substock_id)
WHERE sub.rn = 1
```

Z pohledu je možné zjistit, kolik kusů zboží se nachází na umístění v určitém podskladu. Dále je možné z pohledu zjistit datum poslední změny skladovosti na umístění a identifikátor skladu, kterého se záznam týká. Z důvodu zvýšení efektivity je pro tabulku *stock_movement* vytvořen index *ix_stmvt* nad atributy *(substock_id, location_id, instance_id, created_at DESC)*. Bude-li uživatel chtít získat data k nějakému datu, stačí vzít tento dotaz a omezit jeho poddotaz *sub* dle atributu *created_at*.

3.2.5 Zápis skladového pohybu

Při pohledu na tabulku *stock_movement* si můžeme povšimnout, že zápis jakéhokoli skladového pohybu bude, kvůli atributu *stock_count*, značně komplikovaný. Před vytvořením každého pohybu budeme muset nejprve zjistit, kolik kusů zboží se na umístění v daném podskladu nachází. Na toto množství aplikujeme příslušnou operaci a výslednou hodnotu připojíme k vkládanému záznamu. Ačkoli se to zdá jako jednoduchý úkol, skutečnost je, kvůli způsobu zpracování transakcí v databázovém stroji, trochu jiná.

Naneštěstí databázové stroje nezpracovávají transakce postupně, nýbrž současně. V některých případech si s tímto problém dokáží poradit automaticky tím, že záznam, který se mění, uzamknou. Jiné transakce jej tedy nemohou změnit, dokud se zámek neuvolní. Bohužel v tomto případě nemá databáze co uzamknout, protože vkládáme nový záznam, a do databáze se tedy mohou vložit nevalidní data. Naštěstí tento problém je řešitelný a to tím, že vytvoříme novou tabulku, do které budeme ukládat zámky, a před vložením každého pohybu příslušný zámek aktivujeme.

V databázi se tedy nachází nová tabulka *stock_combination_locks*, která obsahuje atributy *substock_id*, *location_id* a *instance_id*, které představují identi-

fikátor podskladu, identifikátor umístění a identifikátor instance zboží. Průběh vložení nového záznamu je následující:

1. Vytvoříme novou transakci.
2. Pokud nebude v tabulce *stock_combination_locks* existovat příslušná kombinace podskladu, umístění a instance zboží, vytvoříme ji.

```
INSERT INTO stock_combination_locks(
  substock_id, location_id, instance_id
)
VALUES (:substock_id, :location_id, :instance_id)
ON CONFLICT (substock_id, location_id, instance_id)
DO NOTHING
```

3. Vytvoříme zámek pro danou kombinaci.

```
SELECT *
FROM stock_combination_locks
WHERE substock_id = :substock_id
  AND location_id = :location_id
  AND instance_id = :instance_id
FOR UPDATE
```

4. Vložíme nový skladový pohyb.

```
WITH
  current_stock_count (stock_count) AS (
    SELECT
      COALESCE(s.stock_count, 0) AS stock_count
    FROM (
      SELECT (
        SELECT stock_count
        FROM stock_movement
        WHERE substock_id = :substock_id
          AND location_id = :location_id
          AND instance_id = :instance_id
        ORDER BY created_at DESC
        LIMIT 1
      )) s)
INSERT INTO stock_movement (
  instance_id, location_id, substock_id, type,
  created_at, created_by, amount, stock_count, task_id
)
SELECT
  :instance_id, :location_id, :substock_id, :type,
  :created_date, :created_by, :quantity,
  stock_count + :norm_quantity, :task_id
FROM current_stock_count
```

Atribut *norm_quantity* představuje hodnotu, o kterou se původní stav zboží na umístění pozmění. Při vkládání zboží na umístění je hodnota kladná, při odebírání je záporná. Atribut *quantity* je vždy kladný.

5. V případě, že již nic nebudeme měnit, transakci potvrdíme.

3.2.6 Výpis nákupních cen

Jak si můžeme z návrhu databáze povšimnout, ani v tomto případě se zde nenachází tabulka, ze které bychom mohli získat seznam nákupních cen ke zboží, jenž se právě nachází na skladě. Důvod je prostý, i v tomto případě bude systém tuto informaci vypočítávat za běhu. Informaci lze získat pomocí procedur *get_buy_prices(integer in_substock_id)* a *get_buy_prices(integer in_substock_id, integer in_istance_id)*, které přijímají identifikátor podskladu, případě identifikátor instance zboží, a které vracejí tabulku hodnot s následujícími atributy:

- *instance_id* – identifikátor instance zboží,
- *quantity* – počet kusů zboží, pro které nákupní cena platí,
- *buy_price* – nákupní cena bez DPH,
- *vat* – sazba DPH,
- *created_at* – datum a čas vytvoření záznamu,
- *task_id* – identifikátor úlohy, která záznam vytvořila.

Průběh získání nákupních cen lze popsat následujícím pseudokódem:

```
seznam_cen := [];
seznam_zbozi := vyber_seznam_zbozi_v_podskladu(:id_podskladu);
FOR item IN seznam_zbozi DO
  // Seznam serazeny sestupne dle data
  list := vyber_nakupni_ceny_v_podskladu(:id_podskladu, item);
  zbyva := item.pocet_na_sklade;
  FOR cena IN list DO
    IF cena.mnozstvi < zbyva
    THEN
      seznam_cen.push({item, cena.mnozstvi});
      zbyva := zbyva - cena.mnozstvi;
    ELSE
      seznam_cen.push({item, zbyva});
      // Ukonci vnorenou smycku
      EXIT;
    END IF;
  END;
END;
RETURN seznam_cen;
```

Volání funkce *vyber_seznam_zbozi_v_podskladu* představuje dotaz nad pohledem *stock_status_view* a volání funkce *vyber_nakupni_ceny_v_podskladu* představuje dotaz nad tabulkou *stock_movement_buy_price*.

3.2.7 Komunikace s registrem ARES

Komunikace se službou ARES je zajištěna pomocí knihovny *h4kuna/ares* [30]. Tato knihovna nabízí rozhraní, které pro zadané IČO subjektu vrátí seznam jeho údajů. Abych si usnadnil práci, obalil jsem nabízené rozhraní do vlastní třídy *AresProvider*, která navíc implementuje kešování získaných dat v databázi Redis. Výstupem tohoto rozhraní je třída *AresInfo*, ve které můžeme najít jméno subjektu, IČO, DIČ a jeho adresu.

3.2.8 Podepsané URL adresy

Podepsaná URL adresa představuje adresu, která se oproti klasické adrese liší tím, že navíc obsahuje parametr *signature* a volitelně také parametr *expire*. Pomocí parametru *signature* může server ověřit, že se adresa žádným způsobem od jejího vytvoření nezměnila. Toto řešení je vhodné použít v případech, kdy potřebujeme zajistit jistou úroveň zabezpečení, ale zároveň nemůžeme použít standardní mechanismus pomocí hlavičky *Authorization*.

V rámci komponenty skladový systém budeme tento mechanismus používat u zdroje pro zobrazení fotografií zboží. Pro zjednodušení implementace jsem zvolil knihovnu *spatie/url-signer* [31], jejíž rozhraní jsem obalil do třídy *UrlGenerator*. Rozhraní této třídy jsem dále obohatil o podporu generátoru URL adres z frameworku Symfony. Výsledkem je funkce *generateSignedUrl(string $route, array $params, int $expiration)*, která vrací URL adresu pro zadanou cestu a její parametry.

3.3 Automatizované nasazení

Nyní se podívejme na to, jakým způsobem je vyřešena distribuce vytvořených komponent. V první části nejprve popíši, jaké technologie jsem použil, a v druhé části popíši kroky, které musíme pro úspěšnou distribuci komponent podniknout.

3.3.1 Použité technologie

V této podkapitole budu popsány technologie, které se týkají procesu automatizovaného nasazení do produkčního prostředí. Tento výčet nebude úplný, nýbrž zde budou uvedeny pouze technologie, které se týkají samotného procesu nasazení. Ostatní technologie, které se týkají procesu kompilace a sestavení jednotlivých komponent, si dovolím z popisu vynechat.

3.3.1.1 Gitlab CI

Gitlab CI [32] je nástroj umožňující definovat seznam úloh, které se mají automaticky provést po nahrání změn do repositáře projektu v nástroji Gitlab [33]. Před použitím tohoto nástroje musí uživatel nejprve vytvořit v kořenovém adresáři repositáře konfigurační soubor *.gitlab-ci.yml*, do kterého uvede, v syntaxi jazyka YAML, seznam kroků, jenž chce provést, a definuje prostředí, ve kterém se budou tyto kroky provádět. Následně již uživateli stačí nahrát příslušné změny do repositáře a nástroj se o vše postará.

3.3.1.2 Deployer

Deployer [34] je nástroj pro automatizované nasazení, který, podobně jako Gitlab CI, umožňuje definovat seznam úloh, jenž se mají v průběhu nasazení provést. Na rozdíl od předchozího nástroje, tento nástroj je určen především pro nasazení projektů v jazyce PHP a nabízí řadu předpřipravených postupů (receptů), které umožňují snadno a rychle nasadit jakýkoli PHP projekt do produkčního prostředí. Jedním z těchto receptů je také recept, jenž popisuje postup nasazení jakékoli aplikace postavené na frameworku Symfony do produkčního prostředí. Jak již asi tušíme, tento recept bude, v rámci této práce, hojně využíván.

3.3.1.3 Docker

Docker [35] je virtualizační technologie, která umožňuje vytvářet a spravovat běhová prostředí pomocí takzvaných kontejnerů. Kontejner představuje balíček, který obsahuje množinu knihoven potřebných pro běh dané aplikace. Na rozdíl od tradičních virtualizačních technologií, tato technologie nevytváří zcela nový virtuální operační systém, nýbrž využívá linuxového jádra operačního systému, uvnitř kterého pracuje.

3.3.1.4 Shrnutí

V tuto chvíli již víme, jaké technologie budeme při nasazení používat, avšak zatím nevíme, jak budou tyto technologie spolupracovat. Postup bude následující.

Jakmile nástroj Gitlab CI detekuje změny v repositáři, vytvoří nové běhové prostředí pomocí nástroje Docker. V tomto nově vytvořeném prostředí začne vykonávat jednotlivé úlohy, jenž jsme definovali v souboru *.gitlab-ci.yml*. Poslední úloha se bude vždy týkat nasazení sestavené aplikace do produkčního prostředí. K tomu využijeme nástroj Deployer, který aplikaci nasadí. Jakmile nástroj Gitlab CI dokončí tuto poslední úlohu, zruší běhové prostředí v nástroji Docker.

3.3.2 Průběh nasazení

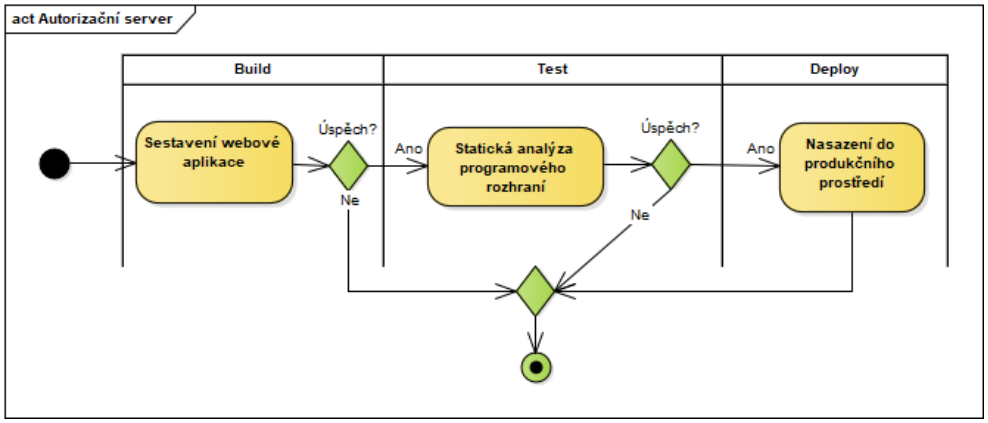
V této podkapitole se budeme zabývat popisem procesů, jenž vedou k nasazení vytvořených komponent do produkčního prostředí.

3.3.2.1 Mobilní aplikace

Na rozdíl od ostatních komponent bude mobilní aplikace distribuována manuálně a to formou zabaleného souboru APK. Ačkoli se může tato volba zdát jako nevhodná, její důvod je prostý. Do výsledné aplikace již neplánujeme přidávat další funkčnost. Z toho plyne, že aplikace bude na příslušná zařízení nahrána pouze jednou. Jistě, objeví-li se nějaké problémy, bude nutné aplikaci aktualizovat, avšak naší snahou bude před distribucí aplikaci otestovat a tuto pravděpodobnost minimalizovat.

3.3.2.2 Autorizační server

Na obrázku 3.1 je popsán proces nasazení komponenty autorizační server.



Obrázek 3.1: Průběh nasazení komponenty autorizační server

Celý proces se skládá z následujících kroků:

1. Sestavení webové aplikace

   V prvním kroku dochází k sestavení webového rozhraní aplikace pomocí sekvence příkazů *yarn install*, který nainstaluje potřebné závislosti, a *yarn run encore production*, který aplikaci sestaví pro produkční prostředí. Do dalšího kroku procesu vstupuje již sestavená aplikace.

2. Statická analýza programového rozhraní

   V této fázi dochází ke statické analýze programového rozhraní pomocí nástroje PHPStan [36]. Podrobnosti viz kapitola 4.

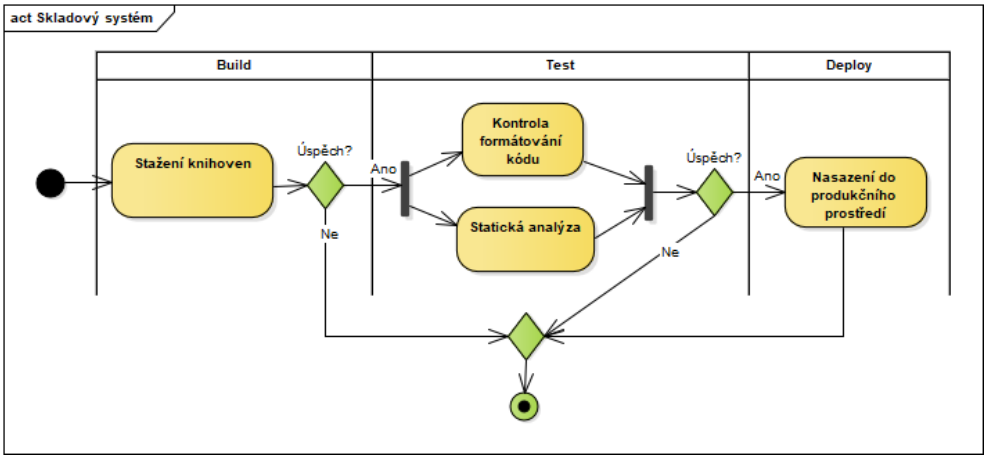
3. Nasazení do produkčního prostředí

   V tomto kroku dochází k nasazení webového i programového do produkčního prostředí pomocí nástroje Deployer. K zajištění správného nasazení používám recept Symfony[4], který je dále obohacen o funkčnost nutnou pro nasazení webové aplikace. Konkrétně jde o jednoduchou úlohu popsanou následujícím kódem:

```
task('deploy:frontend', function () {
    upload(
      './../public/build',
      '{{release_path}}/public'
    );
});
```

3.3.2.3 Skladový systém

Na obrázku 3.2 je popsán proces nasazení komponenty skladový systém.



Obrázek 3.2: Průběh nasazení komponenty skladový systém

Celý proces se skládá z následujících kroků:

1. Stažení knihoven

   V prvním kroku dochází ke stažení knihoven potřebných pro spuštění následných testů. Tuto úlohu má na starosti nástroj Composer [37], který spustíme příkazem *composer install*. Jak již asi tušíme, výstup této úlohy slouží jako vstup do následujících úloh.

[4]Zdrojový kód receptu je dostupný na `https://github.com/deployphp/deployer/blob/master/recipe/symfony.php`

2. Kontrola formátování kódu

   V tomto kroku dochází k ověření správnosti formátování zdrojového kódu vůči kódovacímu standardu. Tímto problémem se budeme podrobněji zabývat v kapitole 4.

3. Statická analýza

   V této úloze dochází ke statické analýze zdrojového kódu aplikace pomocí nástroje PHPStan [36]. Tomuto nástroji se budeme podrobněji věnovat v kapitole 4.

4. Nasazení do produkčního prostředí

   Posledním krokem je nasazení aplikace do produkčního prostředí. Pro tento účel používám nástroj Deployer, který aplikaci nasadí dle předpřipraveného receptu Symfony[4].

3.4 Současný stav

V současnou chvíli je vytvořený systém nasazen pouze v testovacím prostředí, kde čeká na dokončení klientské části. Jakmile bude tato část dokončena, systém bude zcela připraven na nasazení do ostrého provozu. Do budoucna je cílem implementovat chybějící požadavky a dále se věnovat rozvoji vytvořeného systému.

Kapitola 4

Testování

V této kapitole se budu věnovat testování vytvořeného prototypu. Jelikož jsem se tématem testování webových aplikací již zabýval ve své bakalářské práci [38], budu se na tuto práci v některých částech textu odkazovat.

4.1 Automatizované testování

Jak jsme si mohli povšimnout v podkapitole 3.3, systém je pokryt několika druhy testů. Naneštěstí, v době psaní práce, tyto testy cílí pouze na statickou analýzu kódu a již netestují správnost chování aplikace. Důvodem je skutečnost, že příprava těchto testů je velice časově náročná. Do budoucna však s tímto druhem testů počítám. Systém bude pokryt následujícími druhy testů.

4.1.1 Testování formy zdrojového kódu

Testování formy zdrojového kódu se zabývá kontrolou přehlednosti a čistoty kódu. K ověření jeho správnosti se využívá takzvaný kódovací standard, který dle [38] představuje „*sadu doporučení, které říkají, jak by měl být zdrojový kód napsán a strukturován. Příkladem může být doporučení říkající, jak bychom měli pojmenovávat proměnné, respektive jestli mají začínat malým či velkým písmenem.*“

V této práci budu pro ověření formy zdrojového používat nástroj *squizlabs/-PHP_CodeSniffer* [39], který jej bude validovat dle kódovacího standardu z mé bakalářské práce [38]. Pro zasvěcení čtenáře se jedná o kódovací standard, jenž vychází z kódovacího standardu PSR-2 [40] a navíc je obohacený o vybraná pravidla z knihovny *slevomat/coding-standard* [41].

4.1.2 Testování obsahu zdrojového kódu

„*V rámci kontroly obsahu se zaměřujeme na programátorské umění: jak ro-*

bustní, odolný a efektivní kód programátoři dodávají.“ [42]

V této práci budu pro testování obsahu používat nástroj PHPStan [36], jehož cílem je analyzovat zdrojový kód a na základě stanovených pravidel, vycházejících ze syntaxe jazyka, jej vyhodnotit. Díky tomu je nástroj schopen objevit velké množství chyb ještě předtím, než daný kód vůbec spustíme.

4.1.3 Jednotkové testy

Cílem jednotkových testů je ověřit funkčnost vybrané části kódu nezávisle na okolním prostředí. Typicky se jedná o jednu funkci či jednu třídu. Díky tomu tomuto omezení je běh testů velice rychlý, avšak tyto testy nic neříkají o tom, zda bude systém fungovat jako celek. Z toho důvodu by se měly v systému nacházet ještě jiné testy, které jej otestují na vyšší úrovni.

4.1.4 Systémové testy

Posledním druhem testů, které se budou v systému v budoucnu nacházet, jsou systémové testy. „*Během těchto testů je aplikace ověřována jako funkční celek. Tyto testy jsou používány v pozdějších fázích vývoje. Ověřují aplikaci z pohledu zákazníka. Podle připravených scénářů se simulují různé kroky, které v praxi mohou nastat. Obvykle probíhají v několika kolech. Nalezené chyby jsou opraveny a v dalších kolech jsou tyto opravy opět otestovány.*“ [43] V kontextu naší aplikace to znamená, že tyto testy budou zaměřeny na testování programového aplikačního rozhraní (API).

4.2 Manuální testování

Během vývoje byl prototyp testován také manuálně a to jednak mnou, ale i kolegou Bc. Oldřichem Malcem, jenž vytvořená rozhraní používá ve své diplomové práci. Díky tomu se nám podařilo odchytit značné množství chyb, které jsem následně opravil.

Ve zbytku této podkapitoly bych se rád zaměřil na chybu, kterou během integrace objevil kolega Malec. Tato chyba se týkala chybné implementace protokolu *Cross-origin resource sharing* v knihovně třetí strany. Na úvod nejprve připomenu, čím se tento protokol zabývá.

4.2.1 Cross-origin resource sharing

Snad již ve většině moderních webových prohlížečů můžeme najít omezení *same-origin policy*, díky kterému nemůžeme pomocí XHR dotazů snadno získávat data z jiných domén. Ačkoli se může někomu zdát, že je tento mechanismus spíše překážkou, opak je pravdou. Toto omezení představuje důležitý bezpečnostní prvek, díky němuž nemohou škodlivé skripty přistupovat k jiným

doménám, na kterých by mohly napáchat škody. Jak ale docílíme toho, že budeme moci v naší aplikaci přistupovat i ke službám na jiných doménách?

Řešení představuje takzvaný protokol *Cross-origin resource sharing*, dále jen CORS, který definuje mechanismus, pomocí nějž může webový prohlížeč zjistit, zda může ze současné domény přistupovat ke službám na jiné doméně. Za tímto účelem definuje protokol několik HTTP hlaviček, avšak nás budou zajímat pouze některé a ty jsou popsány níže:

- *Origin* – pomocí této hlavičky webový prohlížeč informuje server o tom, odkud přichází (tj. z jaké domény).
- *Access-Control-Request-Method* – pomocí této hlavičky webový prohlížeč informuje server o tom, jakou HTTP metodu se chystá použít.
- *Access-Control-Request-Headers* – pomocí této hlavičky webový prohlížeč informuje server o tom, jaké HTTP hlavičky bude dotaz obsahovat.
- *Access-Control-Allow-Origin* – pomocí této hlavičky server informuje webový prohlížeč o tom, jaké domény mohou s nabízenou službou pracovat. V této hlavičce můžeme najít buď doménu, kterou server získá z hlavičky *Origin*, nebo *, což znamená, že se službou může pracovat kdokoli.
- *Access-Control-Allow-Methods* – pomocí této hlavičky server informuje webový prohlížeč o tom, jaké HTTP metody může používat.
- *Access-Control-Allow-Headers* – pomocí této hlavičky server informuje webový prohlížeč o tom, jaké HTTP hlavičky může používat.

Samotná komunikace se skládá ze dvou částí: *preflight* dotazu a dotazu, který chce webový prohlížeč službě poslat. Předpokládejme, že chceme z domény *client.com* poslat POST dotaz na stránku *server.com/hello*. Průběh komunikace bude následující:

1. Webový prohlížeč (dále jen klient) odešle na adresu *server.com/hello* dotaz s metodou OPTIONS, ve kterém uvede hlavičku *Origins* s hodnotou *client.com*. Dále uvede hlavičky *Access-Control-Request-Method* a *Access-Control-Request-Headers* s příslušnými hodnotami.
2. Server odešle klientovi odpověď, ve které uvede hlavičku *Access-Control-Allow-Origin* s hodnotou *client.com*, pokud je povolen, a dále uvede hlavičky *Access-Control-Allow-Methods* a *Access-Control-Allow-Headers*, ve kterých může klient nalézt povolené metody a hlavičky.
3. Klient odešle POST požadavek na sever, který vrátí příslušnou odpověď.

První dva body představují *preflight* dotaz. Poslední bod představuje standardní dotaz, jenž na službu posíláme. [44]

4.2.2 Řešení problému

V minulé podkapitole jsem popsal princip protokolu CORS. Nyní se vraťme k původnímu problému. V průběhu integrace narazil kolega Malec na následující chybovou hlášku:

```
Cross-Origin Read Blocking (CORB) blocked cross-origin
response https://auth.local/oauth/introspection with
MIME type text/html.
```

Ve zkratce jednou z vlastností technologie CORB je zabránit prohlížeči v načtení dotazů, které definují jiný typ obsahu, hlavička *Content-Type*, než ve skutečnosti obsahují. Bohužel až po dlouhém pátrání se mi podařilo zjistit, že problém nesouvisí s odpovědí, kterou daný zdroj odesílá, nýbrž že se jedná o chybu v knihovně *nelmio/NelmioCorsBundle* [45], kterou rozhraní využívá pro implementaci CORS protokolu.

Zmíněná knihovna, během vyřizování CORS preflight dotazu, odešle klientovi prázdnou odpověď, ke které navíc připojí HTTP hlavičku *Content-Type* s hodnotou *text/html*. V okamžiku, kdy prohlížeč, podporující CORB, tuto odpověď obdrží, zjistí, že odpověď nevyhovuje validačním pravidlům CORB a zobrazí uživateli v konzoli varování. Naštěstí oprava této chyby byla značně jednoduchá, jelikož jediné, co jsem musel udělat, bylo nastavit obsah HTTP hlavičky *Content-Type* na hodnotu *text/plain*.

4.3 Zátěžové testování

Poslední oblastí, jíž se budu v rámci této kapitoly zabývat, je zátěžové testování. Ačkoli nejčastěji si můžeme pod tímto druhem testování představit situaci, kdy simulujeme souběžný přístup mnoha uživatelů a snažíme se systém shodit, v rámci této kapitoly se budeme zabývat jiným druhem zátěžového testování. V rámci této podkapitoly se budu zabývat testováním, které bude ověřovat, jak se aplikace chová pod náporem mnoha dat. Konkrétně se zaměřím na skladové pohyby, jelikož se jedná o nejslabší místo celého systému.

4.3.1 Testovací sestava

Pro referenci veškeré testy budou provedeny na virtuálním stroji, jenž má k dispozici:

- 1 jádro procesoru Intel(R) Core(TM) i5-4278U CPU @ 2.60GHz
- 1 GB operační paměti
- SATA SSD disk
- databázi PostgreSQL 11 s výchozí konfigurací (bez JIT)

4.3.2 Průběh testování

Jak jsem se zmínil již v úvodu, tento druh testování bude zaměřen na skladové pohyby. Konkrétně se budeme zabývat pohledem *stock_status_view*, jenž jsem definoval v podkapitole 3.2.4. Pojďme se podívat, jak tento pohled reaguje při větším množství dat v tabulce *stock_movement*.

Předpokládejme, že se v databázi nachází: 1 sklad, 2 podsklady, 5 skladových umístění a 10 instancí zboží. Dále předpokládejme, že po každém naplnění databáze provedeme aktualizaci statistik pomocí příkazu *ANALYZE*.

Množství záznamů	**Výpočetní čas (ms)**
70	1,346
1 000	1,783
10 000	7,908
100 000	79,129
1 000 000	832,688
5 000 000	28 476,534

Tabulka 4.1: Časová náročnost pohledu stock_status_view v závislosti na počtu řádků v tabulce stock_movement

Jak si můžeme z tabulky 4.1 povšimnout, pohled je při větším množství záznamů nepoužitelný. Pokud bychom jej používali v reálném provozu, brzy bychom došli do stavu, kdy bude uživatel na vykonání každé operace dlouze čekat. Co ale způsobuje toto výrazné zpomalení? Pojďme se podívat na exekuční plán tohoto pohledu.

```
Nested Loop Left Join
    (cost=0.56..527733.26 rows=25046 width=28)
    (actual time=0.045..28598.461 rows=7 loops=1)
  Join Filter:
    (sub.substock_id = stock_subordinate.substock_id)
  Rows Removed by Join Filter: 1
  ->  Subquery Scan on sub
          (cost=0.56..527106.08 rows=25046 width=24)
          (actual time=0.039..28598.441 rows=7 loops=1)
        Filter: (sub.rn = 1)
        Rows Removed by Filter: 5009093
        ->  WindowAgg
              (cost=0.56..464491.05 rows=5009203 width=32)
              (actual time=0.038..28270.333 rows=5009100 loops=1)
              ->  Index Scan using ix_stmvt on stock_movement
                    (cost=0.56..351783.98 rows=5009203 width=24)
                    (actual time=0.030..25360.280 rows=5009100 loops=1)
  ->  Materialize
        (cost=0.00..1.03 rows=2 width=8)
        (actual time=0.001..0.001 rows=1 loops=7)
        ->  Seq Scan on stock_subordinate
             (cost=0.00..1.02 rows=2 width=8)
             (actual time=0.003..0.006 rows=2 loops=1)
```

Jak vidíme, dotaz nejprve vezme data z tabulky *stock_movement* dle indexu *ix_stmvt*, spočítá čísla řádků dle zadaných kritérií a nakonec výstup omezí na první řádek z každé skupiny. Ve výsledku tedy dotaz přečte celou tabulku pohybů, což je velice neefektivní.

Zkusme na to tedy jít jiným způsobem. Co kdybychom nejprve zjistili všechny přípustné kombinace instancí zboží a umístění v podskladech, a následně bychom pro tyto kombinace našli jejich stav v tabulce *stock_movement*? Eliminovali bychom tím nutnost číst celou tabulku skladových pohybů, a navíc bychom se ptali pouze na poslední hodnoty z indexu *ix_stmvt*. Zkusme tedy použít následující dotaz:

```
SELECT
    stock_id, substock_id, location_id,
    instance_id, val AS stock_count, last_change
FROM product_instance
CROSS JOIN (
    SELECT stock.stock_id, substock_id, location_id
    FROM stock_subordinate
    JOIN stock ON
      stock.stock_id = stock_subordinate.parent_stock_id
    JOIN stock_location ON
      stock_location.stock_id = stock.stock_id OR
      stock_location.type = 'USER'
) locations
JOIN LATERAL (
    SELECT stock_count AS val, created_at AS last_change
    FROM stock_movement
    WHERE
      substock_id = locations.substock_id AND
      location_id = locations.location_id AND
      instance_id = product_instance.instance_id
    ORDER BY created_at DESC
    LIMIT 1
) stock_count ON true
```

Množství záznamů	**Výpočetní čas (ms)**
70	1,049
1 000	0,997
10 000	0,761
100 000	0,906
1 000 000	1,014
5 000 000	1,255

Tabulka 4.2: Časová náročnost upraveného pohledu stock_status_view v závislosti na počtu řádků v tabulce stock_movement

Jak si můžeme povšimnout z tabulky 4.2, nový dotaz je výrazně efektivnější. Při pěti milionech záznamů v tabulce *stock_movement* dokáže dotaz za 1,3 ms zjistit stav zboží ve všech podskladech a na všech umístěních. Jistě, tento dotaz nebude takto efektivní za všech okolností. Jeho slabina se nachází ve velikosti množiny, jíž generuje kartézský součin instancí zboží a umístění v podskladech. Ovšem zamyslíme-li se nad touto operací, zjistíme, že množina, která představuje umístění v podskladech, je relativně malá, typicky v řádu desítek. Dále bychom si měli uvědomit, že nás často zajímají informace pouze z jednoho podskladu, nikoli ze všech. Ve výsledku tedy víceméně čteme tabulku instancí zboží a hledáme příslušné záznamy v indexu *ix_stmvt* nad tabulkou *stock_movement*.

Do budoucna je možné tento dotaz ještě více optimalizovat a to tím, že místo generování množiny kombinací použijeme data, která se již nacházejí v tabulce *stock_combination_locks*. V současnou chvíli je ale tento způsob dostačující.

Závěr

Cílem této práce bylo vytvořit nový backend skladového systému, který by zohledňoval funkčnost současného řešení od společnosti Jagu s. r. o. a dále by tuto funkčnost rozšiřoval o funkce, jenž můžeme znát z konkurenčních skladových systémů.

Tento cíl byl splněn. Práce se nejprve zabývala analýzou již existujících řešení, což zahrnovalo analýzu současného skladového systému Sysel od společnosti Jagu s. r. o. a několika dalších vybraných konkurenčních řešení. Na základě této analýzy vznikl seznam funkčních a nefunkčních požadavků, jenž popisuje požadavky kladené na funkčnost nového skladového systému.

Následně se práce zabývala návrhem, který byl rozdělen do dvou fází. Nejprve jsem provedl návrh architektury celého systému, z čehož vyplynulo, že se backend bude skládat ze tří komponent, a to mobilní aplikace, autorizačního serveru a samotného skladového systému. V druhé části návrhu jsem se zabýval podrobným návrhem těchto komponent.

Jakmile byl návrh hotov, zabýval jsem se jeho realizací. V rámci této fáze jsem implementoval všechny tři navržené komponenty, pro které jsem následně definoval postupy, díky nímž je možné tyto komponenty nasadit do produkčního prostředí.

Poslední fáze vývoje byla zaměřena na testování. V průběhu testování jsem se zabýval jednak ověřením funkčnosti vytvořených komponent, ale zabýval jsem se také slabými místy vytvořeného systému. Nedostatky, které jsem v průběhu testování objevil, jsem v průběhu této fáze také opravil.

Výstupem práce není pouze skladový systém, ale také další dvě aplikace, autorizační server a mobilní aplikace, které lze využívat nezávisle na skladovém systému. V současnou chvíli není vytvořený systém nasazen v reálném prostředí, avšak je na tuto možnost zcela připraven. Věřím, že až k nasazení do reálného prostředí dojde, bude vytvořený systém splňovat všechna očekávání svých uživatelů.

Literatura

[1] Produkty [online]. 2019, [vid. 25. 4. 2019]. Dostupné z: https://www.stormware.cz/produkty

[2] Řady systému POHODA 2019 [online]. 2019, [vid. 25. 4. 2019]. Dostupné z: https://www.stormware.cz/pohoda/rady

[3] Cézar pro Windows [online]. 2019, [vid. 25. 4. 2019]. Dostupné z: http://www.cezar-win.cz/uvod

[4] Pokladní systémy — Dotykačka [online]. 2019, [vid. 25. 4. 2019]. Dostupné z: https://www.dotykacka.cz/pokladni-systemy

[5] Dotykačka - Kompletní příručka (sklady) [online]. 2019, [vid. 25. 4. 2019]. Dostupné z: http://manual.dotykacka.cz/index.html?sklady.html

[6] Dotykačka - Kompletní příručka (kategorie) [online]. 2019, [vid. 25. 4. 2019]. Dostupné z: http://manual.dotykacka.cz/index.html?kategorie.html

[7] Functional and Non-functional Requirements: Specification and Types [online]. 2018, [vid. 25. 4. 2019]. Dostupné z: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types

[8] WebView — Android Developers [online]. 2019, [vid. 27. 4. 2019]. Dostupné z: https://developer.android.com/reference/android/webkit/WebView.html

[9] Control the system UI visibility — Android Developers [online]. 2019, [vid. 27. 4. 2019]. Dostupné z: https://developer.android.com/training/system-ui

[10] Intent — Android Developers [online]. 2019, [vid. 27. 4. 2019]. Dostupné z: https://developer.android.com/reference/android/content/Intent

[11] About EMDK For Android - Zebra Technologies Techdocs [online]. 2019, [vid. 27. 4. 2019]. Dostupné z: http://techdocs.zebra.com/emdk-for-android/7-3/guide/about

[12] Barcode Scanning API Programmer's Guide - Zebra Technologies Techdocs [online]. 2019, [vid. 27. 4. 2019]. Dostupné z: http://techdocs.zebra.com/emdk-for-android/7-3/guide/barcode_scanning_guide

[13] HARDT, D.: The OAuth 2.0 Authorization Framework [online]. 2012, [vid. 27. 4. 2019]. Dostupné z: https://tools.ietf.org/html/rfc6749

[14] RICHER, J.: OAuth 2.0 Token Introspection [online]. 2015, [vid. 27. 4. 2019]. Dostupné z: https://tools.ietf.org/html/rfc7662

[15] SAKIMURA, N.: Proof Key for Code Exchange by OAuth Public Clients [online]. 2015, [vid. 27. 4. 2019]. Dostupné z: https://tools.ietf.org/html/rfc7636

[16] Introduction — Vue.js [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://vuejs.org/v2/guide

[17] What is Symfony [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://symfony.com/what-is-symfony

[18] PostgreSQL: About [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://www.postgresql.org/about

[19] Material Design [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://material.io

[20] Vue.js Material Component Framework — Vuetify.js [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://vuetifyjs.com/en/

[21] ARES - Administrativní registr ekonomických subjektů [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://wwwinfo.mfcr.cz/ares/ares.html.cz

[22] Redis [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://redis.io

[23] doctrine/orm: Doctrine Object Relational Mapper (ORM) [online]. 2019, [vid. 29. 4. 2019]. Dostupné z: https://github.com/doctrine/orm

[24] PHP: Hypertext Preprocessor [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://www.php.net

[25] JavaScript — MDN [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[26] nginx [online]. 2019, [vid. 28. 4. 2019]. Dostupné z: http://nginx.org/en

[27] PEYROTT, S.: A Brief History of JavaScript [online]. 2017, [vid. 28. 4. 2019]. Dostupné z: https://auth0.com/blog/a-brief-history-of-javascript

[28] OAuth 2.0 Server - PHP, meet OAuth 2 [online]. 2019, [vid. 30. 4. 2019]. Dostupné z: https://oauth2.thephpleague.com

[29] league/oauth2-server - Packagist [online]. 2019, [vid. 30. 4. 2019]. Dostupné z: https://packagist.org/packages/league/oauth2-server

[30] h4kuna/ares [online]. 2019, [vid. 30. 4. 2019]. Dostupné z: https://github.com/h4kuna/ares

[31] spatie/url-signer [online]. 2019, [vid. 30. 4. 2019]. Dostupné z: https://github.com/spatie/url-signer

[32] Introduction to CI/CD with GitLab — GitLab [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://docs.gitlab.com/ee/ci/introduction

[33] The first single application for the entire DevOps lifecycle - GitLab — GitLab [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://about.gitlab.com

[34] Deployer – A deployment tool for php [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://deployer.org

[35] About Docker Engine — Docker Documentation [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://docs.docker.com/engine

[36] phpstan/phpstan: PHP Static Analysis Tool - discover bugs in your code without running it! [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://github.com/phpstan/phpstan

[37] Composer [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://getcomposer.org

[38] Kovář, P.: *Automatizované testování webového portálu dbs.fit.cvut.cz.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

[39] squizlabs/PHP_CodeSniffer: PHP_CodeSniffer tokenizes PHP, JavaScript and CSS files and detects violations of a defined set of coding standards. [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://github.com/squizlabs/PHP_CodeSniffer

[40] PSR-2: Coding Style Guide [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://www.php-fig.org/psr/psr-2

[41] slevomat/coding-standard: Slevomat Coding Standard for PHP_CodeSniffer complements Consistence Coding Standard by providing sniffs with additional checks [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://github.com/slevomat/coding-standard

[42] Bureš, M.; Renda, M.; Doležel, M.; aj.: *Efektivní testování softwaru*. Praha: Grada, první vydání, 2016, ISBN 978-80-247-5594-6, str. 100.

[43] Hlava, T.: Fáze a úrovně provádění testů [online]. 2011, [vid. 1. 5. 2019]. Dostupné z: http://testovanisoftwaru.cz/tag/systemove-testovani/#system

[44] Cross-Origin Resource Sharing (CORS) [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[45] nelmio/NelmioCorsBundle: Adds CORS headers support in your Symfony2 application [online]. 2019, [vid. 1. 5. 2019]. Dostupné z: https://github.com/nelmio/NelmioCorsBundle

Příloha A

Seznam použitých zkratek

API Application Programming Interface

APK Android Application Package

ARES Administrativní registr ekonomických subjektů

CI Continuous integration

CORB Cross-Origin Read Blocking

CORS Cross-Origin Resource Sharing

CSV Comma-separated values

DIČ Daňové identifikační číslo

DPH Daň z přidané hodnoty

EMDK Enterprise Mobility Development Kit

EPL Eltron Programming Language

FIFO First in, first out

HTTP Hypertext Transfer Protocol

IČO Identifikační číslo osoby

JIT Just in time (způsob kompilace)

JSON JavaScript Object Notation

PDF Portable Document Format

POST druh dotazovací metody Hypertext Transfer protokolu

PSČ poštovní směrovací číslo

REST Representational State Transfer

SPA Single Page Application

SQL Structured Query Language

SRS Software requirements specification

TCP Transmission Control Protocol

UDP User Datagram Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

XHR rozhraní XMLHttpRequest

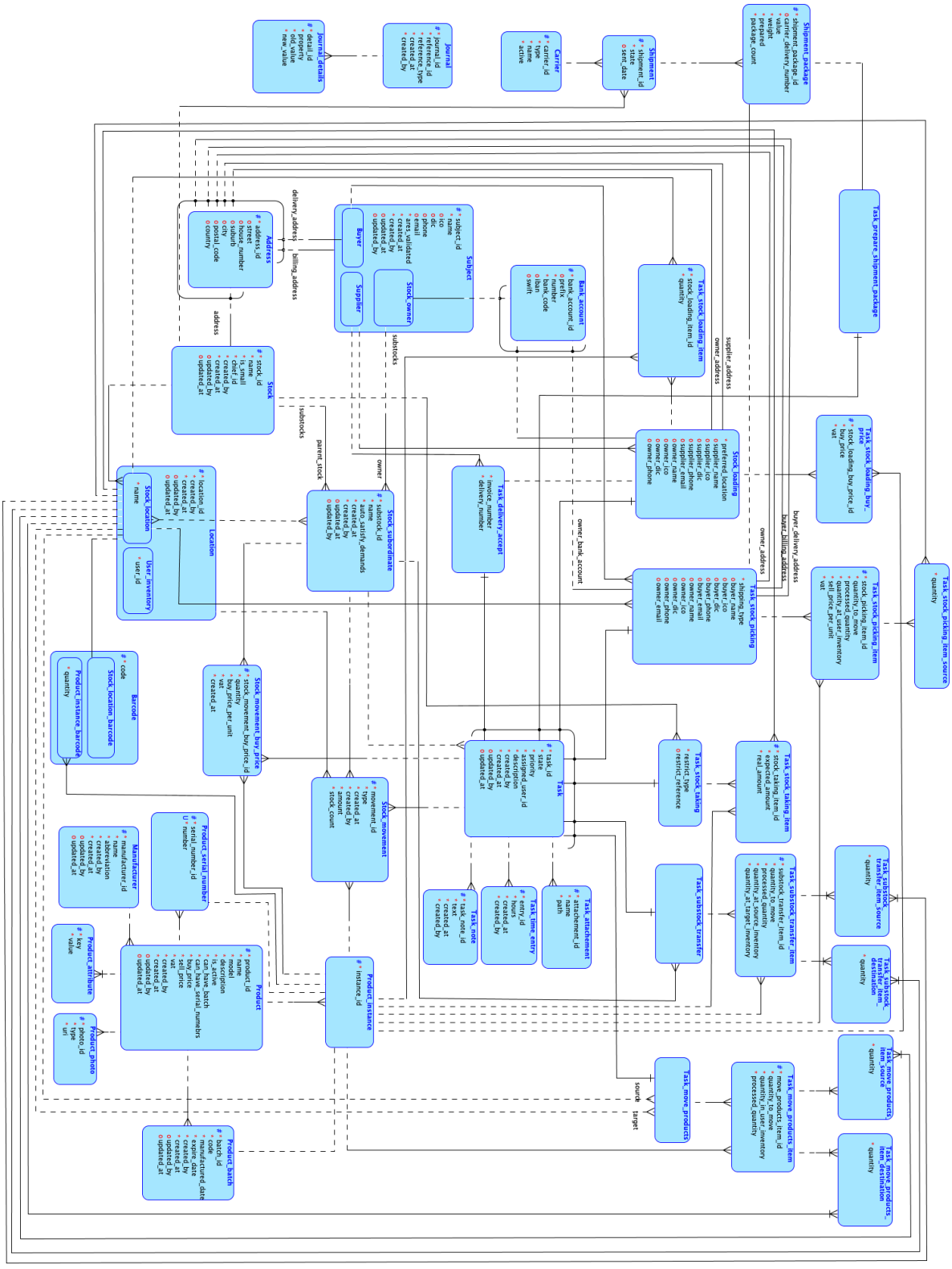
XLSX Formát tabulkového procesoru Microsoft Excel

YAML YAML Ain't Markup Language

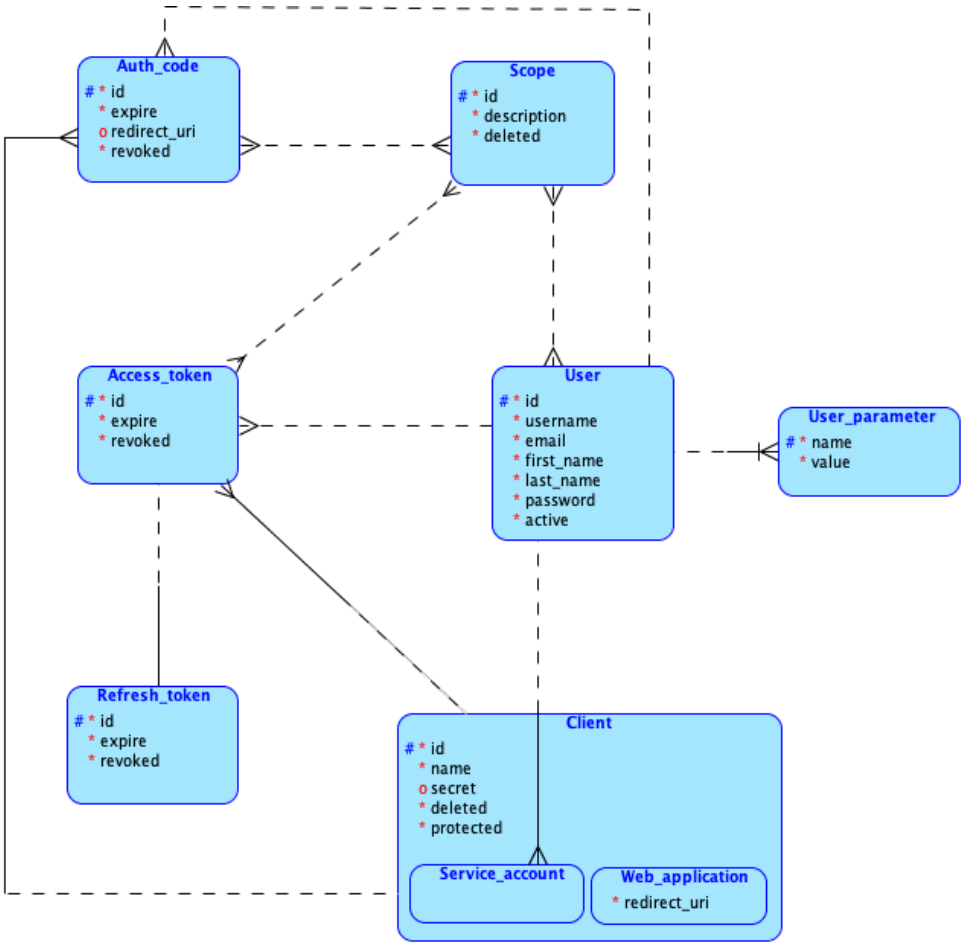
ZPL Zebra Programming Language

Příloha B

Schéma databáze



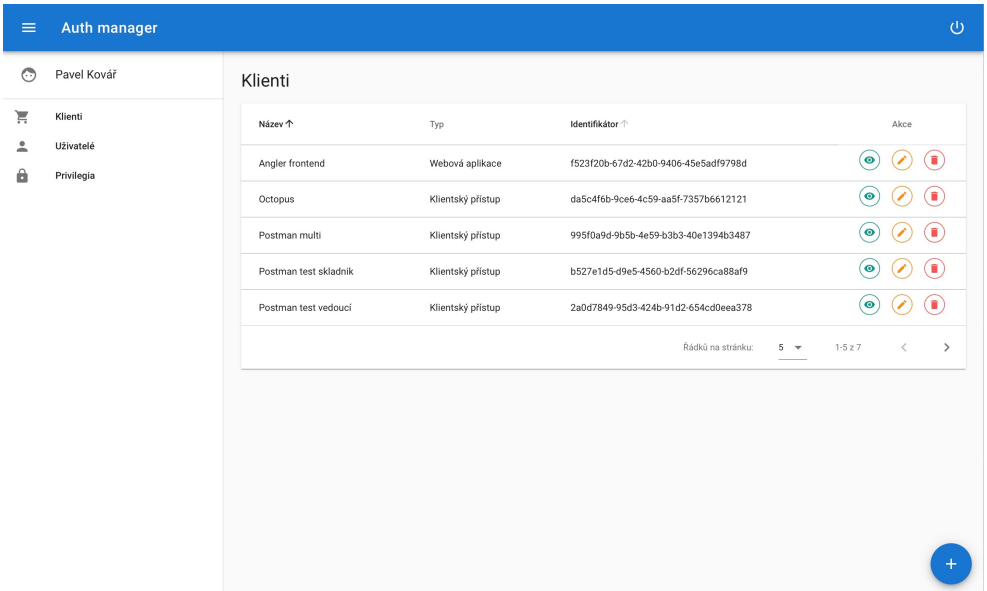
Obrázek B.1: Konceptuální model databáze skladového systému



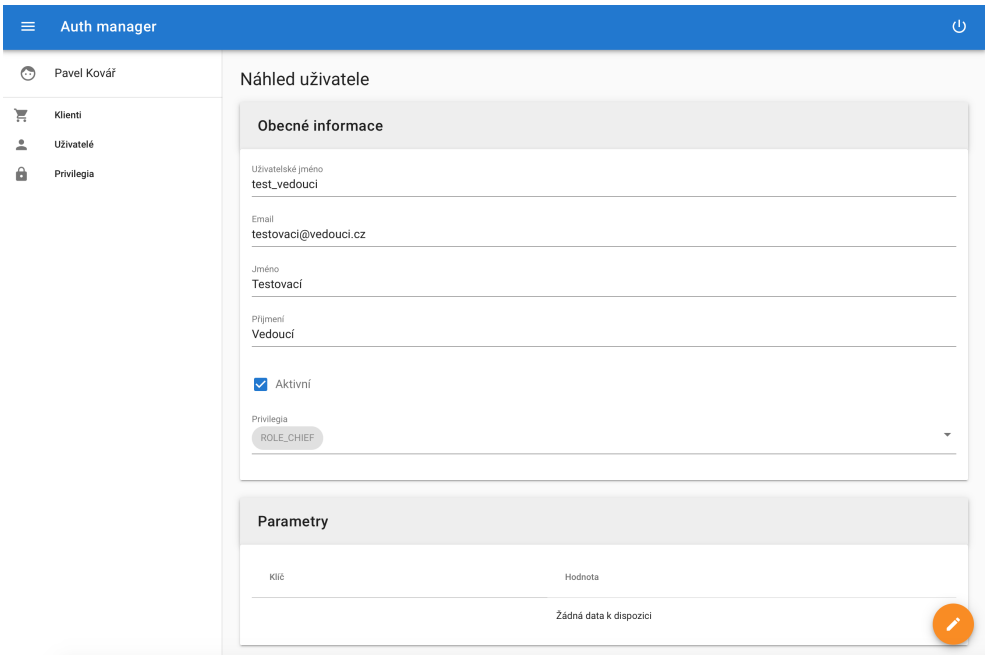
Obrázek B.2: Konceptuální model databáze autorizačního serveru

Příloha C

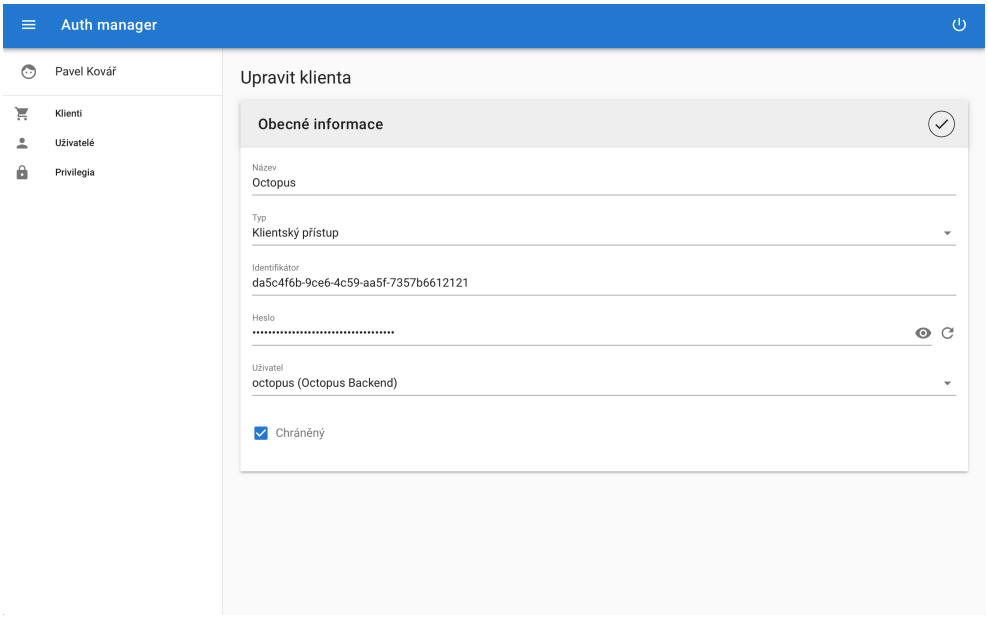
Ukázka administrace autorizačního serveru



Obrázek C.1: Autorizační server: seznam klientů



Obrázek C.2: Autorizační server: náhled uživatele



Obrázek C.3: Autorizační server: úprava klienta

Příloha **D**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Kovář_Pavel_2019.pdf` ..... text práce ve formátu PDF
- `assets` ..... přílohy
  - `angler` ..... náhled administrace autorizačního serveru
  - `database` ..... konceptuální modely databází
  - `diagram` ..... diagramy použité v této práci
  - `docs` ..... dokumentace aplikačních rozhraní