FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Evidenční systém agentury Haul
Student:	Bc. Matěj Sháněl
Vedoucí:	Ing. Tomáš Nováček
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Agentura Haul, organizátor rekreačních kurzů a majitel rekreačních středisek, by ráda automatizovala svůj systém na kontrolu naplněnosti skladu. Každoročně se potýká s problémem pomalé a neintuitivní evidence zboží, která má za následek jak časovou, tak finanční ztrátu.

- Analyzujte aktuální stav evidenčního systému skladů agentury Haul.
- Na základě komunikace s pracovníky agentury analyzujte jejich požadavky na aplikaci.
- Navrhněte architekturu a funkce aplikace tak, aby splňovaly požadavky klienta.
- Návrh musí obsahovat korektní zvolení technologií a synchronizovanou persistenci mezi jednotlivými instancemi.
- Aplikaci implementujte.
- Proveďte nasazení a uživatelské a integrační testy aplikace.
- Pro výslednou aplikaci vytvořte instalační a uživatelský manuál.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 17. října 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Evidenční systém agentury Haul

Bc. Matěj Sháněl

Katedra softwarového inženýrství
Vedoucí práce: Ing. Tomáš Nováček

25. května 2020

Poděkování

Ačkoliv se proces tvorby této práce výrazně podepsal na mém fyzickém i psychickém zdraví, podařilo se mi tuto jedinečnou a s odstupem času snad i docenitelnou událost přežít. To by se ovšem nestalo bez velkého přičinění lidí, kterým bych chtěl alespoň následujícími několika řádky vzdát dík.

První a rozhodně největší poděkování patří vedoucímu mé diplomové práce, Ing. Tomáši Nováčkovi, který mi byl oporou ode dne, kdy se objevil nápad, jež dal za vznik zadání této práce, až po okamžik, kdy jsem vytvořenou práci odevzdával. Bez jeho rad, připomínek, nápadů, ochoty věnovat se mi v kteroukoliv denní dobu a ustavičného pobízení, ať něco dělám, by tato práce nejspíš nebyla hotová ani v létě příštího roku.

Další osobou, které patří mé neskonalé díky, je Mgr. Jana Dvořáková, jež v procesu tvorby této práce představovala hlas agentury Haul. Svou ochotou nalézt si chvilku na odpověď na moje dotazy a odvahou zúčastnit se uživatelského testování dokázala, že je opravdu „holkou pro všechno v agentuře Haul“.

Poděkování také bezesporu patří mým rodičům, kteří byli ochotní poslouchat špatné programátorské metafory a dělit se o názory na problémy, s nimiž jsem se při vytváření této práce potýkal. Zároveň bych jim chtěl poděkovat za možnost si psychicky odpočinout u betonování obrubníku.

Velké poděkování patří i mé přítelkyni Viky, která mě pravidelně zachraňovala před pádem do hlubin beznaděje, ošetřovala poraněné výhonky a podávala krumpáč a mop, když bylo potřeba. Zároveň bych chtěl poděkovat i všem mým přátelům, kteří na mně nešetřili podporou a motivací.

Rád bych poděkoval také Robertu Čeňkovi a Renému Pacholíkovi, kteří mi v rámci analýzy existujících řešení poskytli informace o systémech, které jejich společnosti vyvíjejí. Nerad bych zapomněl i na všechny, kteří mi pomohli s uživatelským testováním. Díky za Vaši trpělivost a velmi podnětné názory.

Speciální díky patří také Květoslavovi za korekturu a stylizaci textu této práce a Apakrychlím, které dohlížely na vytváření této práce z pravého dolního rohu a postaraly se o zabezpečení dat na testovacím serveru.

Ještě jednou velké díky Vám všem. Bez Vás by toto dílo nikdy nevzniklo.

Hash´tar is´h is´h

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. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 25. května 2020 .....

Odkaz na tuto práci

Sháněl, Matěj. *Evidenční systém agentury Haul*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Cílem této práce je vytvoření systému pro agenturu Haul, jenž se bude zabývat evidencí zboží.

V první části práce je vypracována analýza současné podoby procesů, které zaměstnanci agentury Haul používají pro evidenci nákupů a výdejů zboží. V kontextu těchto procesů jsou analyzovány existující programy, jež se zabývají touto problematikou, a vymezeny požadavky na vytvářený systém.

Následně je vytvořen návrh systému, v němž je kladen důraz především na zvolení vyhovujících technologií a nutnost operování nad daty, jež budou sdílena mezi více uživateli. Na základě tohoto návrhu je systém implementován.

Vytvořený systém je na závěr podroben integračnímu a uživatelskému testování. V rámci testování byla provedena Nielsenova heuristická analýza vytvořeného systému.

Výsledný systém umožňuje zaměstnancům agentury Haul přehledně a jednoduše zpracovávat a analyzovat data spojená s evidencí zboží.

Klíčová slova webová aplikace, skladový systém, správa objednávek, evidence zboží, gastronomie, Nette, PHP, MySQL

Abstract

The objective of this thesis is to create a system for the Haul agency, which would deal with the registration of goods.

In the first part of the thesis, an analysis is drawn up of the current state of the processes, which employees of the Haul agency use to register purchases and issues of goods. In the context of these processes, the existing programs, which deal with this kind of issues, are analyzed and the system requirements are defined.

After that, a design of the system is created, which primarily emphasizes the choice of satisfactory technologies and the need for operating with data, that are to be shared among multiple users. Based on this design the system is implemented.

In the end, the created system is subjected to the integration and user testing. As a part of the testing the Nielsen heuristic analysis of the created system has been carried out.

The resulting system allows employees of the Haul agency to process and analyze the data related to the registration of goods in a simple and organized manner.

Keywords web application, storage system, requisition management, registration of goods, gastronomy, Nette, PHP, MySQL

Obsah

Seznam obrázků

Úvod

Úspěšné vedení rozvíjející se společnosti s sebou nese obsáhlý seznam věcí, o něž je potřeba se postarat. Mezi tyto věci se mimo jiné řadí například marketing, zajišťování prostorů pro vykonávání činnosti, investice do rozvoje společnosti a správa lidských a hmotných zdrojů. Poslední jmenovaný bod se stává zásadním, pokud zmíněná společnost potřebuje v rámci své činnosti denně nasytit stovky svých zákazníků.

V takovém případě málokdy stačí si množství surovin dostupných pro zajištění jídla na následující den pouze pamatovat nebo ho mít zaznamenané v arších papírů. Takové způsoby vedení záznamů nejsou příliš spolehlivé ani vhodné pro pozdější účtování. Zároveň příhodně ztracený arch či vynechaný záznam poskytují místo pro únik drahocenných zdrojů společnosti do soukromého vlastnictví.

Řešením, jež se přímo nabízí, by mohlo být přenechání většiny starostí dedikovanému systému. Ten by se mohl postarat o většinu komplexních úkonů, které se s vedením evidence zboží pojí.

Složitost problematiky evidence zboží se velmi často promítá i do systémů, jež si kladou za cíl s evidencí svým uživatelům pomoci. Ona složitost většinou pramení ze snahy systémů o vyhovění co nejširšímu spektru potenciálních zákazníků. To ovšem často vede k přílišné generalizaci, která přímo souvisí se snahou nutnou k pochopení a používání těchto systémů.

Každý systém je ovšem jen natolik užitečný, nakolik s ním jeho uživatelé spolupracují. Chtít po kuchaři, jež právě po třech hodinách dokončil přípravu a vydávání jídla pro čtyři sta mladých tábornic, aby strávil další hodinu evidencí použitých surovin, by nebylo moc rozumné. Unavený a případně znechucený kuchař by zadaný úkol neodvedl vůbec nebo by byl při jeho plnění kvůli únavě velmi náchylný na dělání chyb.

Ideální by tedy bylo, aby společnost disponovala evidenčním systémem, který by byl přizpůsoben jejím potřebám. Takový systém by mohl být díky svému úzkému zaměření dost jednoduchý, intuitivní a jeho obsluha by mohla

k velké radosti jeho uživatelů zabrat naprosté minimum času.

„Kde ale takový systém vzít?“ dotázal se jednoho zářijového odpoledne majitel agentury Haul vedení organizačního týmu Seznamováku FIT ČVUT. Jakmile jsem se rozhlédl po přítomných a uviděl významný pohled Ing. Tomáše Nováčka vyslaný mým směrem, bylo mi jasné, co bude předmětem mé diplomové práce a kdo mi ji povede.

Cíl práce

Cílem této práce je vytvoření systému, jenž by zaměstnancům agentury Haul usnadnil a zpřehlednil evidenci zboží.

Pro dosažení tohoto cíle bude nejprve potřeba analyzovat stávající stav procesů, jež zaměstnancům agentury Haul slouží k evidování zboží. Na základě stavu současných procesů a diskuze se zaměstnanci agentury Haul bude možné vymezit požadavky na vytvářený systém.

Vymezení požadavků položí základ pro výběr vhodných technologií, jež usnadní tvorbu kýženého systému. Dalším aspektem, který zásadně ovlivní výběr technologií, bude nutnost synchronizace persistence dat mezi jednotlivými instancemi systému.

V rámci práce bude následně navržený systém implementován v souladu se standardy softwarového inženýrství. Kromě implementace samotné se tedy práce bude zabývat také řádným otestováním vytvořeného systému integračními a uživatelskými testy.

Práce se dále zaměří také na usnadnění procesů nasazení vytvořeného systému. Výsledný systém bude doplněn o instalační a uživatelský manuál.

Kapitola 1

Analýza

Smyslem této kapitoly je ušetřit čtenáře několikatýdenního martyria pronikání do problematiky spojené se závodním stravováním, pročítání emailového ping-pongu s cílem pochopení představ zákazníka a procházení stohu papírů, jež má ruka neprodyšně pokryla poznámkami ze schůzek.

Na následujících stranách nejprve přiblížím důvody vedení agentury Haul pro potřebu zavedení evidenčního systému skladů a upřednostnění řešení vytvořeného na míru před obecně dostupnými řešeními.

Dále shrnu představy uživatelů o vytvářeném systému. Nejprve identifikuji základní případy použití, na jejichž základě definuji požadavky, jež budou kladeny na funkcionalitu a infrastrukturu budoucího systému.

Ujasnění těchto požadavků mi umožní navázat analýzou aplikací zabývajících se závodním stravováním. V rámci této analýzy identifikuji prvky, jež se pozitivně podílejí na snadném ovládání a plynulosti práce s aplikací, a přístupy, jež se neosvědčily, a jejichž použití by tedy bylo záhodno se vyhnout.

1.1 Důvody zadání práce

Agentura Haul se zabývá pořádání škol v přírodě, letních táborů, adaptačních kurzů či firemních akcí ve svých areálech napříč Českou republikou. Společně s programem zajišťuje také stravování a ubytování pro zúčastněné pomocí vlastních pracovníků.

V současné době provozuje agentura pět areálů s celkovou kapacitou přibližně šestnáct set míst, která jsou v průběhu hlavní sezóny (od dubna do září) skoro nepřetržitě zabraná [1].

Během hlavní sezóny si tudíž zaměstnanci každého z areálů musí poradit s přípravou programu, ubytování a stravy pro dvě stě až skoro čtyři sta účastníků (v závislosti na areálu). Tyto úkony s sebou nesou, jak si čtenář jistě dokáže představit, spoustu vyzývavých problémů, jež je potřeba řešit.

Jedním z těchto problémů je evidování množství uskladněných potravin. Přebytek potravin by vedl ke zbytečnému kažení, nedostatek pak k hladovění ubytovaných.

Zpočátku si zaměstnanci obchodního oddělení a kuchaři jednotlivých areálů vedli lokální záznamy formou elektronických tabulek o současném stavu naskladněných potravin. V tomto systému bylo ovšem velmi problematické zjistit aktuální stav potravin v areálech, neboť si nikdo z pracovníků nemohl být jistý, zda právě vlastní aktuální verzi tabulky.

Z tohoto důvodu začali zaměstnanci agentury Haul používat Tabulky Google [2], které odstranily problém s více verzemi. Jakmile byl problém s přehledností vyřešen, zaměřila se agentura Haul na další z problémů – efektivitu.

Při diskuzích s kuchaři areálu ohledně jejich názoru na nové řešení se ukázalo, že zadávání změn množství potravin do tabulek zabírá příliš mnoho času. Zároveň vzhledem k nejasnému formátu některých informací v tabulkách (např. jednotky u zboží) docházelo k nedorozuměním, která vedla k nesrovnalostem ve vyúčtování.

Vedení agentury Haul se tedy rozhodlo poohlédnout se po softwarovém řešení problému evidence množství potravin s nadějí, že existující řešení, jež se osvědčilo jiným, pomůže zlepšit efektivitu i v jejich společnosti. Po průzkumu trhu se rozhodli zakoupit a nasadit systém Magdalena LK [3] od společnosti Altisima.

K jejich velkému zklamání však systém Magdalena LK nepřinesl kýžené zvýšení efektivity. Nejenom kuchaři, ale i členové obchodního oddělení byli z jeho uživatelského rozhraní naprosto zmatení, a celý proces evidence se namísto zrychlení mnohonásobně zpomalil. Vedení muselo ke své velké nelibosti systém stáhnout z produkce a vrátit se k předchozímu řešení.

Navzdory promrhané investici se vedení agentury Haul rozhodlo své snahy o zvýšení efektivity nevzdat. Z negativní zkušenosti s obecnými řešeními se rozhodli nechat si vytvořit systém na míru jejich požadavkům.

Při hledání dodavatele, jenž by jim byl schopný vytvořit systém na evidenci zásob podle jejich představ, se obrátili na skupinu studentů Fakulty informačních technologií ČVUT, jež každoročně využívá jeden z jejich areálů pro organizaci seznamovacího kurzu pro nové studenty bakalářského studia.

Hlavní apel kladený ze strany agentury Haul byl po předchozích zkušenostech na snadnost používání aplikace i pro méně technicky zdatné uživatele.

1.2 Vytvářený systém

Klíčovým předpokladem pro dosažení maximální spokojenosti zaměstnanců agentury Haul s vytvářeným systémem je zjištění, co přesně od daného systému očekávají.

V následujících sekcích proto nejprve popíši způsob, jakým probíhají procesy spojené s evidencí množství uskladněných potravin. Na jejich základě

popíši typy uživatelů, kteří budou se systémem interagovat, a vymezím základní požadavky na tento systém.

Na základě těchto požadavků navrhnu případy užití, které rozdělím mezi identifikované typy uživatelů tak, aby kompletně pokryly vymezené požadavky.

Následující sekce vznikaly na základě podnětů získaných z konverzací s vedoucí obchodního oddělení agentury Haul, Mgr. Janou Dvořákovou.

1.3 Současná podoba procesů

Smyslem této části sekce je popsání způsobu, jakým v současné době probíhají procesy související s evidencí množství uskladněných potravin mezi zaměstnanci agentury Haul.

1.3.1 Nákup

Prvním z procesů je nákup potravin. Podle velikosti a frekvence se nákupy dělí do následujících tří skupin:

Velký nákup má nejnižší frekvenci (většinou v řádu týdnů), avšak co do objemu je největší. Nákup sjednávají zaměstnanci obchodního oddělení a dodavateli bývají spíše velkobchody jako například Makro. Typickou komoditou je například maso nebo přílohy.

Malý nákup má vyšší frekvenci než „velký nákup“ (řádově dvakrát do týdne), velikostí je pak spíše menšího rázu. Tento typ nákupu sjednávají zaměstnanci konkrétního areálu (povětšinou kuchaři) a dodavateli bývají obchody v blízkém okolí areálu. Typicky se jedná o ovoce a zeleninu.

Každodenní nákup má nejvyšší frekvenci (denní – jak již napovídá název), velikostí je však menší či srovnatelný s „malým nákupem“. Nákup bývá dopředu nasmlouvaný s lokálními dodavateli s tím, že se pouze mění množství dodávaného zboží. Předmětem tohoto nákupu bývá v drtivé většině případů pečivo.

Každý nákup musí zaměstnanci areálu převzít a na základě faktury zaevidovat navýšení množství potravin ve skladu.

V případě „velkého nákupu“ se občas stává, že dodavatel nedodá všechny objednané položky. V takové situaci musí zaměstnanci areálu kontaktovat zaměstnance obchodního oddělení a požádat je o dokoupení chybějících potravin.

1.3.2 Výběr ze skladu

Při vyzvedání zásob ze skladu musí zaměstnanci areálu zavést změny v množství jednotlivých potravin do systému. Zaměstnanci se snaží počet výběrů ze skladu minimalizovat, neboť každý výběr zkracuje jejich čas o režii spojenou se zadáváním změn do systému.

Naneštěstí se velmi zřídka stává, že by stačila pouze jedna návštěva skladu, při níž by zaměstnanci vyzvedli všechny potřebné potraviny. Naopak je naprosto běžné, že kuchař během přípravy jídel posílá své pomocníky do skladu pro chybějící suroviny.

Tyto dodatečné výběry se většinou dějí při přípravě jídel, při níž zaměstnanci nemají času nazbyt. Z tohoto důvodu si záznamy těchto výběrů uchovávají pouze v paměti nebo je zapisují na k tomuto účelu určený list papíru, aby je na závěr dne mohli zadat do systému všechny najednou.

Bohužel se často stává, že si zaměstnanec na konci dne nedokáže přesně vzpomenout, jaké množství potravin ze skladu vyzvedl, nebo se při úklidu ztratí jeden z listů se záznamy vyzvednutých potravin. To následně způsobuje nepřesnost a chybnost záznamů.

1.3.3 Kontrola stavu zásob

Zaměstnanci areálu si potřebují udržovat přehled o množství jednotlivých potravin, aby mohli v případě, že nějaké začnou docházet, včas objednat další. Tato činnost je důležitá především u potravin, které spadají pod „velké nákupy“, neboť ty musejí být zprostředkovány obchodním oddělením.

V současné době se tyto kontroly dějí při vybírání potravin ze skladu. Jak bylo zmíněno v popisu procesu „Výběr ze skladu“, tyto výběry jsou velmi často prováděny v časové tísni, a není tedy neobvyklé, že se na danou skutečnost ve spěchu zapomene.

Vzhledem k nejasnému množství potravin proto zaměstnanci před zadáváním nákupů pro jistotu ještě jednou procházejí sklad, aby zkontrolovali, zda nedocházejí potraviny, které zatím neměli v plánu nakoupit.

V horší situaci se nachází zaměstnanci obchodního oddělení, neboť pokud potřebují zjistit stav zásob oni, musí kontaktovat zaměstnance konkrétního areálu, kteří následně provádějí manuální kontrolu.

Na tomto místě by uvědomělý čtenář mohl namítnout, že stav zásob se dá alespoň částečně odhadnout z informací vedených v systému. To je sice pravda, ovšem kvůli důvodům zmíněným v předchozích odstavcích si obchodní oddělení většinou raději vyžádá novou detailní kontrolu.

1.3.4 Převoz zboží mezi areály

Během sezóny také běžně vyvstává nutnost převozu některých potravin mezi areály.

Tato nutnost bývá zapříčiněna nedostatečným pochopením požadavků ubytovávaných či chybnými záznamy o množství potravin podléhajících rychlé zkáze, jež by kvůli nadbytečnému množství nebyly včas spotřebovány.

Skutečnost bývá reportována zaměstnanci areálu, v němž je problém s nedostatkem či přebytkem potravin, zaměstnancům obchodního oddělení. Obchodní oddělení se následně pokusí najít způsob, jakým pomocí přesunu potravin z či do jiného blízkého areálu pro ně situaci vyřešit.

V případě nalezení takového řešení uvědomí zaměstnanci obchodního oddělení zaměstnance areálů o detailech převozu, jež se mezi jejich areály uskuteční.

Zaměstnanci obchodního oddělení zároveň zaznamenají přesun potravin do systému, neboť se nepředpokládá, že by se převoz nezdařil. I přesto si většinou nechávají příjem potravin v cílovém areálu potvrdit tamějšími zaměstnanci.

1.3.5 Komunikace mezi zaměstnanci

Pro hladkou koordinaci nákupů je potřeba pravidelné komunikace mezi zaměstnanci obchodního oddělení na straně jedné a zaměstnanci jednotlivých areálů na straně druhé.

V rámci této komunikace je domlouván především obsah „velkých nákupů“. Kromě toho tyto dvě skupiny zaměstnanců řeší také případné převozy potravin mezi areály nebo jiné organizační informace.

Zaměstnanci pro tuto komunikaci v současné době využívají email.

1.3.6 Analýza dat

Obchodní oddělení si potřebuje kromě koordinace „velkých nákupů“ a převozu zboží udržovat přehled o dění napříč areály.

Základní informací, jež musí být zaměstnancům obchodního oddělení dostupná, je součet výdajů za nákup potravin v rámci každého z areálů.

Dále obchodní oddělení analyzuje vývoj výdajů za jednotlivá období v sezóně a porovnává výdaje v závislosti na počtu ubytovaných v rámci každého areálu a mezi areály v průběhu času.

Na základě těchto informací jsou regulovány požadavky na nákupy pro areály a upravuje se cenová politika agentury.

V současné době musí tyto informace získávat zaměstnanci obchodního oddělení manuálním prohledáváním záznamů a faktur z nákupů. Tento proces zabírá značné množství času a je kvůli své komplexitě velmi náchylný na chybu zapříčiněnou nepozorností.

1.4 Uživatelé

Napříč definovanými procesy figurují následující dva typy uživatelů:

Zaměstnanci jednotlivých areálů jsou především kuchaři a jejich pomocníci. Procesy, v nichž tento typ uživatelů figuruje, pokrývají širokou škálu činností, avšak jsou vždy úzce spjaty pouze s jedním konkrétním areálem.

Zaměstnanci obchodního oddělení mají za úkol koordinovat zásobování všech spravovaných areálů agentury. K tomu potřebují mít přehled o celkovém pohybu potravin či vývoji výdajů v průběhu času.

Kromě těchto dvou základních typů uživatelů bude potřeba uživatelů, kteří budou moci systém za běhu spravovat. Tito uživatelé budou mít speciální práva, která jim umožní například přidávat další uživatele či mazat chybně vytvořené typy potravin.

1.5 Požadavky

Díky analýze současné podoby procesů jsme mohli společně s Mgr. Janou Dvořákovou vymezit požadavky, jež jsou kladeny na vytvářený systém. Tyto požadavky jsou podrobněji rozepsány v následujících sekcích.

Vzhledem k obsáhlosti vytvářeného systému jsou požadavky rozděleny na klíčové („must-have“) a doplňkové („nice-to-have“).

Klíčové požadavky zajišťují základní funkcionalitu systému a v případě jejich nesplnění nebude možné systém nasadit do provozu. Z tohoto důvodu musí být všechny splněny již v první verzi vyvíjeného systému.

Oproti tomu požadavky doplňkové pouze usnadňují a zrychlují práci se systémem a přinášejí více pohodlí pro uživatele. Jejich nenaplnění ovšem neohrozí možnost nasazení systému. Kvůli tomu nejsou vyžadovány v první verzi.

Všechny aspekty požadavků, jež rozeberu na dalších řádcích, jsou implicitně brány jako klíčové. V případě, že některé části požadavku budou pouze doplňkové, zmíním to u těchto částí explicitně.

1.5.1 Funkční požadavky

V této sekci jsou podrobně rozepsány požadavky, jež přímo vymezují funkcionalitu vytvářeného systému.

F1 – Nákupy

Systém bude umožňovat svým uživatelům práci s nákupy. Tímto se rozumí vytváření záznamu o nákupu a následné přidání nakupovaných položek.

Podle svého typu budou uživatelé moci pracovat s nákupy následovně:

- Zaměstnanci obchodního oddělení budou moci pracovat pouze s „velkými nákupy“.

- Zaměstnanci jednotlivých areálů budou mít zpřístupněny „malé“ a „každodenní nákupy“.

V rámci usnadnění práce budou uživatelé moci také ukládat rozpracované verze nákupů pro pozdější dokončení. Zároveň budou mít možnost rozdělit nákup, na kterém právě pracují. Tyto možnosti jsou doplňkovými požadavky.

F2 – Požadavky nákupu

Zaměstnanci jednotlivých areálů budou mít možnost zaslat v rámci systému seznam požadavků na velký nákup zaměstnancům obchodního oddělení.

Tyto návrhy budou zpracovávat zaměstnanci obchodního oddělení. V rámci zpracování požadavku ho budou moci upravit podobně, jako by tento nákup vytvářeli sami.

Zároveň bude systém poskytovat možnost uložit rozpracovanou verzi seznamu požadavků pro pozdější dokončení. Tato možnost se řadí k doplňkovým požadavkům.

F3 – Přijem nákupu

Systém bude umožňovat zaměstnancům jednotlivých areálů zaznamenávat přijetí nákupu. Přijmout bude možné všechny typy nákupu.

V rámci přijímání nákupu bude systém poskytovat možnost upravit přijímaný nákup. Tímto se rozumí odebrání položek nebo změna kvantity v rámci položky.

F4 – Recepty

Zaměstnanci jednotlivých areálů budou mít možnost sestavit a upravit v systému recept pro snazší budoucí zadávání výběru potravin ze skladu.

Recept se bude skládat z položek, z nichž každá bude mít typ potraviny a množství potřebné pro vytvoření určitého počtu porcí. Tento počet budou uživatelé nastavovat při vytváření receptu.

Recepty, jež uživatel vytvoří, nebudou primárně sdíleny s ostatními uživateli.

Systém bude umožňovat vytvoření tisknutelné verze receptu. Tato verze bude obsahovat tabulku potřebných potravin s množstvím odpovídajícím předem zadanému počtu porcí. Vytváření tisknutelné verze receptů patří mezi doplňkové požadavky.

F5 – Šablony receptů

Systém bude poskytovat zaměstnancům jednotlivých areálů možnost vytvořit ze svého receptu šablonu. Šablony receptů budou sdílené se všemi zaměstnanci areálů.

Správu šablon receptů budou zajišťovat zaměstnanci obchodního oddělení. Správou je myšleno schvalování vytvořených šablon, jejich úprava a mazání.

Práce se šablonami je kategorizovaná jako doplňkový požadavek.

F6 – Výběr potravin ze skladu

Zaměstnanci jednotlivých areálů budou mít možnost vybrat potraviny ze skladu. Systém bude umožňovat zadat výběr manuálně či na základě receptu.

V případě manuálního výběru zvolí uživatelé typ potravin ze současného obsahu skladu a nastaví vybírané množství ručně.

Při výběru potravin na základě receptu systém typy potravin a jejich množství vyplní a uživatelé budou mít možnost tyto údaje upravit. Úpravou je myšleno odebrání nebo přidání položek a změna množství potravin v rámci položek.

F7 – Převoz potravin

Zaměstnanci obchodního oddělení budou mít možnost zaevidovat do systému převoz potravin mezi areály.

Systém bude vyžadovat potvrzení naložení od zaměstnanců areálu, z něhož budou potraviny převáženy, a potvrzení přijetí od zaměstnanců areálu, do něhož potraviny míří.

F8 – Zobrazování přehledů

Systém bude umožňovat zaměstnancům obchodního oddělení zobrazovat přehledy výdajů napříč areály.

Zaměstnanci si budou moci nechat zobrazit přehled celkových výdajů během zvoleného časového období daného areálu. Dále budou mít možnost tyto výdaje porovnat mezi areály či je nechat přepočítat na počet strávníků, jež se v daném období v areálu stravoval.

Všechna tato porovnání budou brát v potaz pouze celkové náklady. Porovnávání na základě složitějších kritérií je doplňkový požadavek.

Dalším z doplňkových požadavků je možnost ukládání přehledů resp. způsobu jejich nastavení.

F9 – Přehled o stavu potravin

Všichni uživatelé systému budou mít možnost nahlížet na stav potravin v areálech. Zaměstnanci jednotlivých areálů však budou mít náhled omezený pouze na areál, v němž pracují.

Systém bude zároveň umožňovat zaměstnancům jednotlivých areálů nastavit pro každý typ potravin hranici nutného minimálního množství. Tato hranice má platnost v rámci areálu působnosti daného zaměstnance.

Jakmile stav dané potraviny klesne pod nastavenou úroveň, bude systém zobrazovat upozornění, že daná potravina dochází, všem zaměstnancům z daného areálu.

F10 – Počet strávníků

Systém bude umožňovat zaměstnancům jednotlivých areálů nastavovat počet aktuálně ubytovaných strávníků. Na základě tohoto nastavení bude systém vyplňovat množství potravin v receptech.

Uživatelé budou zadávat počet osob a dobu, po jakou bude toto množství platné. Po uběhnutí této doby požádá systém uživatele o zadání nových hodnot. Pokud uživatel hodnoty nezadá, bude systém zobrazovat upozornění ohledně neplatných hodnot do doby, dokud nebudou zadány nové hodnoty.

F11 – Správa systému

Uživatelé se speciálními právy budou mít přístup ke správě systému. Tento požadavek umožňuje především možnost řídit systém za běhu bez nutnosti vytváření jeho nových verzí.

V rámci správy budou moci uživatelé se speciálními právy přidávat a odebírat uživatele s výjimkou sebe sama navzájem. Zároveň budou moci změnit typ již existujícího uživatele.

Jakýkoliv zásah správce do účtu jiného uživatele nahlásí tomuto uživateli systém zasláním informačního emailu na emailovou adresu propojenou s jeho účtem.

Uživatelé se speciálními právy budou mít možnost přidávat, upravovat a mazat areály. Jakýkoliv zásah do těchto hodnot bude vyžadovat opětovné potvrzení ze strany uživatele, který změnu provádí.

Zároveň budou mít tito uživatelé možnost nechat si zobrazit a případně smazat hodnoty v systému. Těmito hodnotami jsou myšleny potraviny, jejich typy a dodavatelé. Tato možnost má primárně sloužit především k odstranění hodnot, jež byly do systému zavedeny omylem či nekorektním používáním.

Doplňkovým požadavkem v rámci správy je možnost zobrazování historie provedených akcí v rámci systému. Tyto akce by mělo být možné filtrovat podle jejich atributů.

F12 – Správa účtu

Každý uživatel bude mít možnost spravovat svůj účet v rámci systému. Touto správou je myšlena úprava kontaktní emailové adresy a změna hesla.

Zároveň budou mít uživatelé možnost obnovit si heslo. Proces bude vyžadovat autentizaci pomocí kontaktní emailové adresy.

F13 – Oblíbené položky

Zaměstnanci jednotlivých areálů budou mít možnost si své recepty jako „oblíbené“ či toto označení zrušit. Zaměstnanci obchodního oddělení budou moci takto označit své přehledy.

Všechny takto označené položky se budou zobrazovat před ostatními hodnotami při vybírání. Tento požadavek patří mezi doplňkové.

F14 – Komunikace

Systém bude umožňovat posílání textových zpráv mezi uživateli. Bude se jednat o prostý textový formát, nebudou proto podporovány emotikony, obrázky ani přenos souborů. Systém bude zobrazovat uživateli upozornění, pokud bude mít nové zprávy.

Podpora komunikace v rámci systému je doplňkový požadavek.

1.5.2 Nefunkční požadavky

Tato sekce je věnována požadavkům na systém, které se spíše nežli funkcionality týkají technických aspektů, jimiž musí systém disponovat.

N1 – Bezpečnost

Vzhledem k citlivé povaze ukládaných dat musí systém umožňovat přístup pouze autorizovaným uživatelům.

Agentura Haul v současné době nepoužívá žádný normalizovaný způsob autorizace svých zaměstnanců. Z tohoto důvodu nebude potřeba při návrhu způsobu autorizace brát v potaz omezení spojená s napojováním na již existující systém.

N2 – Dostupnost

Vyvíjený systém musí být dostupný z počítačů, které používají zaměstnanci agentury Haul. U těchto strojů může být počítáno s následující specifikací:

- Operační paměť: 2 GB
- Operační systém: Windows 10
- Přístup k internetu

Zbytek specifikací může být upřesněn po dohodě s vedením agentury Haul. Vzhledem k podstatě systému by však neměl mít žádné speciální nároky na grafickou kartu, velikost úložiště či počet jader.

Systém bude primárně určený pro stolní počítače a notebooky, při návrhu by však měla být zohledněna také kompatibilita s mobilními zařízeními a tablety.

N3 – Použité technologie

Vyvíjený systém by měl omezit na nejnutnější minimum nutnost využití technologií, jež pro svůj provoz vyžadují placené licence.

Použití jakékoliv technologie vyžadující placenou licenci musí být nejprve schváleno vedením agentury Haul.

1.6 Existující řešení

Problémy, s nimiž se potýká agentura Haul, nejsou v tomto odvětví nijak neobvyklé. V současné době existuje mnoho systémů, jež se snaží vypomoci podobným organizacím s evidencí zásob a nákladů spojených s provozem stravovacích zařízení.

V rámci své práce jsem se rozhodl některé z těchto systémů analyzovat, neb by mi výsledky analýzy mohly pomoci vyhnout se chybám, jichž se dopustili druzí přede mnou, a neopomenout aspekty, které jsou pro podobné systémy klíčové.

Zároveň je možné, že naleznu systém, jež bude plně vyhovovat vydefinovaným požadavkům agentury Haul. V takovém případě by bylo nesmyslné ztrácet čas vývojem nového systému a stačilo by pouze doporučit vedení agentury Haul systém existující.

U každého z analyzovaných systému zmíním, z jakého důvodu je pro potřeby analýzy relevantní, a vymezím jeho přednosti a nedostatky v kontextu požadavků agentury Haul.

V závěru této sekce zhodnotím získané poznatky.

1.6.1 Magdalena LK

Systém Magdalena LK je skladový program, který je podle webových stránek společnosti Altisima vhodný zejména pro menší a střední provozy společného stravování. Zároveň se má vyznačovat vysokou stabilitou a intuitivním ovládáním [3].

Tento systém jsem se rozhodl do analýzy zařadit, neboť se jedná o systém, jež agentura Haul zvolila jako první dedikované řešení. Hlavním cílem analýzy Magdaleny LK tedy bude přijít na to, proč tento systém zaměstnancům agentury Haul nevyhovoval.

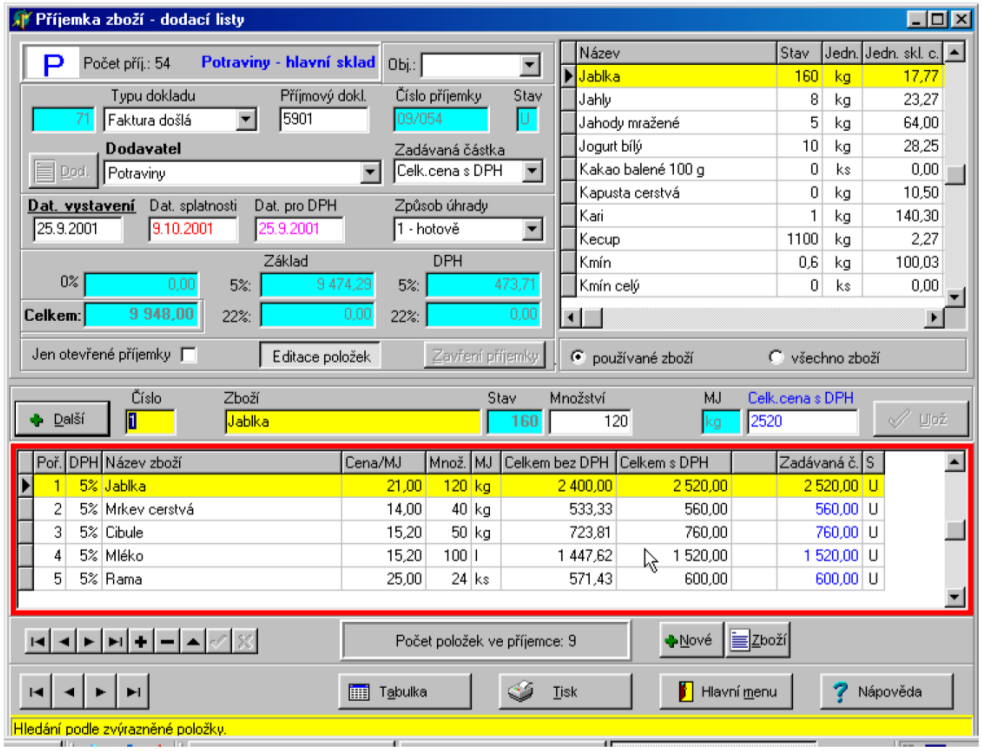
Původně jsem doufal, že mi vedení agentury Haul umožní přístup k systému. Bohužel se stažením z provozu smazala agentura tento systém ze všech svých strojů.

O funkční verzi jsem požádal i společnost Altisima, která mi místo Magdaleny LK nabídla na vyzkoušení jejího nástupce GEKON. Z tohoto důvodu jsem si při analýze Magdaleny LK musel vystačit s její uživatelskou dokumentací [4].

Systém Magdalena LK byl hlavním produktem společnosti Altisima mezi lety 2000 a 2005. Jedná se o desktopovou aplikaci, jejíž jednotlivé instance sdílejí data pomocí společné databáze na serveru společnosti Altisima.

Již při prvním pohledu do stránek dokumentace je patrné, že Magdalena LK nepatří mezi systémy vyvinuté v posledním desetiletí. Charakteristický vzhled Windows 95 či instrukce ze sekce *4.1 Zálohování dat* dokumentace Magdaleny, která vyzývá uživatele k pravidelnému ukládání záloh na diskety, možná vyvolají nostalgické vzpomínky, kvalitě rozhraní však podle mého názoru příliš nepřidávají.

Zásadní důvod pro upuštění od používání tohoto systému však vidím v jeho komplikovaném rozhraní (pro představu vizte obrázek 1.1). Více než dvacet tlačítek a polí spolu se dvěma rozměrnějšími tabulkami a pestrobarevným označením na jedné obrazovce s velkou pravděpodobností většinu běžných uživatelů spíše odradí.



Obrázek 1.1: Obrazovka dokladu z dokumentace systému Magdalena LK

Druhý problém vidím v zaměření systému na dotované stravování, které je hlídáno legislativou. Kvůli tomuto zaměření nutí systém uživatele přesně dodržovat dané normy (včasné uzavírání příjemek a výdejek, kvóty na stravování, ...).

Toto ovšem není v případě zaměstnanců agentury Haul žádoucí, neboť jejich způsob stravování není dotován a daným normám tedy nepodléhá. Místo toho by museli zaměstnanci do systému zadávat pro ně irelevantní data.

Magdalena LK ovšem nemá pouze stinné stránky. Většina uživatelského rozhraní umožňuje efektivní ovládání pomocí klávesových zkratek. Zároveň umožňuje propojení mezi záznamy o potravinách s recepty, což vede ke snazšímu plánování struktury jídelníčku.

Z rozhodnutí vedení agentury Haul je ovšem patrné, že hlavní očekávanou kvalitou vytvářeného systému musí být snadné a intuitivní ovládání. V případě podcenění tohoto aspektu se může snadno stát, že vyvíjený systém skončí stejně jako Magdalena LK.

1.6.2 GEKON

Druhým analyzovaným systémem je GEKON [5], nástupce systému Magdalena LK od společnosti Altisima.

Tento systém jsem se rozhodl analyzovat, neboť mi byl zástupci společnosti Altisima nabídnut k vyzkoušení coby jejich preferovaný gastro-ekonomický systém. Zároveň by se GEKON mohl ukázat jako ideální řešení pro agenturu Haul za předpokladu, že se vývojářům společnosti Altisima při jeho vývoji podařilo odstranit nedostatky, jimiž trpěl systém Magdalena LK.

Systém GEKON byl uveden na trh v roce 2016 a je v současné době hlavním produktem společnosti Altisima. Jádrem systému je webová aplikace, k níž uživatelé mohou přistupovat přímo přes internetový prohlížeč nebo dedikovanou desktopovou aplikaci.

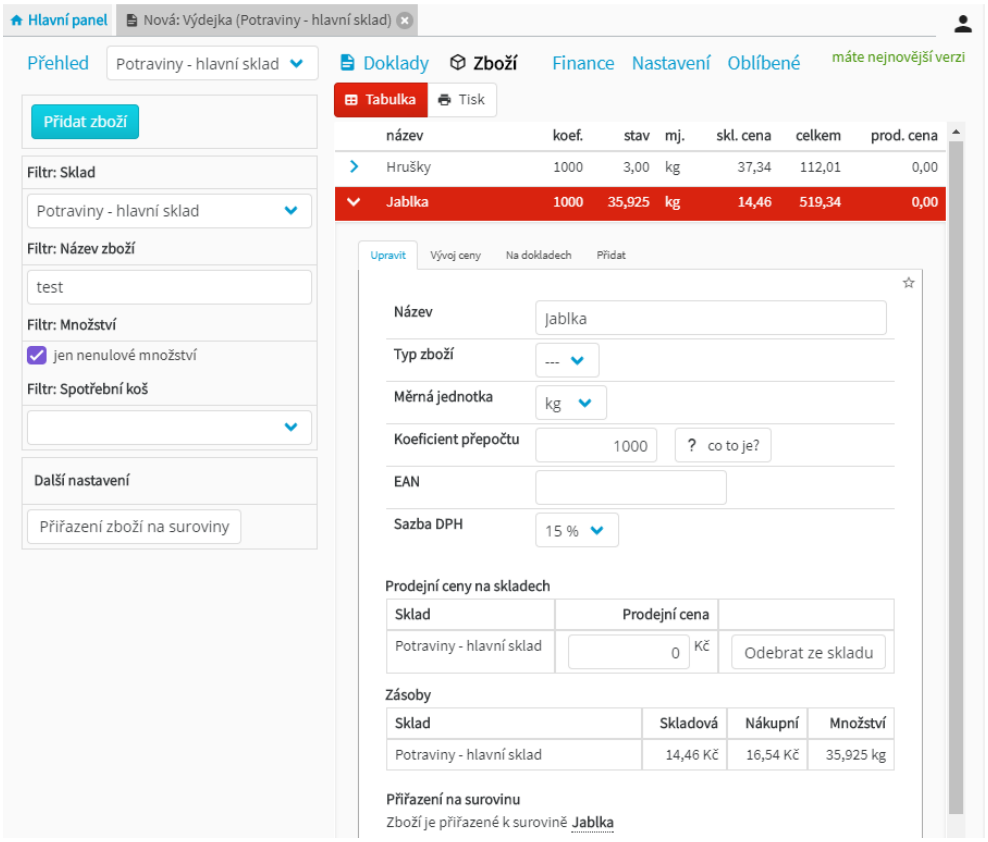
Úroveň uživatelského rozhraní systému GEKON je v porovnání s jeho předchůdcem Magdalena LK výrazně lepší (pro představu vizte obrázek 1.2). Tvůrci použili vzhled kontrolních prvků, který je typický pro moderní aplikace, a obstojně se vypořádali s dřívější přeplněností obrazovek.

Zároveň se jim podařilo zachovat drtivou většinu kladných aspektů, jimiž disponoval systém Magdalena LK. K možnosti ovládání pomocí klávesových zkratek a vytváření receptů přidali tvůrci možnost propojit konkrétní potraviny se surovinami.

Toto propojení potraviny se surovinou (neboli „typem potraviny“) umožňuje uživatelům používat při vytváření receptů obecnější popis ingrediencí. Díky tomu je následně možné při vaření podle receptu použít jakékoliv potraviny, které mají přiřazený odpovídající typ uvedený v receptu.

Stejně jako Magdalena LK i GEKON zůstává zaměřený na stravovací instituce, které kvůli dotování podléhají legislativním regulacím. Z tohoto důvodu zde stále můžeme najít mnoho případů, v nichž by zaměstnanci agentury Haul byli nuceni vyplňovat irelevantní informace.

Vzhledem k tomu, že se jedná o relativně nový systém, jsem nebyl příliš překvapený, když se mi během zběžného průchodu podařilo najít několik menších chyb (nefunkční filtrování tabulek, zástupné texty, apod.). Tyto chyby



Obrázek 1.2: Obrazovka detailu zboží v systému GEKON

nemají příliš velký dopad na funkcionalitu systému, avšak celkový dojem příliš nezlepšily.

Podstatně výraznějším nedostatkem systému GEKON je však velmi malá podpora některých procesů, které by potřebovali používat zaměstnanci agentury Haul. Mezi hlavní nedostatky v tomto ohledu bych zařadil chybějící podporu vytváření požadavků nákupu, převozu zboží mezi areály a celkově využití více typů zaměstnanců.

Systém GEKON celkově působí, jako by byl primárně určen zaměstnancům obchodního oddělení. Tento pocit je ovšem možná zapříčiněn specifickým nastavením zkušební verze, na základě které jsem systém analyzoval.

Ačkoliv by kvůli nalezeným nedostatkům systém GEKON zaměstnancům agentury Haul s největší pravděpodobností nevyhovoval, jeho přednosti mi mohou posloužit jako inspirace při návrhu vyvíjeného systému.

1.6.3 MSklad

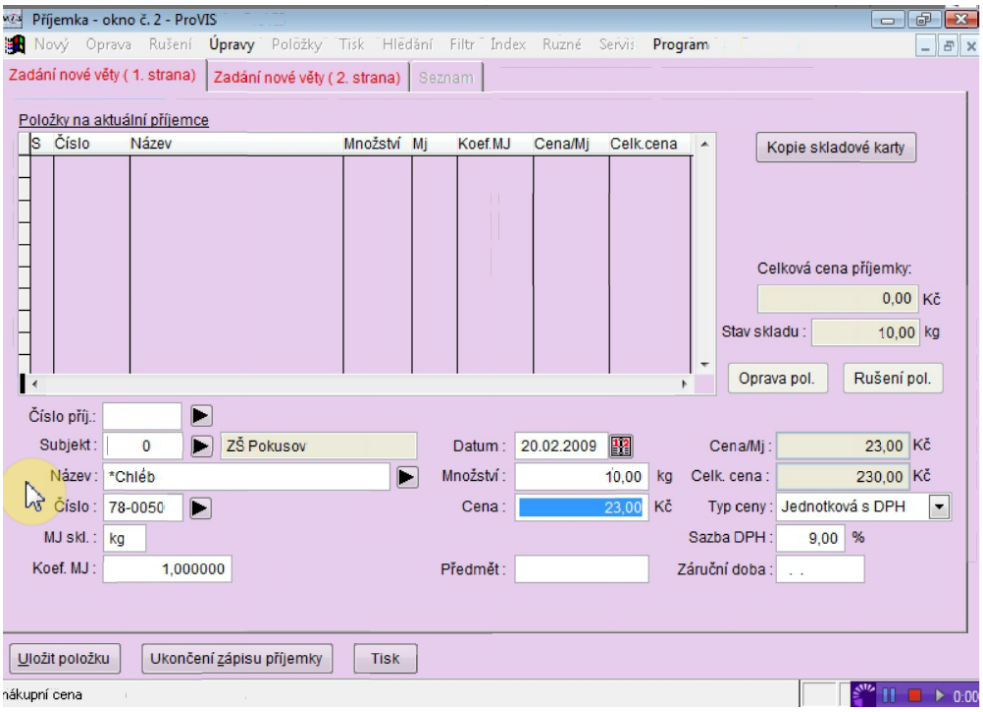
Dalším systémem, jenž jsem se rozhodl analyzovat, je MSklad [6] od plzeňské společnosti VIS.

Systém MSklad představuje v současné době jeden z nejpoužívanějších programů pro správu skladů s gastronomickým zaměřením v České republice. Tento fakt mu zajistil místo mezi systémy, jež by mohly potenciálně vyhovět požadavkům agentury Haul, a tedy i v mojí analýze.

Na stránkách společnosti VIS jsem bohužel nenašel možnost vyzkoušet si zkušební verzi tohoto systému, takže jsem svoji analýzu založil na informacích a instruktážních videích z jejich stránek [6].

Autoři systému MSklad se nechali při jeho vytváření inspirovat myšlenkou modulů. Z tohoto důvodu je základ systému relativně strohý. Z modulů s pokročilými funkcemi si pak mohou zákazníci vybrat pouze ty, které potřebují.

Základem systému MSklad je desktopová aplikace s lokální databází. Podpora více uživatelů je poskytnuta v *Síťovém* modulu. Z důvodu modulární architektury se však bude nejspíše jednat pouze o synchronizaci dat mezi instancemi na základě principu *Peer-to-Peer*.



Obrázek 1.3: Obrazovka vytváření příjemky v systému MSklad

Podobně jako předchozí analyzované systémy i MSklad řeší legislativu spojenou se školním stravováním a s ní spojená omezení. Tato skutečnost bohužel

nevyznívá příznivě v kontextu možnosti přímého využití pro potřeby agentury Haul.

Uživatelské rozhraní (pro ukázku vizte obrázek 1.3) nevypadá příliš dobře. Podobně jako u předchozích systémů se ovšem na obrazovkách nachází nemalé množství prvků, které nejsou upravitelné, či s daným úkolem přímo nesouvisí.

Systém MSklad také v základní verzi nepodporuje nastavení různých rolí uživatelů. Tento nedostatek je napraven v dodatečném modulu *Přístupová práva*, který nabízí možnost omezení přístupu jednotlivých uživatelů k různým částem systému.

Kromě těchto nedostatků systém většinu ostatních požadavků agentury Haul pokrývá. Zároveň poskytuje možnost kontroly trvanlivosti naskladněných položek. Tento koncept by mohl představovat jedno z budoucích rozšíření vyvíjeného systému.

1.6.4 BonAp Sklad

Druhou společností, jež dominuje na české scéně gastronomických systémů, je Z-Ware. Z úzce propojené trojice programů z jejich dílny se do mé analýzy nejvíce hodí BonAp Sklad [7].

Systém BonAp tuto analýzu doplní nejen coby další charakteristický zástupce gastronomických systémů, ale zároveň jako zajímavá alternativa k systému MSklad.

Jedná se o „aplikaci typu *Client-Server* na bázi SQL“ [7]. Systém je primárně určen pro používání v rámci zmíněné trojice programů, avšak při konfiguraci některých dodatečných údajů se dá používat i samostatně.

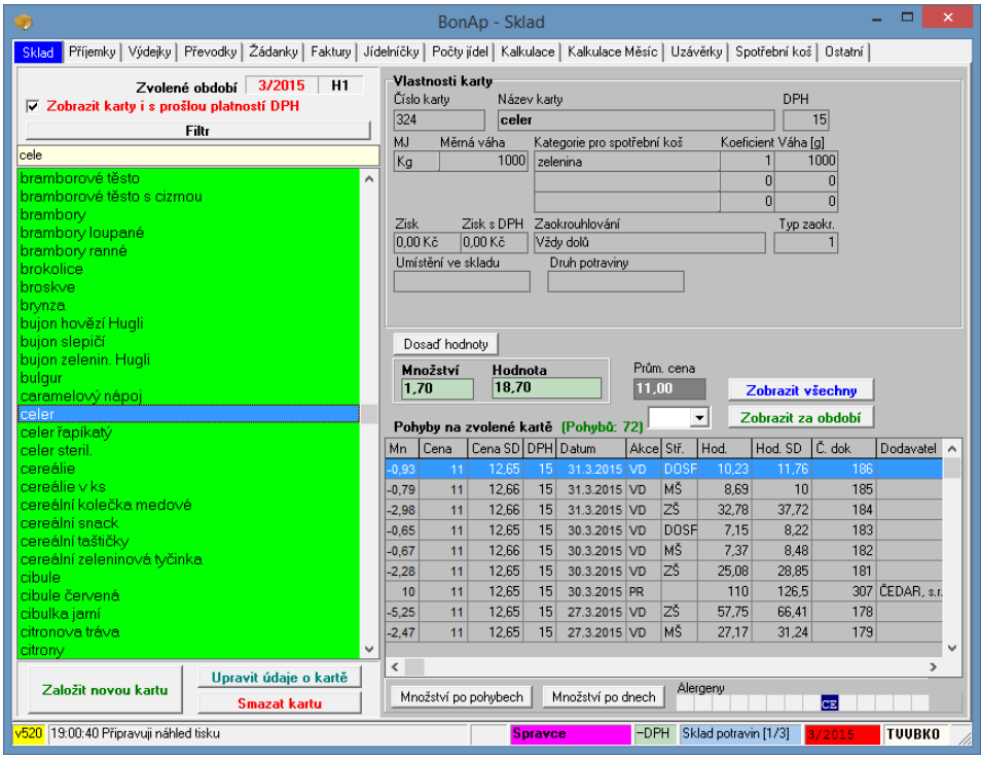
Společnost Z-Ware nabízí na svých stránkách zkušební verze některých jejich produktů, BonAp Sklad však není jedním z nich. Analýzu jsem byl tedy nucen založit pouze na několika obrázcích a obecném popisu aplikace.

Ani tento systém není výjimkou, co se týče dodržování legislativních omezení, čímž se vzdaluje představám vedení agentury Haul.

Kromě toho poskytuje svým uživatelům širokou paletu dalších nadstandardních funkcí od inventarizace přes pokročilé nastavování uživatelských práv až po integraci skladových karet. Bohužel by tyto funkce (vzhledem k vymezeným požadavkům) zaměstnance agentury Haul při používání systému spíše mátly.

Uživatelské rozhraní sice působí lehce zastarale (pro představu vizte obrázek 1.4), avšak jinak je relativně rozumně navržené. Stáří systému se mi nepodařilo zjistit, avšak kompatibilita s Windows 2000 je v tomto ohledu více než výmluvná.

BonAp Sklad také nabízí pokročilou práci s jídelníčky a seznamy alergenů. Vzhledem k charakteru stravování v areálech agentury Haul si ovšem myslím, že by tyto funkce zůstaly spíše nevyužité. Každopádně představují možné směry, jimiž se může vyvíjený systém v dalších verzích vydat.



Obrázek 1.4: Obrazovka přehledu skladu v systému BonAp Sklad

1.6.5 Sysel

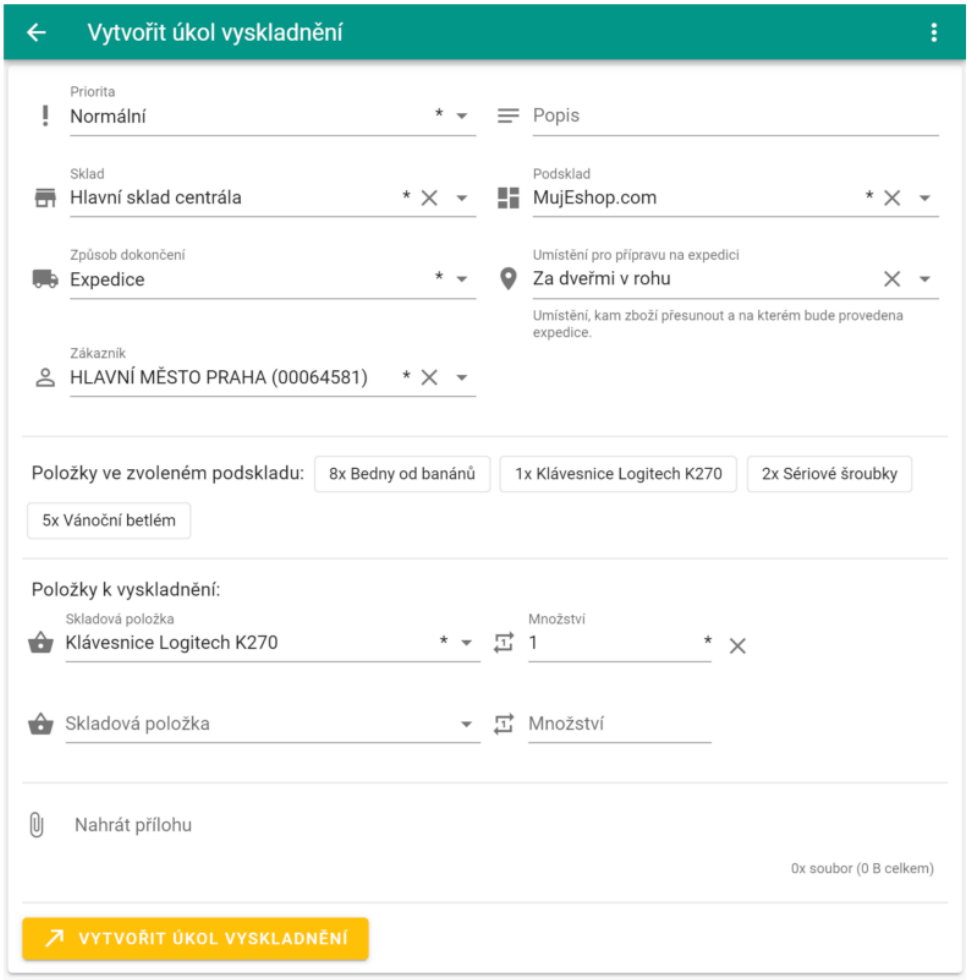
Posledním z analyzovaných systémů je skladový systém Sysel, který byl vyvinut v rámci diplomové práce Ing. Pavla Kováře [8] (backend) a Ing. Oldřicha Malce [9] (frontend).

Skladový systém Sysel jsem se rozhodl zařadit do analýzy coby zástupce čistě skladových systémů. Zároveň se jedná o výtvor mých spolužáků z Fakulty informačních technologií při ČVUT, takže se nemusím bát, že by nepřispěl žádnými podněty k mé práci.

Absence zaměření na stravování bude podle mého názoru vylučovat možnost přímého použití pro účely agentury Haul. Přesto věřím, že během analýzy narazím na aspekty, které v rámci své práce budu moci využít.

Tento systém vznikl jako rozšíření existujícího řešení od společnosti Jagu s.r.o. [10]. Sysel je webová aplikace, jejíž backend je implementován v jazyce PHP a frontend využívá Javascriptový framework *Vue.js* [11] s grafickou knihovnou *Vuetify* [12].

Systém Sysel zcela naplnil a místy předčil má očekávání. Kromě funkcí, které by každý od skladového systému očekával (jako naskladnění a výdej



Obrázek 1.5: Obrazovka vytváření úkolu vyskladnění v systému Sysel

zboží) nabízí systém Sysel také možnost práce s čárovými kódy a štítky, exportování a importování dat a evidenci zákazníků (včetně správy objednávek a jejich automatického uspokojování). Systém také nabízí podporu správy úloh zaměstnanců a inventury skladu.

Uživatelské rozhraní vypadá na první pohled přívětivě a promyšleně. Jediným nedostatkem, který bych mu mohl vytknout, je velká koncentrace prvků rozhraní na některých stránkách (pro příklad vizte obrázek 1.5). Tohoto nedostatku si však byl vědom i autor (Ing. Oldřich Malec), který ve své práci [9] (sekce 2.3.1.) uvedl, že jeho náprava bude předmětem budoucích úprav.

V celkovém souhrnu bych tedy řekl, že se jedná o velmi komplexní a povedeně zpracovaný systém. Vzhledem k jeho komplexitě by však bylo využití zajímavějších konceptů tohoto systému (např. práce s čárovými kódy či po-

kročilá konfigurace autorizačního serveru) v rámci mojí práce jako „jít s kanónem na vrabce“.

1.6.6 Zhodnocení

Všechny analyzované systémy zabývající se gastronomickým stravováním kladou velký důraz na dodržování legislativních omezení, čímž výrazně komplikují svá již tak mnohdy příliš komplikovaná uživatelská rozhraní.

Zároveň málokdy zcela pokrývají procesy, které zaměstnanci agentury Haul potřebují využívat, a nabízejí místo nich jiné, které by naopak nejspíše zůstaly nevyužity.

Systém Sysel by splňoval kritéria pro skladovou činnost, avšak postrádá klíčové prvky spojené s gastronomickým zaměřením. Jeho přizpůsobení by stálo přílišnou námahu a s největší pravděpodobností by se projevilo na následné rozšiřitelnosti systému.

Z analyzovaných systémů požadavkům agentury Haul tedy nevyhovuje žádný. Z tohoto důvodu jsem se rozhodl systém vyvinout. Tento přístup mi umožní zcela pokrýt požadavky agentury Haul a nekomplikovat systém prvky, které by nikdo nevyužil.

1.7 Případy užití

V této sekci podrobněji rozeberu případy užití vytvářeného systému. Pro každý případ užití nejprve stručně vymezím, co je jeho náplní a pro které uživatele je relevantní. Následně rozepíši kroky základního scénáře a vymezím jeho alternativy.

Pro přehledné zobrazení pokrytí funkčních požadavků případy užití, vizte tabulku B.1 v příloze této práce.

Vzhledem k relativně velkému překryvu některých případů užití zde nebudou všechny rozepsány podrobně. Namísto toho se raději odkážu na podobnosti a rozdíly mezi podobnými případy užití.

Pro zjednodušení označení popisovaných jevů jsem se rozhodl použít následující zkratky:

UC pro *případ užití* (z angl. „use case“)

GUCE pro *zobecněný prvek případu užití* (z angl. „generalized use case element“)

AS pro *alternativní scénář*

1.7.1 Zobecněné prvky případů užití

V průběhu popisu případů užití jsem se setkal s několika aspekty, které se až na kontext, v němž jsou použity, ničím neliší.

Pokud bych je stále dokola podrobně rozepisoval, s velkou pravděpodobností bych ohrozil pozornost čtenářů a jejich vůli pokračovat ve čtení. V případě jejich vynechání bych se vystavil riziku obvinění z nedůslednosti.

Z tohoto důvodu jsem se rozhodl je podrobněji rozepsat zde a následně se na ně pouze odkazovat. Pokud by se použití zobecněných prvků příliš lišilo od jejich definice, rozepíši tuto odlišnost podrobněji na místě jejich použití.

GUCE – Potvrzení akce

Napříč následujícími případy užití se vyskytují akce, jejichž vykonání může mít velký dopad na systém. Mezi takovéto akce patří například odesílání nákupů, úprava rolí uživatele či jakékoliv mazání.

Z důvodu zabránění uživateli, aby tyto akce zvolil omylem, budou tyto akce vyžadovat potvrzení. Toto potvrzení bude probíhat následovně:

1. Uživatel zvolí akci, jež vyžaduje potvrzení.
2. Systém uživatele upozorní, že daná akce musí být potvrzena, a zeptá se ho, zda chce opravdu danou akci vykonat.
3. Uživatel svou volbu potvrdí.
4. Systém danou akci vykoná a scénář původního případu užití pokračuje svým následujícím krokem.

V případě, že uživatel svou volbu nepotvrdí, vrátí se průběh před krok scénáře původního případu užití, který vyvolal spuštění akce vyžadující potvrzení.

GUCE – Zrušení akce

V každém z případů užití bude mít uživatel po celou dobu možnost zrušit provádění akce, jež daný případ užití popisuje.

Toto zrušení bude vyžadovat potvrzení, při němž bude uživatel upozorněn, že zrušením dojde ke ztrátě všech neuložených změn.

Po potvrzení zrušení akce se systém vrátí do stavu před započetím rušené akce. Pokud ovšem během akce uživatel vytvořil například nového dodavatele či typ potraviny, tyto změny nebudou zrušeny.

GUCE – Přidání položky do seznamu

Vytvářený systém obsahuje vícero míst, na nichž se pracuje se seznamem položek. Pakliže se nejedná o velmi komplexní položky, probíhá přidání nové položky do seznamu následujícím způsobem:

1. Uživatel zvolí možnost „Přidat ⟨*název položky*⟩“.

2. Systém zobrazí novou položku s potřebnými poli, jejichž hodnoty budou nastaveny na základní hodnoty.
3. Uživatel pole vyplní a zvolí možnost „Uložit“.
4. Systém přidá do seznamu vytvořenou položku.

Přidávání položky do seznamu může být zrušeno.

GUCE – Úprava položky seznamu

Položky v seznamu mohou také nabízet možnost úpravy. Tato možnost bude vždy zobrazena u dané položky. Úprava položky probíhá následovně:

1. Uživatel zvolí možnost „Upravit ⟨*název položky*⟩“.
2. Systém zobrazí u zvolené položky pole pro úpravu, jejichž hodnoty budou nastaveny na původní hodnoty upravované položky.
3. Uživatel pole upraví a zvolí možnost „Uložit“.
4. Systém upraví hodnoty upravované položky tak, aby odpovídaly hodnotám zadaným uživatelem.

Upravování položky může být zrušeno.

GUCE – Smazání položky seznamu

Uživatel bude mít ve většině seznamů možnost položky smazat. Tato možnost bude podobně jako u úpravy zobrazena u dané položky.

Smazání položky může vyžadovat potvrzení. V případech, jichž se to týká, tuto skutečnost explicitně zmíním.

GUCE – Vybrání existující hodnoty

Napříč systémem je několik míst, v nichž uživatel bude potřebovat vybrat jednu z existujících hodnot určitého typu. Výběr hodnoty bude probíhat následovně:

1. Uživatel zvolí možnost „Vybrat ⟨*typ hodnoty*⟩“.
2. Systém zobrazí pole pro vyhledání hodnoty. Společně s ním zobrazí pod tímto polem také (zkrácený) seznam existujících hodnot.
3. Uživatel začne zadávat název požadované hodnoty.
4. Systém bude postupně filtrovat hodnoty v seznamu tak, aby v něm byly obsažené pouze ty, které mají stejný prefix jako doposud zadaná hodnota uživatelem.

5. Uživatel bude pokračovat v zadávání názvu do doby, než se jím požadovaná hodnota objeví v zobrazeném seznamu. Následně na tuto hodnotu v seznamu klikne, čímž ji zvolí.

GUCE – Přidání nové hodnoty

Může se stát, že uživatel při vybírání existující hodnoty zjistí, že požadovaná hodnota zatím neexistuje. V takovém okamžiku bude mít možnost přidat novou hodnotu daného typu. Proces přidávání nové hodnoty probíhá následovně:

1. Uživatel zvolí možnost „Přidat nový ⟨*typ hodnoty*⟩“.
2. Systém zobrazí pole pro zadání názvu nové hodnoty. Zároveň zobrazí seznam již existujících hodnot tohoto typu.
3. Uživatel začne zadávat název nové hodnoty.
4. Systém bude postupně filtrovat hodnoty v seznamu tak, aby v něm zbyly pouze hodnoty, jež jsou podobné názvu vytvářené hodnoty. Toto opatření si klade za cíl předejít neúmyslnému vytváření duplicitních hodnot.
5. Uživatel dopíše název nové hodnoty a zvolí možnost „Vytvořit nový ⟨*typ hodnoty*⟩“. Tato akce bude vyžadovat potvrzení.
6. Systém zadanou hodnotu vytvoří a použije ji jako vybranou.

Pokud by uživatel při zadávání zjistil, že již existuje hodnota, jež mu bude vyhovovat, může tuto hodnotu v seznamu označit a zvolit možnost „Použít zvolenou existující hodnotu“. V takovém případě systém nevytvoří novou hodnotu a pouze použije zvolenou hodnotu jako vybranou.

Přidání hodnoty není implicitně povoleno. Pokud tato možnost povolena při výběru hodnot bude, explicitně to u daného případu zmíním.

Proces přidávání nové hodnoty může být zrušen.

1.7.2 UC – Vytvoření nákupu

Tento případ užití popisuje, jakým způsobem docílí uživatel vytvoření nákupu v systému. Roli uživatele mohou v tomto případu užití plnit zaměstnanci obchodního oddělení („velký nákup“) i zaměstnanci jednotlivých areálů („malý“ či „každodenní nákup“).

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Vytvořit nákup“.

2. Systém zobrazí obrazovku umožňující vytvoření nákupu. Tato obrazovka bude obsahovat prvky umožňující výběr dodavatele a (prozatím) prázdný seznam položek.
3. Uživatel vybere dodavatele. V rámci tohoto výběru bude umožněno vytvořit nového dodavatele.
4. Systém uloží volbu dodavatele.
5. Uživatel přidá položku. Její atributy budou potravina, množství a cena. Výběr potraviny je podrobněji popsán v UC – Výběr potraviny.
6. Systém zobrazí aktuální podobu seznamu položek nákupu.
7. Uživatel zvolí možnost „Dokončit nákup“. Tato možnost bude vyžadovat potvrzení.
8. Systém uloží informace spojené s nákupem, vytvoří upozornění pro zaměstnance areálu, do něhož nákup směřuje, a zobrazí opět hlavní obrazovku systému.

Pakliže je uživatelem zaměstnanec areálu, bude moci navíc zvolit možnost „Vytvořit jako každodenní“. V takovém případě systém kromě vytvoření nákupu vytvoří také každý den notifikaci, která bude uživatele upozorňovat, že by měl nákup zopakovat.

Po rozkliknutí notifikace systém zobrazí nákup se stejným obsahem i dodavatelem, jaké měl nákup původní, a průběh bude pokračovat krokem 6 základního scénáře.

AS – Zvolení cílového areálu

Tento alternativní scénář začíná po kroku 3 základního scénáře v případě, že uživatelem je zaměstnanec obchodního oddělení a nákup nevytváří na základě požadavku zaměstnance areálu.

Pokud nákup vytváří zaměstnanec areálu, je za cílový areál automaticky zvolen areál, do něhož tento zaměstnanec přísluší.

Podobně tomu bude i v případě, že je nákup vytvářen na základě požadavku zaměstnance areálu. V takovém případě bude automaticky zvolen areál, k němuž přísluší zaměstnanec, jež požadavek vytvořil.

Zvolení areálu probíhá podle GUCE – Vybrání existující hodnoty. Po dokončení pokračuje průběh krokem 4 základního scénáře.

AS – Úprava seznamu

Tento scénář začíná po kroku 6 základního scénáře v případě, že uživatel není hotov s úpravami seznamu. V takovém případě přidá další položku, některou z existujících upraví nebo smaže.

Po dokončení pokračuje průběh krokem 6 základního scénáře.

AS – Uložení rozpracovaného nákupu

Tento scénář začíná po kroku 6 základního scénáře v případě, že se uživatel rozhodne přerušit vytváření nákupu s možností dokončit vytváření později.

1. Uživatel zvolí možnost „Uložit jako rozpracovaný“.
2. Systém uloží současný stav nákupu a vrátí uživatele na hlavní obrazovku.

AS – Rozdělení nákupu

Tento scénář začíná po kroku 6 základního scénáře v případě, že se uživatel rozhodne vyčlenit některé položky současného nákupu do nového nákupu.

1. Uživatel zvolí možnost „Rozdělit nákup“.
2. Systém zpřístupní možnost označování položek nákupu a vyzve uživatele, aby označil položky, které si přeje vyčlenit do nového nákupu.
3. Uživatel označí položky, které si přeje vyčlenit do nového nákupu a zvolí možnost „Přesunout označené položky do nového nákupu“. Tato akce bude vyžadovat potvrzení.
4. Systém vytvoří nový nákup ve stavu „Rozpracovaný“ se stejným dodavatelem a cílovým areálem, přesune do něj označené položky a zároveň tyto označené položky odstraní ze současného nákupu.

Po skončení scénáře pokračuje průběh krokem 6 základního scénáře.

1.7.3 UC – Výběr potraviny

Tento případ užití popisuje, jakým způsobem budou moci uživatelé vybrat potravinu v rámci úpravy vlastností položky nákupu, požadavku na nákup či receptu. Roli uživatele mohou v tomto případu užití plnit zaměstnanci obchodního oddělení i zaměstnanci jednotlivých areálů.

Základní scénář

Případ užití začíná na obrazovce obsahující prvky pro výběr potraviny a pokračuje následovně:

1. Uživatel vybere typ potraviny. V rámci tohoto výběru bude umožněno vytvořit nový typ. Při vytváření nového typu bude uživatel muset specifikovat jednotky, s nimiž má být nový typ asociovaný.
2. Systém uloží volbu typu potraviny a zpřístupní volbu konkrétní potraviny.

3. Uživatel pokračuje výběrem konkrétní potraviny podle případu užití UC – Výběr konkrétní potraviny.

Vzhledem k tomu, že požadavek na nákup a recept nevyžadují výběr konkrétní potraviny (tj. pouze typ postačí), může se uživatel v těchto případech rozhodnout výběr konkrétní potraviny přeskočit.

1.7.4 UC – Výběr konkrétní potraviny

Tento případ užití popisuje, jakým způsobem budou moci uživatelé vybrat konkrétní potravinu z vybraného typu potravin. Roli uživatele mohou v tomto případu užití plnit zaměstnanci obchodního oddělení i zaměstnanci jednotlivých areálů.

Výběr konkrétní potraviny probíhá podle GUCE – Vybrání existující hodnoty. Uživatelé budou mít možnost vytvořit novou konkrétní potravinu.

1.7.5 UC – Úprava rozpracovaného nákupu

Tento případ užití popisuje způsob, jakým bude uživatel moci pokračovat v úpravě nákupu, který v minulosti uložil jako rozpracovaný. Roli uživatele mohou v tomto případu užití plnit zaměstnanci obchodního oddělení i jednotlivých areálů.

Vzhledem k velkému překryvu s UC – Vytvoření nákupu zde rozepíši detailně pouze části, v nichž se tyto dva případy užití liší.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Zobrazit rozpracované nákupy“.
2. Systém zobrazí seznam nákupů, které uživatel má uživatel uložené jako rozpracované.
3. Uživatel si v seznamu vybere nákup, v jehož úpravě chce pokračovat, a zvolí možnost „Pokračovat v úpravě nákupu“.
4. Případ užití od této chvíle pokračuje téměř identicky s UC – Vytvoření nákupu (od kroku 6 základního scénáře) včetně alternativních scénářů a omezení.

1.7.6 UC – Vytvoření požadavku nákupu

Tento případ užití popisuje, jakým způsobem může uživatel vytvořit požadavek („velkého“) nákupu. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Kvůli podobnosti s případem užití UC – Vytvoření nákupu zde zmíním pouze aspekty, v nichž se tyto případy užití liší.

Výjimky

Případ užití bude taktéž začínat na hlavní obrazovce systému, avšak uživatel zvolí možnost „Vytvořit požadavek nákupu“.

Vzhledem k tomu, že cílovým areálem budoucího nákupu bude vždy areál, k němuž uživatel přísluší, dojde k vynechání kroků se zadáváním cílového areálu.

Zaměstnanci areálu u „velkých nákupů“ nerozhodují o dodavateli. Z tohoto důvodu dojde také k vynechání kroků se zadáváním cílového areálu.

Další změna se týká položek požadavku nákupu. Tyto položky budou obsahovat pouze atributy potravina a množství, nikoliv však cena. Zároveň nebude nutné vybírat konkrétní potravinu.

Při vytváření požadavku nákupu nebudou moci uživatelé zvolit možnost „Vytvořit jako každodenní“.

1.7.7 UC – Úprava rozpracovaného požadavku nákupu

Tento případ užití popisuje způsob, jakým bude uživatel moci pokračovat v úpravě požadavku nákupu, který v minulosti uložil jako rozpracovaný. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Až na drobnosti kopíruje tento případ užití kroky UC – Úprava rozpracovaného nákupu rozšířeného o výjimky zmíněné v případu užití UC – Vytváření požadavku nákupu. Z tohoto důvodu ho zde nebudu podrobněji rozebírat.

1.7.8 UC – Vyřízení požadavku nákupu

Tento případ užití popisuje způsob, jakým se bude uživatel moci přihlásit k vyřízení požadavku nákupu. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Zobrazit nevyřízené požadavky nákupu“.
2. Systém zobrazí seznam požadavků čekajících na vyřízení. Každá položka seznamu bude obsahovat datum vytvoření, jméno autora a jméno areálu, do něhož má nákup směřovat.
3. Uživatel zvolí možnost „Zobrazit požadavek nákupu“.
4. Systém zobrazí seznam s položkami požadavku nákupu.

5. Uživatel zvolí možnost „Vyřídit požadavek nákupu“. Tato akce bude vyžadovat potvrzení.

6. Systém odstraní požadavek ze seznamu požadavků čekajících na vyřízení. Scénář pokračuje dále jako v případu užití UC – Úprava rozpracovaného nákupu.

AS – Odmítnutí vyřízení požadavku

Tento scénář začíná v kroku 5 základního scénáře v případě, že se uživatel rozhodne, že požadavek nákupu nemá být vyřízen.

1. Uživatel zvolí možnost „Zamítnout požadavek“.

2. Systém vyzve uživatele, aby zadal důvod zamítnutí požadavku.

3. Uživatel vyplní pole pro udání důvodu a zvolí možnost „Zamítnout s daným důvodem“. Tato možnost bude vyžadovat potvrzení.

4. Systém odstraní požadavek ze seznamu požadavků čekajících na vyřízení a vytvoří notifikaci pro zaměstnance areálu, který požadavek vytvořil, oznamující mu důvod zamítnutí požadavku.

1.7.9 UC – Příjem nákupu

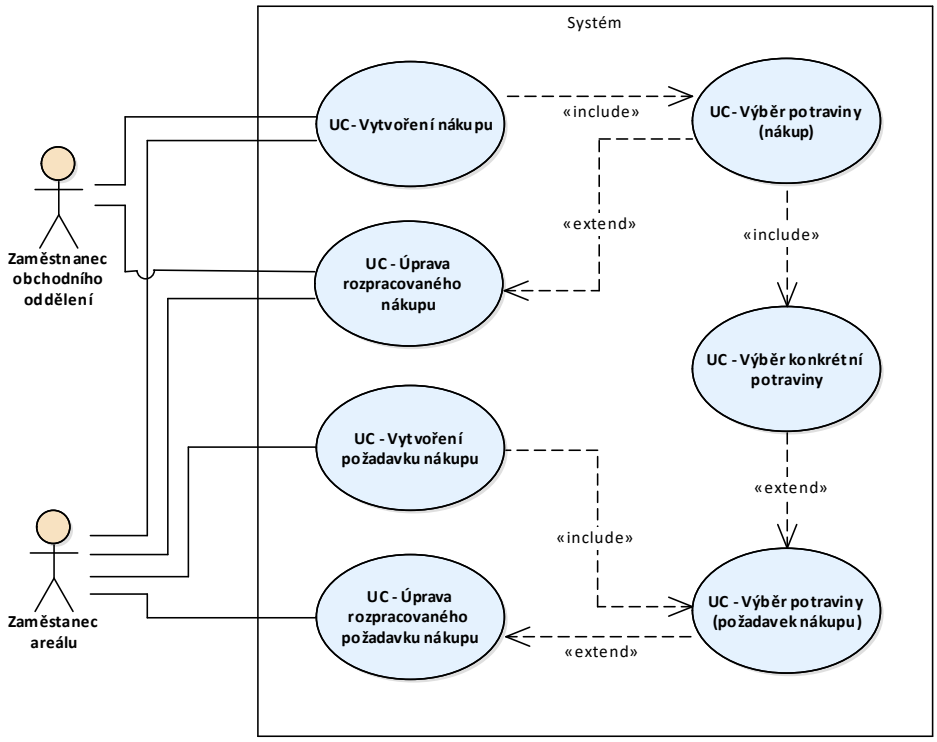
Tento případ užití popisuje způsob, jakým bude uživatel moci evidovat příjem nákupu. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Základní scénář

Případ užití začíná na obrazovce s přehledem položek nákupu a pokračuje následovně:

1. Uživatel zkontroluje, že položky nákupu odpovídají převzatému zboží, a zvolí možnost „Naskladnit nákup“. Tato akce bude vyžadovat potvrzení.

2. Systém připíše specifikované množství potravin do zásob skladu, uloží informace o převzetí nákupu a vrátí uživatele na hlavní obrazovku.



Obrázek 1.6: Diagram případů užití spojených s nákupem

AS – Úprava položek přijímaného nákupu

Tento scénář začíná v kroku 1 základního scénáře v případě, že se položky nákupu neshodují s převzatým zbožím.

V takovém případě bude uživatel moci upravit množství či smazat položky z nákupu. Tyto úkony vykoná podle odpovídajících alternativních scénářů případu užití UC – Vytvoření nákupu.

Jakmile stav nákupu odpovídá převzatému zboží, vrací se scénář do kroku 1.

1.7.10 UC – Vytvoření receptu

Tento případ užití popisuje způsob, jakým bude uživatel moci vytvořit recept. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Vytváření receptu se velmi podobá případu užití UC – Vytváření požadavku nákupu. Z tohoto důvodu zde nebudu znova rozepisovat celý případ užití, ale zmíním pouze body, v nichž se tyto případy užití liší.

Výjimky

Případ užití bude taktéž začínat na hlavní obrazovce systému, avšak uživatel zvolí možnost „Vytvořit recept“.

Každý recept potřebuje ke své identifikaci jméno. Pro zadání tohoto jména bude při vytváření (a úpravě) receptu zobrazeno pole. Bez vyplnění jména receptu nebude možné recept uložit. Jméno receptu bude muset být v rámci receptů uživatele unikátní.

Zároveň bude nutné u každého receptu určit, kolika porcím odpovídá množství surovin uvedených v receptu. Tuto informaci bude muset uživatel zadat do k tomuto účelu určeného pole. Bez zadání této informace nebude možné recept uložit.

Na rozdíl od nákupu a požadavku nákupu nejsou recepty jednotlivých uživatelů dále šířeny systémem (za předpokladu, že z nich nebudou vytvořeny šablony receptů). Díky této skutečnosti není potřeba vytváření receptů potvrzovat.

Zároveň také nemá smysl ukládat recepty jako rozpracované – pokud uživatel potřebuje úpravu přerušit, zkrátka recept uloží a následně upraví dříve uložený recept.

Z tohoto důvodu nejsou přístupné možnosti, kroky a alternativní scénáře spojené s těmito úkony.

Místo možnosti dokončení požadavku bude možnost „Uložit recept“, která uloží recept k ostatním uživatelovým receptům.

1.7.11 UC – Úprava receptu

Tento případ užití popisuje způsob, jakým bude uživatel moci upravit existující recept. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Upravování receptu se řídí stejným základním průběhem jako případ užití UC – Úprava rozpracovaného požadavku nákupu s odpovídajícími výjimkami případu užití UC – Vytvoření receptu.

1.7.12 UC – Smazání receptu

Tento případ užití popisuje způsob, jakým bude uživatel moci smazat existující recept. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Scénář pokračuje podle vzoru GUCE – Smazání položky seznamu. Tato akce bude vyžadovat potvrzení.

1.7.13 UC – Vytvoření šablony receptu

Tento případ užití popisuje způsob, jakým bude uživatel moci vytvořit ze svého receptu šablonu. Roli uživatele mohou v tomto případu užití plnit pouze za-

městnanci areálů.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Zobrazit mé recepty“.
2. Systém zobrazí seznam receptů uživatele.
3. Uživatel zvolí možnost „Vytvořit šablonu z receptu“ přidruženou k receptu, z něhož chce vytvořit šablonu. Tato akce bude vyžadovat potvrzení.
4. Systém vytvoří upozornění pro zaměstnance obchodního oddělení ohledně nového návrhu šablony receptu.

1.7.14 UC – Vytvoření receptu ze šablony

Tento případ užití popisuje způsob, jakým si bude uživatel moci vytvořit z existující šablony recept. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Zobrazit šablony receptů“.
2. Systém zobrazí seznam šablon receptů.
3. Uživatel zvolí možnost „Vytvořit recept ze šablony“ přidruženou k šabloně, z níž si chce vytvořit recept.
4. Případ užití pokračuje stejně jako případ užití UC – Vytvoření receptu s tím, že systém nastaví jméno receptu a jeho položky tak, aby odpovídaly šabloně, z níž je recept vytvářen.

1.7.15 UC – Výběr potravin ze skladu

Tento případ užití popisuje způsob, jakým bude uživatel moci evidovat výběr potravin ze skladu. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Vybrat potraviny ze skladu“.
2. Systém zobrazí obrazovku s prázdným seznamem položek, které si uživatel přeje vybrat ze skladu.
3. Uživatel naplní seznam položkami podobně jako v UC – Vytváření požadavku nákupu. Jedinou změnou je, že u položek bude muset specifikovat konkrétní potravinu.
4. Jakmile uživatel zadá všechny položky, zvolí možnost „Potvrdit výběr“. Tato akce bude vyžadovat potvrzení.
5. Systém odebere zadané množství položek ze skladu, uloží záznam o výběru a vrátí uživatele na hlavní obrazovku.

Systém bude u každé položky zobrazovat, jaké množství jí zbývá na skladě. V případě, že by se uživatel pokoušel vybrat větší množství než je na skladě k dispozici, systém uživatele na tuto skutečnost upozorní a nedovolí mu zvolit možnost „Potvrdit výběr“, dokud hodnoty nebudou v pořádku.

1.7.16 UC – Výběr potravin ze skladu podle receptu

Tento případ užití popisuje způsob, jakým bude uživatel moci vybrat potraviny ze skladu podle receptu. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálů.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Vybrat potraviny ze skladu podle receptu“.
2. Systém zobrazí seznam receptů uživatele.
3. Uživatel zvolí v seznamu recept, podle něhož chce vybrat potraviny.
4. Systém se uživatele dotáže na počet stravovaných. Pokud je tento počet v systému nastaven, bude pole pro zadání počtu stravovaných na začátku vyplněno touto hodnotou.
5. Uživatel tuto hodnotu zkontroluje (a případně upraví) a potvrdí ji.
6. Scénář pokračuje krokem 2 případu užití UC – Výběr potravin ze skladu s tím, že seznam je naplněn položkami podle specifikace v receptu a aktuálně nastaveným počtem strávníků.

1.7.17 UC – Převoz potravin

Tento případ užití popisuje způsob, jakým bude uživatel moci iniciovat převoz potravin mezi areály. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Připravit převoz zboží“.
2. Systém vyzve uživatele, aby zadal výchozí a cílový areál.
3. Uživatel vybere, který areál má být výchozí a který má být cílový.
4. Systém zobrazí obrazovku s prázdným seznamem položek, které uživatel chce převézt mezi zvolenými areály.
5. Uživatel naplní seznam stejně jako v UC – Výběr potravin ze skladu.
6. Jakmile uživatel zadá všechny položky, zvolí možnost „Zahájit převoz“. Tato akce bude vyžadovat potvrzení.
7. Systém vytvoří notifikaci pro zaměstnance výchozího areálu o naložení zboží a uloží informace o převozu zboží.

Systém bude stejně jako v případu užití UC – Výběr potravin ze skladu kontrolovat stav zásob ve výchozím areálu a nedovolí zahájení převozu, pokud by se uživatel snažil převézt více potravin, než je ve výchozím areálu na skladě.

Zároveň bude uživatel moci po celou dobu upravit výchozí a cílový areál podobně, jako se upravuje cílový areál při vytváření „velkého“ nákupu (vizte odpovídající alternativní scénář UC – Vytvoření nákupu).

1.7.18 UC – Naložení potravin pro převoz

Tento případ užití popisuje způsob, jakým bude uživatel moci potvrdit naložení potravin určených pro převoz mezi areály. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálu, jež byl označen jako výchozí v daném převozu potravin.

Scénář probíhá téměř totožně se scénářem v případu užití UC – Výběr potravin ze skladu s tím, že seznam položek bude naplněn položkami, které byly zvolené pro převoz.

Po potvrzení naložení potravin systém navíc vytvoří notifikaci pro zaměstnance cílového areálu o příchozích potravinách.

1.7.19 UC – Příjem potravin v rámci převozu

Tento případ užití popisuje způsob, jakým bude uživatel moci potvrdit příjem potravin určených pro převoz mezi areály. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci areálu, jež byl označen jako cílový v daném převozu potravin.

Scénář probíhá identicky se scénářem v případu užití UC – Příjem nákupu. Jedinou výjimkou budou upravené názvy akcí.

1.7.20 UC – Zobrazení přehledu výdajů areálů

Tento případ užití popisuje způsob, jakým si bude uživatel moci nechat zobrazit přehled výdajů areálů. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Zobrazit přehledy výdajů“.
2. Systém zobrazí seznam uložených přehledů.
3. Uživatel zvolí možnost „Zobrazit přehled“ přidruženou k přehledu, který si chce nechat zobrazit.
4. Systém zobrazí tabulku naplněnou daty odpovídajícími nastavení zvoleného přehledu.

Po zobrazení výsledků přehledu bude mít uživatel následující možnosti:

Zavřít náhled Systém se po zvolení této možnosti vrátí na obrazovku se seznamem přehledů. Pokud bude náhled obsahovat neuložené změny, systém uživatele na tuto skutečnost upozorní a zeptá se ho, zda si přeje změny uložit.

Uložit náhled Tato možnost bude dostupná pouze, pokud náhled bude obsahovat neuložené změny v nastavení přehledu. Při zvolení této možnosti systém uloží neuložené změny.

Upravit náhled Pokud uživatel zvolí tuto možnost, systém bude pokračovat případem užití UC – Úprava přehledu výdajů areálu.

1.7.21 UC – Vytvoření přehledu výdajů areálů

Tento případ užití popisuje způsob, jakým bude uživatel moci vytvořit nový přehled výdajů areálů. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Základní scénář

Případ užití začíná na obrazovce se seznamem uložených přehledů a pokračuje následovně:

1. Uživatel zvolí možnost „Vytvořit nový přehled výdajů“.
2. Systém zobrazí obrazovku s nastavením přehledu.
3. Uživatel vyplní potřebná data a zvolí možnost „Vytvořit přehled“.
4. Systém uloží informace o přehledu a vrátí uživatele zpět na původní obrazovku.

Nastavení přehledu bude obsahovat následující možnosti:

- Pole pro název přehledu (*povinné*)
- Pole pro zadání rozmezí dat, mezi nimiž mají být výdaje počítány (*povinné*)
- Seznam s areály, které mají být zahrnuty v přehledu (minimálně jeden musí být zvolen)
- Seznam s položkami, které mají být porovnány. Přidávání položek bude probíhat identicky s odpovídajícími alternativními scénáři případu užití UC – Vytvoření nákupu. Každá položka bude obsahovat pouze typ potraviny nebo konkrétní potravinu.
- Dodatečné možnosti:

  Zahrnout celkové výdaje Při zvolení této možnosti bude do porovnání zahrnuta položka s celkovými výdaji za dané období. Tato možnost *bude* na začátku zvolena.

  Přepočítat na osobu Pokud bude tato možnost zvolena, veškeré výdaje budou přepočítány na počet osob, které se ve zvoleném období v daném areálu stravovaly. Pokud se tento počet osob v daném období měnil, vypočítá se hodnota pro každou z hodnot a výsledky se zprůměrují. Tato možnost *nebude* na začátku zvolena.

  Vztáhnout vůči areálu Po zvolení této možnosti bude muset uživatel vybrat areál, vůči kterému se budou přehledy ostatních areálů vztahovat. Hodnoty tohoto areálu budou nastaveny na nuly, ostatní areály pak budou mít hodnoty relativní vůči tomuto areálu. Zvolení této možnosti bude podmíněno vybráním alespoň dvou areálů. Z tohoto důvodu *nebude* tato možnost na začátku zvolena.

Všechny seznamy a pole budou na začátku prázdné. Uživatel bude mít také možnost vytvořený přehled přímo zobrazit.

1.7.22 UC – Úprava přehledu výdajů areálu

Tento případ užití popisuje způsob, jakým bude uživatel moci upravit existující přehled výdajů areálů. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Scénář tohoto případu užití je téměř identický se scénářem UC – Vytvoření nového přehledu výdajů areálů. Jedinými rozdíly jsou názvy možností a původní stav nastavení, který bude odpovídat původnímu stavu upravovaného přehledu.

1.7.23 UC – Smazání přehledu výdajů areálu

Tento případ užití popisuje způsob, jakým bude uživatel moci smazat existující přehled výdajů areálů. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Scénář pokračuje podle vzoru GUCE – Smazání položky seznamu. Smazání bude vyžadovat potvrzení.

1.7.24 UC – Zobrazení stavu skladu

Tento případ užití popisuje způsob, jakým si bude uživatel moci nechat zobrazit stav skladu. Roli uživatele mohou v tomto případu užití plnit zaměstnanci obchodního oddělení i zaměstnanci jednotlivých areálů.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Zobrazit stav skladu“.
2. Systém uživatele vyzve, aby vybral areál, ve kterém chce zobrazit stav skladu.
3. Uživatel vybere areál a svou volbu potvrdí.
4. Systém zobrazí seznam položek, které jsou právě ve vybraném skladu. Položky budou obsahovat konkrétní potravinu a množství.

V případě, že je uživatelem zaměstnanec areálu, pokračuje se po kroku 1 rovnou krokem 4 s tím, že zvoleným areálem bude areál, k němuž zaměstnanec náleží.

1.7.25 UC – Nastavení kontroly množství potravin

Tento případ užití popisuje způsob, jakým bude uživatel moci nastavit kontrolu množství potravin. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci jednotlivých areálů.

Kontrola množství probíhá nastavením mezních hodnot pro konkrétní potraviny. Jakmile klesne množství potraviny na skladě areálu, k němuž zaměstnanec přísluší, pod nastavenou mezní hodnotu, systém upozorní uživatele na tuto skutečnost.

Toto upozornění bude odstraněno, až když množství potraviny na skladě stoupne nad mezní hodnotu nebo bude daná kontrola změněna či odstraněna.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Spravovat kontrolu potravin“
2. Systém zobrazí obrazovku s aktuálně nastavenými kontrolními hranicemi.
3. Uživatel přidá kontrolu. Pro zadání informací bude potřeba prvek rozhraní pro zadání konkrétní potraviny a pole pro zadání mezní hodnoty množství.
4. Systém vytvoří novou kontrolu podle zadaných hodnot a vrátí uživatele na obrazovku s aktuálně nastavenými kontrolami.

Uživatel bude mít zároveň možnost kontroly v seznamu upravovat a mazat.

1.7.26 UC – Nastavení počtu strávníků

Tento případ užití popisuje způsob, jakým bude uživatel moci nastavit počet ubytovaných. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci jednotlivých areálů.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Nastavit počet strávníků“.
2. Systém zobrazí seznam nastavených záznamů od aktuálního dne.
3. Uživatel přidá nový záznam. V rámci přidávání budou zobrazena pole pro datum začátku a konce doby, po níž budou strávníci ubytováni, a pole pro zadání počtu strávníků.
4. Systém vytvoří nový záznam podle zadaných informací a vrátí uživatele na obrazovku se seznam nastavených záznamů.

Kromě vytváření bude mít uživatel také možnost záznamy upravovat a mazat. Těmito možnostmi ovšem nebude moci měnit data v minulosti.

1.7.27 UC – Zobrazení historie

Tento případ užití popisuje způsob, jakým si bude uživatel moci nechat zobrazit historii provedených akcí. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Obrazovka se záznamy provedených akcí bude dostupná z hlavní obrazovky systému. Položky tohoto seznamu budou obsahovat:

- Datum provedení akce
- Typ akce (nákup, požadavek, převoz, ...)
- Jméno uživatele, který akci provedl
- Areál(y), s nímž akce souvisí

Uživatel bude mít možnost akce pomocí těchto atributů filtrovat a nechat si zobrazit detail jednotlivých akcí.

1.7.28 UC – Schválení šablony receptu

Tento případ užití popisuje způsob, jakým bude uživatel moci schvalovat šablony receptů. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Základní scénář

Případ užití začíná na hlavní obrazovce systému a pokračuje následovně:

1. Uživatel zvolí možnost „Zobrazit šablony receptů ke schválení“.
2. Systém zobrazí seznam šablon ke schválení.
3. Uživatel si vybere šablonu, kterou chce schválit, a zvolí k ní přidruženou možnost „Schválit šablonu“.
4. Systém šablonu přidá k ostatním schváleným šablonám a zpřístupní ji všem zaměstnancům areálu.

Uživatel bude mít také možnost prohlédnout si recept, který je navržen na šablonu. Po zvolení této možnosti systém zobrazí přehled receptu jako v případu užití UC – Úprava receptu. Uživatel však nebude moci zobrazené hodnoty upravovat.

AS – Zamítnutí šablony

Tento scénář v kroku 3 základního scénáře v případě, že se uživatel rozhodne zamítnout vybranou šablonu. Systém se uživatele zeptá, zda chce opravdu šablonu smazat, a v případě potvrzení návrh smaže.

1.7.29 UC – Smazání šablony receptu

Tento případ užití popisuje způsob, jakým bude uživatel moci mazat šablony receptů. Roli uživatele mohou v tomto případu užití plnit pouze zaměstnanci obchodního oddělení.

Scénář se řídí podle GUCE – Smazání položky seznamu. Smazání šablony receptu bude vyžadovat potvrzení.

1.7.30 UC – Správa hodnot

Tento případ užití popisuje způsob, jakým bude uživatel moci spravovat existující hodnoty systému. Roli uživatele mohou v tomto případu užití plnit pouze uživatelé se speciálními právy.

Uživatel bude mít na hlavní obrazovce systému možnost nechat si zobrazit všechny hodnoty, které jsou v systému uložené v rámci následujících typů:

- Konkrétní potraviny
- Typy potravin
- Dodavatelé

Po zvolení této možnosti systém zobrazí seznam hodnot daného typu. V rámci těchto seznamů bude mít uživatel možnost záznamy upravovat a mazat. Mazání hodnot bude vyžadovat potvrzení.

1.7.31 UC – Správa uživatelů

Tento případ užití popisuje způsob, jakým bude uživatel moci spravovat ostatní uživatele systému. Roli uživatele mohou v tomto případu užití plnit pouze uživatelé se speciálními právy.

Uživatel bude mít na hlavní obrazovce systému možnost nechat si zobrazit seznam všech uživatelů systému. Systém zobrazí seznam uživatelů s tím, že každá položka seznamu bude obsahovat jméno, typ a kontaktní email uživatele.

Na obrazovce s tímto seznamem bude mít uživatel následující možnosti:

Vytvoření nového uživatele Zde bude muset uživatel zadat kontaktní e-mailovou adresu a jméno nového uživatele. V případě, že bude uživatel přidávat zaměstnance areálu, bude muset zvolit i areál, k němuž bude zaměstnanec náležet. Systém následně pošle na zadanou emailovou adresu instrukce pro první přihlášení a nastavení hesla.

Úprava typu uživatele Po zvolení této možnosti zadá uživatel nový typ uživatele, k jehož položce v seznamu byla možnost přidružena. V případě, že bude uživatel přidávat zaměstnance areálu, bude muset zvolit i areál, k němuž bude zaměstnanec náležet. Tato akce bude vyžadovat potvrzení.

Úprava příslušnosti k areálu Uživatel bude moci zvolit tuto možnost pouze u zaměstnanců areálu. Po zvolení vybere areál, k němuž má zvolený zaměstnanec nově náležet. Tato akce bude vyžadovat potvrzení.

Smazání uživatele Tato možnost bude vyžadovat potvrzení. Po potvrzení bude uživatel smazán ze systému i ze všech existujících záznamů, v nichž jeho jméno bude nahrazeno zástupnou hodnotou.

Po provedení jakékoliv z těchto možností bude uživateli, jehož se změna týkala, odeslán na kontaktní emailovou adresu email informující ho o provedených změnách.

Vzhledem k tomu, jak jsou tyto procesy v informačních systémech běžné, je zde nebudu dále rozebírat.

1.7.32 UC – Správa areálů

Tento případ užití popisuje způsob, jakým bude uživatel moci spravovat areály v rámci systému. Roli uživatele mohou v tomto případu užití plnit pouze uživatelé se speciálními právy.

Uživatel bude mít na hlavní obrazovce systému možnost nechat si zobrazit seznam všech areálů. Každý z areálů je identifikován jménem, které musí být jedinečné.

V rámci tohoto seznamu bude uživatel moci areály přidat, upravit či smazat. Úprava a smazání budou vyžadovat potvrzení.

1.7.33 Další případy užití

V této části zmíním další případy užití, které se vyskytují v každém běžném informačním systému, a které tedy podle mého názoru není třeba podrobně rozepisovat, neboť by pouze kopírovaly již zaběhnuté standardy.

Z tohoto důvodu je nyní pouze vyjmenuji a případně k nim dodám vysvětlení či upřesnění. Těmito případy užití jsou:

- *UC – Přihlášení do systému*; Přihlašování bude probíhat pomocí kontaktního emailu a uživatelského hesla.
- *UC – Obnova zapomenutého hesla*; Odkaz pro obnovu hesla bude zaslán na kontaktní emailovou adresu.
- *UC – Správa účtu*; Uživatel bude moci upravovat svoji kontaktní emailovou adresu a měnit si heslo.
- *UC – Správa oblíbených položek*; Přepínání bude realizováno pomocí „hvězdičky“ u typů potravin, konkrétních potravin a dodavatelů.
- *UC – Posílání zpráv*; Aplikace bude umožňovat posílání textových zpráv v rámci konverzací.

Kapitola 2

Návrh

Druhá kapitola této práce je věnovaná mé snaze formalizovat myšlenkové pochody spojené s procesem návrhu vyvíjeného systému.

V této kapitole se nejprve zaměřím na výběr typu vyvíjeného systému. Zmíním kritéria, na jejichž základě jsem typ vybíral, zhodnotím dostupné možnosti v kontextu těchto kritérií a jednu z nich vyberu.

Následně zde vylíčím proces volby technologie, v níž systém budu vyvíjet, a důvody, které k jejímu zvolení vedly.

Zároveň se zde budu věnovat návrhu uživatelského rozhraní (resp. jeho *Lo-Fi* prototypu). Popíši způsob, jakým návrh probíhal, a klíčové koncepty a zásady, kterých jsem se při tomto procesu snažil držet. Zároveň přiložím několik ukázek tohoto návrhu.

V rámci této kapitoly také rozeberu návrh databázového modelu vyvíjeného systému včetně jeho schématu.

2.1 Výběr typu systému

V průběhu analýzy jsem narazil na několik různých způsobů, jakými by se dal vyvíjený systém sestavit. Nejčastěji používanými přístupy byly *desktopová aplikace s přístupem ke sdílenému úložišti* a *webová aplikace*. Z tohoto důvodu jsem se rozhodl použít jeden z nich.

Pro porovnání těchto dvou přístupů ke tvorbě systému jsem vymezil následujících sedm kritérií:

- Jakým způsobem bude probíhat synchronizace dat mezi instancemi systému?
- Která z variant bude finančně náročnější?
- Jak složité bude nasazení a aktualizace systému?
- Jak obtížné bude nalézání a ladění chyb?

- Bude možné využívat systém na mobilních zařízeních?
- Který z přístupů spíše preferují uživatelé?
- Jaké jsou mé zkušenosti s daným přístupem?

Nyní každé kritérium rozeberu podrobněji. Na závěr této sekce zhodnotím a porovnám celkovou vhodnost obou systémů a učiním finální rozhodnutí.

2.1.1 Synchronizace dat

Prvním kritériem pro výběr typu systému je způsob, jakým bude systém zajišťovat, aby všichni jeho uživatelé pracovali se stejnými daty.

V případě webové aplikace je tento aspekt vyřešen velmi snadno. Veškerá data by byla uložena v jediné databázi na serveru a uživatelé k nim budou skrze klienty přistupovat.

Desktopová aplikace by se mohla vydat stejným způsobem. Oproti webové aplikaci by však bylo možné, aby uživatelé pokračovali v práci bez připojení k internetu.

Toho by se docílilo použitím lokální databáze pro každou instanci systému. Tyto instance by se po obnovení připojení k internetu musely navzájem synchronizovat s centrálním úložištěm, což by mohlo vést ke konfliktům. V rámci prevence chyb by tyto konflikty museli řešit nejspíše zaměstnanci obchodního oddělení.

Po diskuzi s vedením agentury Haul jsem byl však ujištěn, že při návrhu systému mohu počítat s tím, že jednotlivé stroje zaměstnanců budou disponovat připojením k internetu. Z tohoto důvodu nepřipisuji této potenciální výhodě desktopové aplikace příliš velkou váhu.

2.1.2 Finanční stránka

Vzhledem k tomu, že webová aplikace musí běžet na serveru, vyvstává důležitá otázka, zdali by provoz tohoto serveru nepředstavoval pro agenturu Haul nadbytečné náklady, kterým by se potenciálně dalo zvolením desktopové aplikace vyhnout.

Přístup využívající desktopovou aplikaci však také vyžaduje přístup k centrálními úložišti dat. Toto úložiště bych musel umístit na nějaký server nebo bych mohl využít některé z cloudových řešení. Cena za využití těchto řešení je však srovnatelná či lehce vyšší než za pronájem serveru.

V současné době agentura Haul žádný server nevlastní ani si žádný nepronajímá. Jejich webové stránky spravuje Ing. Otto Zelenka, který mi sdělil, že jsou umístěné na soukromém serveru společnosti, pro níž pan Ing. Zelenka pracuje, a umístění mého systému na tento server nepřipadá v úvahu.

Souběžně s mojí prací vzniká i práce Anny Vitmanové [13], která se zabývá zjednodušením registrací na akce agentury Haul. Vzhledem k tomu, že Anna

plánuje tohoto zjednodušení dosáhnout vyvinutím a nasazením webové aplikace, bude muset agentura Haul stejně do pronájmu serveru investovat.

Desktopová aplikace by tedy v tomto bodě získala lehce navrch v případě, že bych byl schopný najít cloudové řešení centrálního úložiště cenově odpovídající pronájmu serveru, neboť bych se nemusel zabývat jeho implementací. Vzhledem k tomu, že agentura Haul bude muset server zajišťovat, zůstává webová aplikace stejně vhodným řešením.

2.1.3 Nasazení systému

Dalším klíčovým aspektem, který je potřeba při výběru typu systému zvážit, je způsob a obtížnost nasazování.

V případě webové aplikace vyžaduje nasazení nové verze systému aktualizaci pouze na serveru. Během nasazování může být systém na chvíli nedostupný, avšak toto časové rozmezí je většinou minimální a nemělo by při rozumném naplánování výrazně ovlivnit používání systému.

To samé ovšem ani zdaleka neplatí pro přístup využívající desktopovou aplikaci. V tomto případě by nasazení nové verze vyžadovalo nejen aktualizaci centrálního úložiště, ale také všech klientských instancí systému.

Zároveň by systém musel buď aktualizace vynutit znemožněním přístupu instancím, které nemají nejnovější verzi, nebo by se musel přizpůsobit situacím, kdy klientské instance mají různé verze.

Samotná aktualizace desktopové aplikace také bývá složitějším procesem vzhledem k nutnosti instalace doplňujících komponent a knihoven. Uživatelé však mohou být od tohoto martyria z velké části uchráněni technologiemi pro usnadnění instalace a aktualizace programů jako například *ClickOnce* [14].

V tomto ohledu tedy nekompromisně vítězí jednoduchost nasazení webové aplikace nad komplikacemi spojenými s nasazením všech instancí aplikace desktopové.

2.1.4 Ladění chyb

V rámci údržby vytvořeného systému bude potřeba nejen rozšiřovat systém o další funkce, ale zároveň opravovat chyby, jež se mohou při používání systému objevit. Z tohoto důvodu je rozumné se zamyslet nad tím, jak složitý proces ladění chyb bude.

Přístup využívající desktopovou aplikaci v tomto ohledu lehce pokulhává. Systém se totiž může chovat odlišně v závislosti na velkém množství faktorů jako je například verze operačního systému, nainstalované drivery či uživatelské nastavení.

Zároveň kvalita reportování chyby velmi závisí na uživateli, jež chybu nalezl. V případě, že tento uživatel není příliš technicky zdatný, se může replikování podmínek, při nichž chyba vznikla, velmi prodloužit.

Proces ladění chyb je webové aplikaci mnohem snazší. Vzhledem k tomu, že všechny procesy probíhají na serveru, jsou faktory ovlivňující chování systému redukovány na minimum. Zároveň je možné aplikaci nastavit tak, aby generovala logy, jejichž pomocí se dají případné chyby nalézt mnohem snadněji.

Toto kritérium tedy jasně mluví ve prospěch webové aplikace. Nejen, že její použití výrazně ulehčí práci vývojářům, ale zároveň nebude klást takové nároky na uživatele.

2.1.5 Mobilní zařízení

Se zvyšující se cenovou dostupností a využívaností mobilních zařízeních v procesech společností je jen otázkou času, kdy se vedení agentury Haul rozhodne rozšířit technologické zázemí svých zaměstnanců o některá z těchto zařízení.

Mobilní zařízení by mohla umožnit zaměstnancům areálů přístup k systému přímo v prostoru skladu, což může být velmi praktické. Z tohoto důvodu je dobré zamyslet se nad tím, jak složité úpravy budou potřeba, aby byl vyvíjený systém dostupný i na mobilních zařízeních.

Jak již sám název napovídá, desktopová aplikace bude vyžadovat v lepším případě speciální prezentační vrstvu, v tom horším bude potřeba vytvořit dedikovanou aplikaci pro mobilní zařízení. V případě dvou souběžných aplikací bude zároveň velmi složité udržovat konzistentní jejich části starající se o logickou stránku procesů.

Oproti tomu pro přístup k webové aplikaci stačí mobilnímu zařízení pouze internetový prohlížeč a připojení k internetu – věci, jež dnes snad už žádné mobilní zařízení nepostrádá. Při vývoji webové aplikace je také běžně relativně snadné přizpůsobit uživatelské rozhraní tak, aby vypadalo dobře na různě velkých obrazovkách.

Z pohledu možnosti rozšíření na mobilní zařízení tedy nemůže desktopová aplikace webové nikterak konkurovat.

2.1.6 Preference uživatelů

Při takto důležitém rozhodnutí by se nemělo zapomínat na uživatele. Rozhodl jsem se proto prozkoumat, který z přístupů je u uživatelů oblíbenější.

Nejprve jsem se pokusil přijít na to, zda existuje na tento problém obecně převládající názor. Uživatelská rozhraní těchto dvou typů aplikací jsou však natolik podobná, že většina uživatelů rozdíl mezi nimi nepozná či nijak výrazněji nevnímá.

Následně jsem se tedy rozhodl zjistit, zda neexistuje konsenzus alespoň na vývojářské rovině. Po přečtení několika vyhrocených diskuzí na internetových fórech jsem však zjistil, že každý přístup má svá pro a proti a diskutující členové se shodnou pouze na tom, že se neshodnou.

Obrátil jsem se proto přímo na Mgr. Janu Dvořákovou s dotazem, jestli v tomto ohledu mají zaměstnanci agentury Haul nějaké preference. Odpovědí bylo ujištění, že zaměstnanci takto malé nuance ani nepostřehnou.

V rámci tohoto kritéria se mi tedy nepodařilo nalézt ani jednu rovinu, na níž by některý z přístupů převažoval.

2.1.7 Osobní zkušenosti

V poslední řadě také stojí za to zvážit mé vlastní možnosti. Vývoj systému v technologiích, s nimiž se setkávám denně, bude vykazovat jistě vyšší úroveň a menší náchylnost k chybám, než v technologiích, jež ovládám málo či vůbec.

Většinu svého studia na vysoké škole jsem se věnoval programováním v jazycích, jež se hodí více ke tvorbě desktopových aplikací. Zároveň již druhým rokem v rámci své práce pomáhám vyvíjet desktopovou aplikaci zaměřující se na návrh a analýzu budov.

Naproti tomu jsem svými studii prošel webovými téměř nepolíben. Vyzkoušel jsem si sice nějaké základy, avšak pokud bych se rozhodl vytvořit webovou aplikaci, znamenalo by to skoro jistě, že bych se musel zvolenou technologii naučit takříkajíc „od píky“.

V tomto ohledu bych tedy velmi výrazně preferoval desktopovou aplikaci, neboť bych jejím vývojem strávil v porovnání podstatně méně času než v případě webové aplikace.

2.1.8 Shrnutí

Po zvážení všech vymezených kritérií je nyní čas shrnout výhody a nevýhody jednotlivých přístupů a učinit konečné rozhodnutí, který z nich upřednostním.

V kontextu preference uživatelů, synchronizace dat či dodatečných nákladů na provoz, nezískal ani jeden z přístupů nad tím druhým žádnou větší výhodu.

Přístup využívající webovou aplikaci získal navrch v nasazování a ladění. Zároveň umožňuje velmi snadné rozšíření o podporu mobilních zařízení, pokud to bude v budoucnosti potřeba.

V porovnání s tímto seznamem výhod může zvolení desktopové aplikace nabídnout pouze větší komfort pro mě coby autora, a to navíc pouze při prvotním vývoji.

Z těchto důvodů jsem se rozhodl systém vyvinout jako webovou aplikaci.

2.2 Volba technologie

Nyní, když jsem rozhodl, jaký typ bude vyvíjený systém mít, je potřeba zvolit technologie, které pro jeho tvorbu použiji.

Vzhledem k mému nedostatku zkušeností s vývojem webových aplikací pro mě tento úkol představoval velkou výzvu. Nejenom, že existuje nepřeberné množství programovacích jazyků, v nichž je možné webové aplikace vytvářet,

ale k většině těchto jazyků také existuje adekvátní množství frameworků, jež zjednodušují typické aspekty spojené s vytvářením webových aplikací jako je například přístup k databázi, zabezpečení či práce s emaily.

Jakmile mi došlo, jak velký je prostor pro výběr, uvědomil jsem si rychle, že pouze zkoušením jednotlivých možností a výběrem ideální technologie bych mohl strávit celý semestr. Tolik času mi bohužel přáno nebylo a musel jsem vymyslet rozumný způsob, jak vybrat technologii, abych svého výběru během následujících měsíců nelitoval.

Obrátil jsem se tedy na vedoucího mé diplomové práce, Ing. Tomáše Nováčka, s prosbou, zdali by mi nepomohl s touto těžkou volbou. Po krátkém uvažování mi doporučil framework Nette [15].

Framework Nette vznikl v České republice v roce 2008 coby rodina samostatných knihoven pro práci s jazykem PHP. Od té doby na něm probíhá aktivní vývoj, který je v současné době podporován několika českými společnostmi.

Nette je pro programátory lákavé především svojí přívětivostí, kterou přináší podrobná dokumentace společně s mnoha tutoriály. Zároveň za tímto frameworkem stojí velmi aktivní komunita vývojářů, která přispívá jeho rozvoji vývojem doplňků či diskuzemi na oficiálním fóru.

Není proto divu, že se v roce 2015 jednalo o nejoblíbenější PHP framework v České republice a třetí nejoblíbenější PHP framework na světě [16]. Od té doby jeho popularita lehce upadla, avšak stále se jedná o jeden z nejpoužívanějších PHP frameworků v České republice.

Nejpodstatnější výhodu však vidím ve skutečnosti, že se několik mých přátel aktivně podílelo na vývoji webových aplikací pomocí frameworku Nette. Společně s vývojářskou komunitou okolo tohoto frameworku tedy tvoří záchrannou síť, o níž se budu moci zachytit, pakliže se v moři nových poznatků spojených s vývojem webových aplikací ztratím.

Cítě se pln elánu a optimismu jsem tedy zvolil framework Nette coby prostředek pro tvorbu vytvářeného systému a pustil se do studování tutoriálů a dokumentace.

2.3 Návrh uživatelského rozhraní

Již při formulaci požadavků si zadavatelé vytvářejí v hlavě obrázek o tom, jak by měl vytvářený systém vypadat. Většině z nich však na mysli nevyvstanou zapeklitá integritní omezení formující se v databázi či důmyslná rozhraní, mezi nimiž se elegantně serializují komplikované datové struktury. Myslí zadavatelů naopak probleskují tlačítka, tabulky a obrázky, které jim nabídne uživatelské rozhraní jejich vysněného systému.

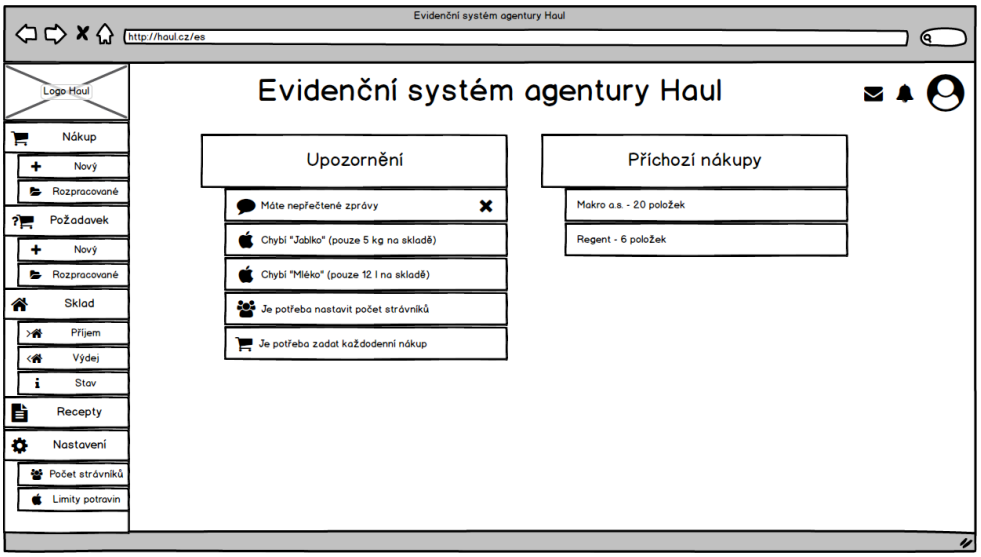
Čím honosnější jsou jejich představy, tím více bývají zpravidla zklamáni, když se jejich vize nesetkají s vizemi týmu vývojářů. V tom horším případě

dojde k tomuto střetu až při předávání produktu, kdy již není čas s tím nic udělat.

Aby se předešlo podobným zklamáním, začleňuje se do procesu vývoje systémů návrh uživatelského rozhraní, v rámci něhož se tým vývojářů snaží co nejtěsněji přiblížit k vizi zadavatele. Vzhledem k tomu, že především první verze tohoto návrhu mají velkou šanci být vyměněny za jiné, vytváří se návrh většinou černobíle bez složitých grafických prvků.

Skutečnost, že hlavním požadavkem agentury Haul na vyvíjený systém byla jednoduchost používání, jež přímo souvisí s kvalitou uživatelského rozhraní, vedla k mému rozhodnutí nechat nic náhodě a věnovat jeho návrhu nemalé množství času.

Pro tvorbu *Lo-Fi* prototypů jsem se rozhodl zvolit nástroj Balsamiq [17]. Jedná se o známý a používaný software pro vytváření těchto prototypů. Balsamiq je relativně jednoduchý na naučení a používání, takže jsem neměl problém dát dohromady první návrh uživatelského rozhraní v řádu několika hodin. Vzhled této verze Vám může přiblížit návrh domovské stránky systému na obrázku 2.1.

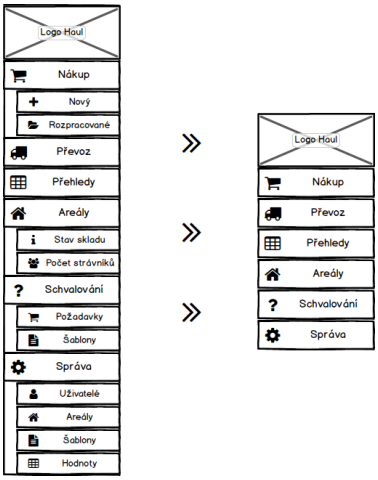


Obrázek 2.1: Domovská stránka systému – pohled zaměstnance areálu

Mým hlavním záměrem bylo vyhnout se přílišnému odvádění pozornosti uživatele nabízením možností a představováním údajů, jež nejsou relevantní pro činnost (či její část), které se uživatel právě věnuje. Tento postoj jsem zaujal především proto, že většina analyzovaných systémů měla s tímto aspektem svých rozhraní problém. Podobný neduh vedl také k nepochopení a následnému odstavení Magdaleny LK, což je osud, kterého bych chtěl vyvíjený systém ušetřit.

Příkladem mého úsilí v tomto ohledu může být například menu, které jsem strukturoval tak, aby se mysl uživatele v jakýkoliv okamžik nemusela soustředit na více než pět možností.

V rámci návrhu jsou jednotlivé části menu bohužel neustále rozbalené, neboť docílení efektu zavírání a otvírání by vedlo k mnohonásobnému navýšení počtu navrhovaných obrazovek. Způsob, jakým bude zabalování (resp. rozbalování) položek menu probíhat, je znázorněn na obrázku 2.2.



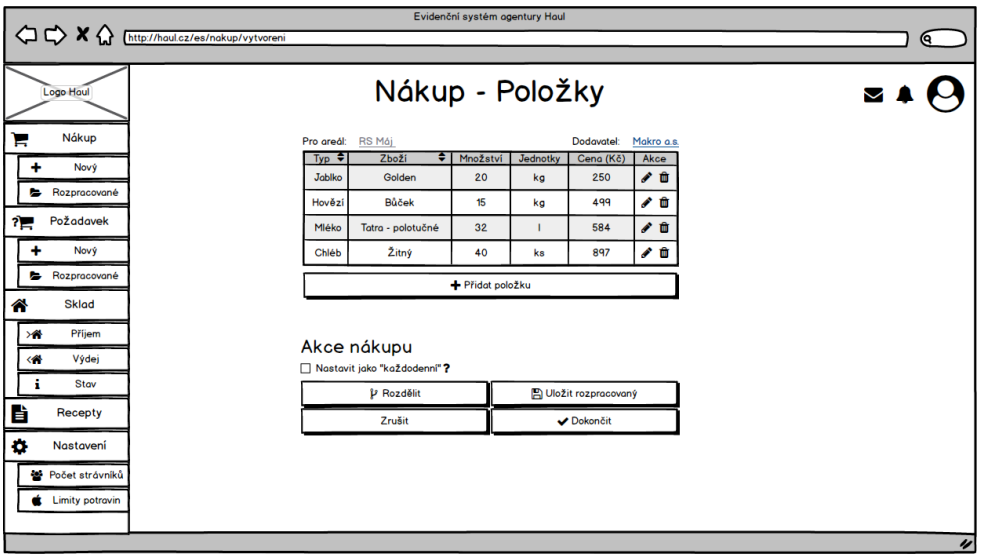
Obrázek 2.2: Způsob zabalování položek menu

Celkový návrh uživatelského rozhraní byl primárně řízen případy užití. Jejich struktura mi pomohla rozdělit procesy do kategorií podle typu uživatele, jemuž jsou přístupné, a následně podle částí systému, jichž se bezprostředně týkají.

Zároveň jsem mohl využít jejich scénáře (resp. kroky těchto scénářů), abych rozdělil zachycované procesy do částí, které spolu souvisí méně, a mohou být tedy rozděleny na více obrazovek. Od tohoto přístupu si slibuji maximalizaci přehlednosti systému při zachování srovnatelné úrovně použitelnosti.

Jako příklad výsledků mé snahy jsem se rozhodl zde uvést ukázky obrazovek *Vytváření nákupu* z pohledu zaměstnance areálu (vizte obrázek 2.3) a *Vytváření přehledu* (obrázek 2.4). I přesto, že se jedná o nejsložitější obrazovky v celém systému, neobsahují více než deset (resp. patnáct) kontrolních prvků (pro porovnání: ukázková obrazovka ze systému Magdalena LK jich má více než padesát).

Vzhledem k velké odlišnosti dostupných funkcí pro jednotlivé typy uživatelů jsem návrh rozdělil do dvou částí. Jedna představuje systém z pohledu zaměstnance areálu, druhá obsahuje funkce dostupné zaměstnanci obchodního oddělení. V rámci druhé části jsou zároveň zahrnuty funkce dostupné uživatelům se speciálními právy.



Obrázek 2.3: Vytváření nákupu – pohled zaměstnance areálu

Obě části výsledné verze návrhu jsem vzhledem k jejich velikosti (dohromady zhruba sto obrazovek) neumístil do příloh, nýbrž budou k nalezení na médiu přiloženém k mé práci.

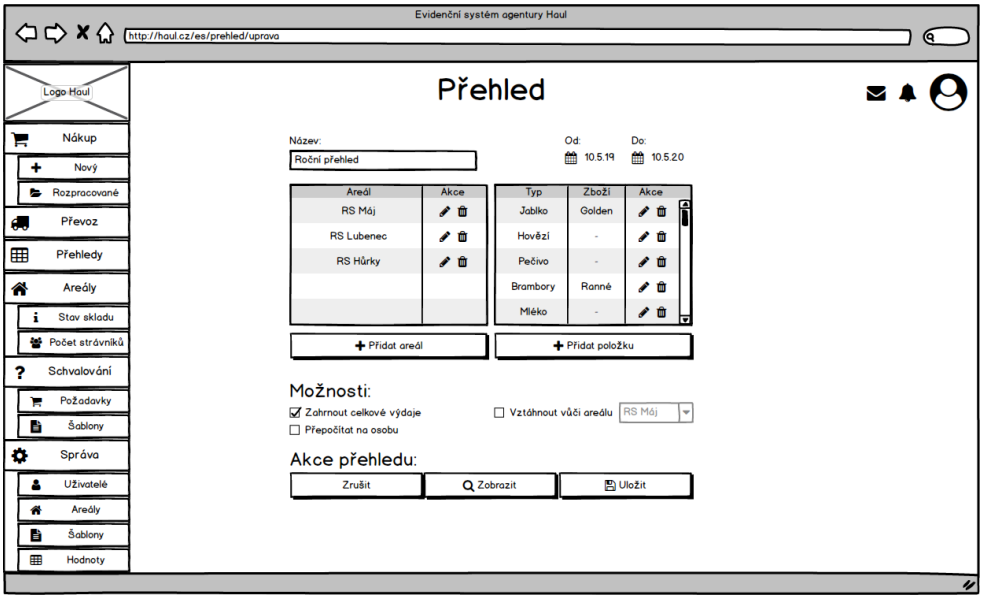
2.4 Databázový model

Nedílnou součástí drtivé většiny systémů je persistence dat, s nimiž tyto systémy pracují. Způsob, jakým jsou data ukládána a načítána, hraje nezanedbatelnou roli v rychlosti odezvy systému. Z tohoto důvodu jsem se rozhodl zde návrh databázové modelu podrobněji rozepsat.

Kompletní model obsahuje dvacet čtyři tabulek a přes třicet vazeb mezi nimi. Vzhledem k jeho velikosti jsem kompletní schéma umístil pouze do přílohy C, kde jsem ho i tak musel kvůli čitelnosti rozdělit na dvě části.

V rámci této sekce jsem databázový model rozdělil do osmi skupin, přičemž každá z nich se zaměřuje na specifickou část vyvíjeného systému. Každé části věnuji oddíl, v němž zmíním vyvstavší problémy a způsob, jakým jsem se rozhodl je řešit.

Zároveň bude v každém oddílu schéma dané části databázového modelu. V rámci těchto schémat jsou pro zachování kontextu zobrazeny i tabulky z jiných částí. Tyto tabulky mají záměrně skryty sloupce a popis integritních omezení, aby zbytečně neodváděly pozornost od klíčových aspektů zobrazované části.

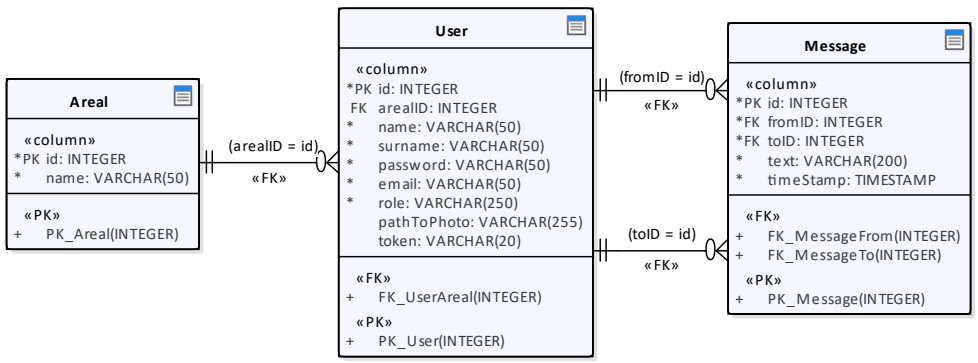


Obrázek 2.4: Vytváření přehledu – pohled zaměstnance obchodního oddělení

2.4.1 Uživatelé

První část databázového modelu se zabývá dvěma základními typy entit – areály a uživateli. Do této části jsem se rozhodl zařadit i zprávy, neboť jsou přímo a výhradně vázané na uživatele.

Schéma této části se nachází na obrázku 2.5. Z něho jsou patrné základní atributy areálu (*název*), zprávy (*text*, *datum odeslání*, *ID odesílatele* a *ID příjemce*) a uživatele (*jméno*, *příjmení*, *emailová adresa*, *role* a *hash hesla*).



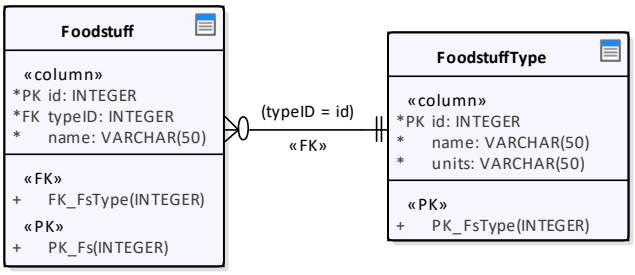
Obrázek 2.5: Databázový model – část *Uživatelé*

Kromě zmíněných atributů obsahuje tabulka pro uživatele také *název souboru*, v němž je uložena profilová fotografie uživatele, a případný *autorizační token*, který se používá v rámci procesu obnovy uživatelského hesla.

2.4.2 Potraviny

Posledním základním typem entit ve vyvíjeném systému jsou potraviny. U potravin je potřeba rozlišovat *typ* od *konkrétní* potraviny. Způsobu reprezentace těchto entit se věnují následující odstavce.

Vztah mezi *konkrétní* potravinou (např. „Jablko Golden“) a jejím *typem* (např. „Jablko“) je hierarchický. Tuto skutečnost odráží integritní omezení mezi odpovídajícími tabulkami (vizte obrázek 2.6).

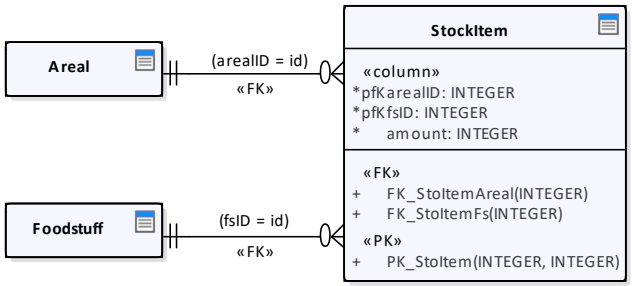


Obrázek 2.6: Databázový model – část *Potraviny*

Z diskuze s Mgr. Janou Dvořákovou vyplynulo, že všechny potraviny jednoho typu musí být evidovány ve stejných jednotkách. Z tohoto důvodu je odpovídající sloupec přiřazen tabulce uchovávající *typ* potravin.

2.4.3 Sklad

Dalším z aspektů, jež je potřeba uchovávat, je stav potravin ve skladech areálů. Odpovídající část schématu představuje obrázek 2.7.



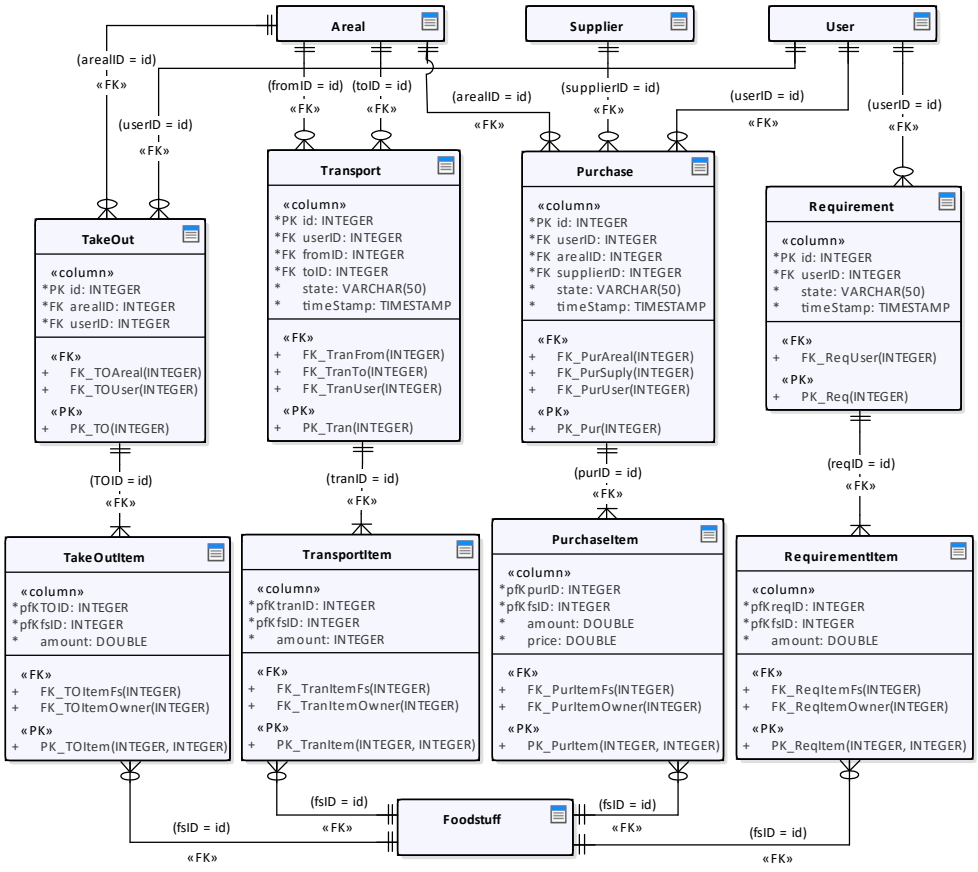
Obrázek 2.7: Databázový model – část *Sklad*

Vzhledem k tomu, že je žádoucí, aby existoval vždy pouze jeden pár *sklad-potravina*, dá se tato skutečnost zachytit *M:N dekompozicí*. Tato vazba je zároveň obohacena o *množství* daných potravin na skladě. Tento počet se bude při přidávání a odebírání aktualizovat.

2.4.4 Přesun zboží

Klíčovou částí vyvíjeného systému je přesun zboží, ať už se jedná o *nákupy*, *požadavky* na ně, *převoz zboží* nebo samotné *vydávání*. Způsobem uložení informací spojených s těmito procesy se zabývá tento oddíl.

Schéma této části schématu databázového modelu zachycuje obrázek 2.8. Toto schéma obsahuje jednu tabulku, jež je zkrácena, ale nebyla doposud popsána. Touto tabulkou je tabulka *Supplier*, která obsahuje pouze *název* dodavatele. Její schéma je možné nalézt v příloze C.



Obrázek 2.8: Databázový model – část *Přesun zboží*

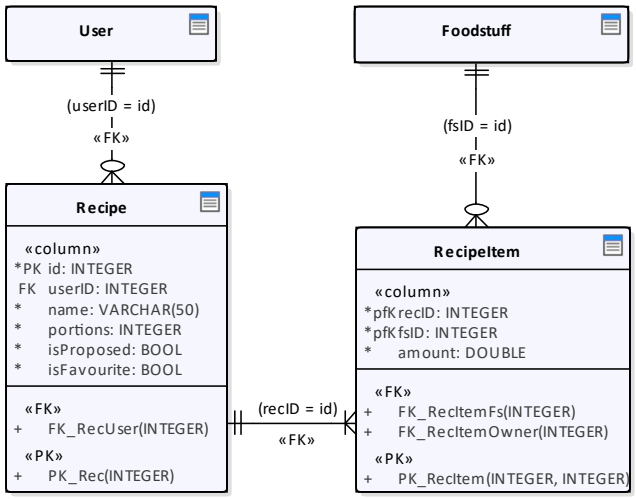
Data o každém ze zmíněných čtyřech procesů jsou ukládána podobným

způsobem. Nejprve jsou uloženy informace o samotném procesu (např. tabulka *Purchase*). Tabulky zachycující tyto informace obsahují *ID* entit, s nimiž souvisí, a kromě *vydávání* také *stav*, v němž se daný proces nachází, a *čas* vytvoření.

Zmíněné tabulky jsou pak vždy spojeny *M:N dekompozicí* s *konkrétní potravinou*. Tyto vazby jsou rozšířeny o *množství* potravin a v případě *nákupu* i o *celkovou cenu* za danou položku.

2.4.5 Recepty

Elegantním způsobem, jak usnadnit uživatelům vydávání potravin ze skladu, jsou recepty. Těm je společně se šablonami a procesy s nimi spojenými dedikován tento oddíl a část schématu na obrázku 2.9.



Obrázek 2.9: Databázový model – část *Recepty*

Ukládání receptů je založeno na stejném principu jako ukládání procesů přesunu zboží. U receptů je však potřeba uchovávat také jejich jméno a počet porcí, vůči kterým jsou jejich údaje vztažené.

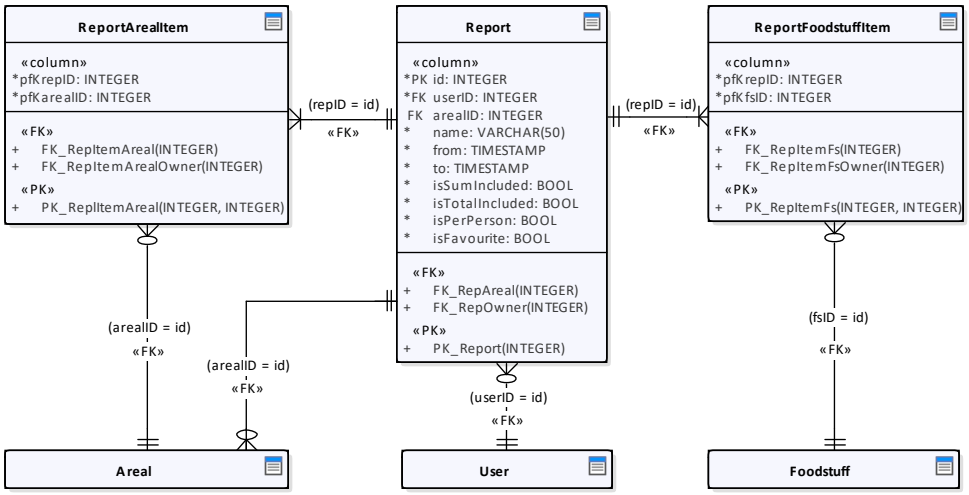
Pozorný čtenář si jistě všiml, že chybí tabulka pro uchování šablon receptů. Vzhledem k téměř totožné struktuře obou entit jsem se rozhodl je ukládat v rámci jedné tabulky. Klíčovým rozdílem mezi nimi je vztah k uživateli. Zatímco každý recept musí náležet některému z uživatelů, šablony jsou sdíleny mezi všemi, a budou mít tedy ve sloupci *userID* hodnotu *NULL*.

Zaměstnanci areálů mají možnost své recepty navrhovat ke sdílení. V případě, že se takto rozhodnou, bude mít daný recept nastavený příznak *isProposed*. Tento příznak se u šablon zanedbává.

Posledním sloupcem této tabulky je sloupec *isFavourite*, který obsahuje hodnotu indikující, zda má uživatel tento recept označený jako „oblíbený“.

2.4.6 Přehledy

Přehledy představují hlavním způsob, jakým mohou zaměstnanci obchodního oddělení nahlížet na stav a dění v systému. Tento oddíl a schéma na obrázku 2.10 jsou jim věnovány.



Obrázek 2.10: Databázový model – část *Přehledy*

Centrální tabulkou této části je *Report*, která obsahuje základní údaje přehledu jako *jméno*, rozmezí dat *od-do* a *ID uživatele*, jež přehled vytvořil.

Přehledy mají strukturu tabulky se dvěma dimenzemi. Jednou jsou areály, jež mají být porovnávány, tou druhou potraviny, s nimiž spojené výdaje mají být porovnány. Každá z těchto dimenzí je realizována *M:N dekompozicí* daného vztahu.

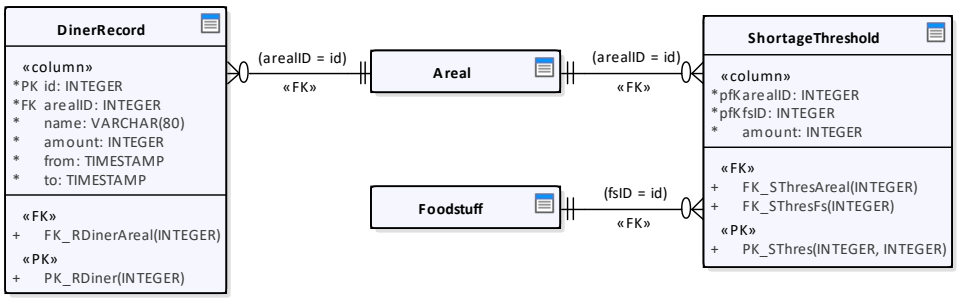
V rámci přehledu mají zaměstnanci obchodního oddělení možnost nastavit čtyři dodatečné parametry. Přítomnost *celkových výdajů*, *součtu položek* a přepočítání *na osobu* jsou realizovány odpovídajícími příznaky.

Areál, k němuž mají být vztaženy hodnoty, je vybrán přiřazením jeho *ID*. Pokud hodnoty nemají být vztaženy ke konkrétnímu areálu, bude v odpovídajícím sloupci hodnota *NULL*.

Přehledy mají, podobně jako recepty, příznak indikující, jestli patří mezi oblíbené daného uživatele.

2.4.7 Nastavení areálu

Každý z areálů podléhá několika nastavením. Jedná se o nastavení *počtu strávníků* a *minimálního množství potravin*. Realizace persistence těchto nastavení je předmětem tohoto oddílu a diagramu na obrázku 2.11.



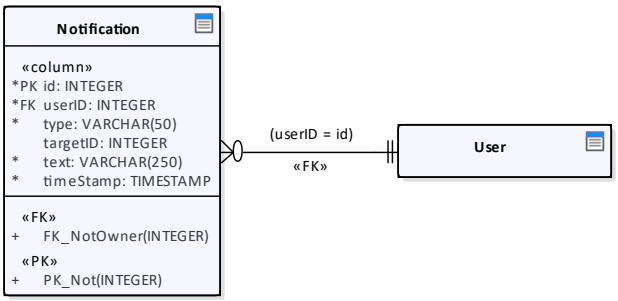
Obrázek 2.11: Databázový model – část *Nastavení areálu*

Tabulka uchovávající informace o nastavení *počtu strávníků* pro jednotlivé areály obsahuje sloupce zachycující *ID areálu*, jehož se nastavení týká, jméno akce, v rámci níž je počet strávníků evidován, nastavený *počet strávníků* a rozmezí *od* – *do*, v němž daná informace platí.

Vzhledem k charakteru vztahu *M:N* areálu a potravin, jejichž *minimální množství* je možné nastavit, jsou informace o něm ukládány v tabulce realizující odpovídající dekompozici.

2.4.8 Notifikace

Posledním z aspektů vyvíjeného systému, jež vyžaduje ukládání, jsou notifikace. V rámci tohoto oddílu je nastíněn způsob, jakým je tento problém řešen. Schéma návrhu tabulky pro notifikace se nachází na obrázku 2.12.



Obrázek 2.12: Databázový model – část *Notifikace*

Návrh persistence notifikací představuje (minimálně pro mě) nemalou výzvu, neboť jejich typů bývá v systémech velký počet. Zároveň se jednotlivé typy notifikací většinou vztahují k různým částem systému.

Klasické modelování této hierarchie by vedlo na velmi řídkou tabulku s velkým počtem sloupců či na velký počet malých tabulek.

Z tohoto důvodu jsem se rozhodl ukládat všechny typy notifikací v rámci jedné tabulky. Tato tabulka obsahuje sloupce, jež pokrývají společné vlastnosti všech typů notifikací. Těmi jsou *ID uživatele*, jemuž notifikace náleží, *datum vytvoření* a *text* dané notifikace.

K rozpoznávání jednotlivých typů slouží sloupec obsahující *typ* notifikace. Formulaci a použití notifikace upřesní vrstva systému obsahující výpočetní logiku za pomocí tohoto typu a *ID předmětu notifikace*, které je taktéž uloženo.

Kapitola 3

Realizace

Tato kapitola popisuje průběh a výsledek mých snah o převedení nápadů a představ z předchozích dvou kapitol ve skutečnost.

V následujících sekcích nejprve představím způsob, jakým jsou organizovány části vyvinutého systému, a vymezím, které z požadavků systém pokrývá.

Dále váženého čtenáře seznámím s některými z komplikovanějších problémů, jež jsem během vývoje musel řešit, a vylepšeními, jimiž jsem se snažil zvýšit rozšiřitelnost, udržitelnost a celkově kvalitu vyvinutého systému.

Na tyto sekce navážu výčtem míst a možností, jež skýtají prostor pro budoucí vylepšení. Závěr kapitoly je věnován nasazení vyvinutého systému.

V rámci realizace byla také vytvořena uživatelská příručka, kterou je možné nalézt na médiu přiloženém k této práci.

3.1 Struktura aplikace

Framework Nette [15], jež jsem zvolil jako základ pro vývoj systému, je úzce spojen s návrhovým vzorem *Model-View-Controller* [18]. Z tohoto důvodu jsem se rozhodl zbytečně se od tohoto vzoru při vytváření systému neodklánět.

Při čtení následujících odstavců může čtenáře udivit, že pojmenování tříd *Presenters* neodpovídá pojmenování části návrhového vzoru *Controller*. Toto pojmenování vychází ze starších verzí frameworku Nette [19].

Systém je rozdělen na tři základní části:

Model má na starost ukládání a poskytování dat aplikace. V rámci systému je představován především třídami, jež se nacházejí v adresářích *Model* a *Entities*.

View se primárně stará o způsob, jakým jsou uživatelům vykreslena data. Ve vyvinutém systému je tato část zastoupena soubory typu *.latte*.

Controller vybírá potřebná data z modelu, připravuje je pro zobrazení a následně je předá view, čímž propojuje zmíněné dvě části. Třídy zastupující tuto část se nacházejí v adresářích *Forms* a *Presenters* a jsou typu *.php*.

Zodpovědnost tříd v části *Controller* je dále dělena mezi „presentery“ (třídy z adresáře *Presenters*) a „formuláře“ (třídy z adresáře *Forms*).

„Presentery“ se starají především o zpracování a validaci příchozích požadavků, kontrolu přístupových práv uživatelů, případná přesměrování a delegaci vykreslení na správný *formulář*. Zodpovědností *formulářů* je příprava dat pro zobrazení uživatelům, kontrola dat zadaných uživateli a případné upravení dat aplikace skrze třídy části *model*.

Kromě těchto tří základních částí obsahuje systém také:

Komponenty – zobecněné části uživatelského rozhraní, jež se používají na více místech systému.

Router – pomocnou třídu, jež převádí *url* požadavků na akce *presenterů*.

Služby – pomocné třídy, které se starají například o odesílání emailů, porovnávání hodnot, kontrolu dat od uživatelů či správu jejich relací.

Správu databáze – třídy starající se o vytvoření a prvotní naplnění databáze daty.

Kaskádové styly a skripty – soubory typu *.css* a *.js*, jež slouží ke správnému vykreslování a chování webových stránek.

Skripty pro nasazení – soubory, jejichž pomocí se dá jednoduše vyřešit konfigurace systému.

3.2 Implementované požadavky

V této sekci projdu funkce a vlastnosti vytvořeného systému a propojím je s funkčními a nefunkčními požadavky ze sekce 1.5, které svojí přítomností splňují. Odkazy na odpovídající požadavky jsou uvedeny v závorce za související funkcí (resp. vlastností).

3.2.1 Klíčové požadavky

Finální verze vyvinutého systému umožňuje vytváření požadavků (F2) a všech tří druhů nákupů (F1). Zaměstnanci areálů mohou přijímat příchozí nákupy (F3) a vydávat zboží ze skladu (F6). Zaměstnanci obchodního oddělení mají v systému přístup k vytváření a sledování převozů (F7).

Systém také poskytuje zaměstnancům areálu možnost vytváření a upravování receptů (F4) a zaměstnancům obchodního oddělení možnost práce s přehledy (F8).

Uživatelé si dále mohou nechat zobrazit množství potravin ve skladu (F9) a záznamy o počtu strávníků (F10). Každý uživatel má možnost spravovat svůj uživatelský profil (F12).

Uživatelům se speciálními právy je umožněno spravovat záznamy o areálech, uživatelích a hodnotách v rámci systému (F11).

Práce se systémem je umožněna pouze uživatelům, jež se autorizují přihlašovacím jménem a heslem (N1).

Pro provoz systému nebyly použity žádné technologie s placenou licencí (N3). Rozložení jednotlivých stránek vytvořeného systému je přizpůsobeno rozlišení stolních počítačů i mobilních zařízení a jeho nároky na hardware nepřekračují specifikované hodnoty (N2).

Tímto by měly být pokryty všechny klíčové funkční i nefunkční požadavky.

3.2.2 Doplňkové požadavky

Vytvořený systém dále podporuje práci se šablonami receptů (F5), možnost označovat recepty a přehledy jako „oblíbené“ (F13). Uživatelé si také mohou v rámci systému posílat zprávy (F14).

Kromě základní práce s přehledy a nákupy nabízí systém také možnost tyto dva typy záznamů ukládat pro pozdější dokončení a rozdělovat (doplňkové části požadavků F1 a F2).

Při vytváření přehledů mají zaměstnanci obchodního oddělení kromě porovnání celkových výdajů také možnost porovnávat náklady pro konkrétní zboží a nechat si zobrazit součet těchto dílčích porovnání. Zároveň si mohou vytvořené přehledy uložit (doplňkové části požadavku F8).

3.2.3 Propojení se systémem registrací

Během procesu vývoje systémů nadnesl vedoucí mé práce, Ing. Tomáš Nováček, návrh, zda by bylo možné propojit systém vyvíjený v rámci této práce se systémem, jež bude spravovat registrace účastníků na akce pořádané agenturou Haul. Tento systém je vyvíjen v rámci bakalářské práce Anny Vitmanové [13].

Propojení systémů by mělo spočívat v aktualizaci záznamů o počtu účastníků v mnou vyvíjeném systému na základě dat ze systému spravujícího registrace. Přenos dat by probíhal pomocí přenosu souboru formátu *.json*. Periodicita obnovy těchto dat nebyla doposud specifikována a bude předmětem debat s Mgr. Janou Dvořákovou.

Přenášený soubor bude obsahovat seznam záznamů o akcích pořádaných agenturou Haul. Každý záznam bude obsahovat datum začátku a konce dané akce, jméno areálu, v němž bude akce probíhat, a počet registrovaných účastníků. Tento formát byl vytvořen po dohodě s Annou Vitmanovou.

Přenášená data bohužel kompletně nepokrývají strukturu záznamu o počtu účastníků v rámci mnou vyvíjeného systému. V něm je potřeba u jednotlivých

akcí specifikovat kromě přenášených informací také název dané akce. Tato informace nemůže být přenesena mezi systémy, neboť systém vyvíjený Annou Vitmanovou řeší pouze tábory, u nichž by uchovávání jmen bylo podle provedené analýzy zbytečné.

Z tohoto důvodu bude všem akcím přeneseným tímto způsobem přiřazeno specifické jméno. Konkrétní podoba tohoto jména bude předmětem diskuzí s Mgr. Janou Dvořákovou.

V rámci přenosu budou do mnou vyvíjeného systému doplněny chybějící záznamy a aktualizovány záznamy stávající. Záznamy jsou pro účely kontroly jejich přítomnosti v systému považovány za totožné, pakliže se shoduje jejich název a data začátku a konce.

3.3 Řešené problémy

Při psaní závěrečné práce se od jejího autora očekává, že se během procesu tvorby vypořádá s nejedním netriviálním problémem. Zmíněné netriviální problémy se ovšem dělí na základní dva druhy.

Prvním jsou problémy spojené s problematikou práce, jež nutí autora použít vědomosti a zkušenosti získané za léta strávená studiem. Řešení tohoto druhu problémů zabere mnohdy dny až týdny, avšak po jejich zdolání zůstává v řešiteli hřejivý pocit uspokojení.

Druhým typem jsou problémy související s použitými technologiemi, konflikty mezi operačním systémem a firmwarem různých zařízení či chybou uvnitř použité knihovny. Tyto problémy bývají zřídkakdy konstruktivní, jejich řešení mnohdy nebere konce a po jejich vyřešení zbývá většinou pouze trpký pocit prázdna a otázka, proč tento problém nemohl vyřešit někdo jiný dříve.

V následujících odstavcích bych se rád podělil o způsob, jakým jsem řešil některé ze složitějších problémů, na něž jsem při vytváření systému narazil. Jedná se především o problémy druhého typu (dle výše zmíněné kategorizace), neboť většině problémů prvního typu jsem se (nejspíše) vyhnul při návrhu systému, nebo jsem jejich řešení zařadil do sekce 3.4.

3.3.1 Výběr hodnot uvnitř tabulek

Napříč vytvořeným systémem se vyskytuje mnoho míst, na nichž je potřeba vybírat jednu z velkého množství existujících hodnot. Jedná se především o výběr areálu, dodavatele, zboží nebo jeho typu.

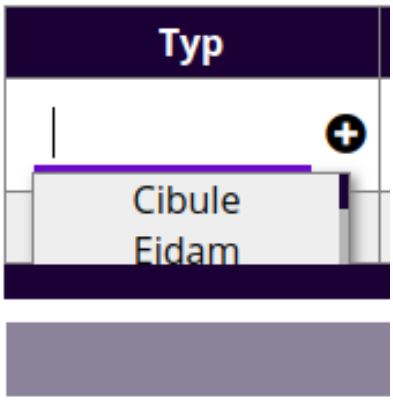
Pro tuto činnost jsem se rozhodl použít textové pole, které je spojené s nabídkou položek, jež jsou podobné zadávanému výrazu. Zároveň je toto pole po zvolení jedné z možností nahrazeno tlačítkem, které obsahuje vybranou možnost. Po stisknutí tohoto tlačítka je tlačítko nahrazeno původním textovým polem.

Tento způsob volby jedné z existujících hodnot je použit v klasických formulářích a během úpravy řádků tabulek. Všechno fungovalo v pořádku do mo-

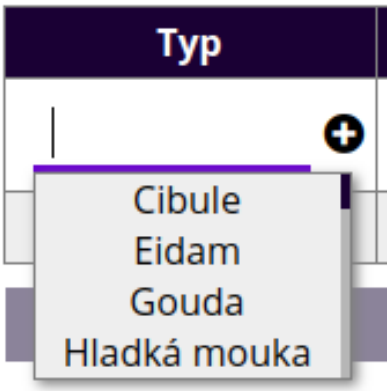
mentu, kdy bylo potřeba zařídit možnost „scrollování“ tabulek, které obsahovaly tento prvek uživatelského rozhraní.

Aby bylo možné část webové stránky „scrollovat“, je nutné, aby byla obsažena v prvku, který má pevné rozměry. Jakmile je část webové stránky větší než rozměry prvku, který ji obaluje, dojde k automatickému zobrazení posuvníku na straně a skrytí všech přesahujících částí.

Problém nastává při zobrazování nabídky podobných položek. Ta totiž musí být zobrazena výše než všechny ostatní položky. Jakmile je ovšem uvnitř tabulky, která je omezená rozměry svého rodiče (kvůli „scrollování“), vztahují se na ni pravidla pro skrývání při přesahu rozměrů rodiče tabulky. Příklad špatného a správného chování je zachycen na obrázcích 3.1 a 3.2.



Obrázek 3.1: Špatné chování prvku pro výběr položek



Obrázek 3.2: Správné chování prvku pro výběr položek

Tento problém je možné řešit nastavením pozice seznamu podobných položek na absolutní a dopočítání jeho pozice podle pozice textového pole. Tím bohužel ztrácíme vazbu na původní textové pole, což se projeví nemožností přizpůsobit šířku seznamu podobných položek (proměnlivé) šířce textového pole.

Přizpůsobení šířky bylo vyřešeno nastavením stejné (pevné) šířky pro seznam podobných položek i textové pole. Toto řešení je sice funkční, ale zároveň suboptimální, neboť pevná šířka těchto elementů komplikuje řešení následujícího problému.

3.3.2 Šířka tabulek

Vzhledem k požadavku na kompatibilitu s mobilními zařízeními bylo potřeba vytvořit responzivní rozložení obrazovek. To ovšem vyvolalo hned několik problémů.

Prvním z nich je zobrazování tabulek, které jsou širší než dostupná šířka obrazovky. Tato situace běžně nastává u obsáhlejších tabulek na mobilních zařízeních.

Existuje více přístupů, jak tento problém řešit. Jedná se o umožnění horizontálního „scrollování“ tabulky, přeskládání tabulky takovým způsobem, aby každý záznam představoval jednu menší tabulku či skrytí některých sloupců.

Každé ze zmíněných řešení má své výhody a nevýhody. V současné době neexistuje v programátorské komunitě konsensus ohledně toho, které z řešení by se mělo použít. Pro jeho jednoduchost a kompatibilitu napříč typy rozlišení obrazovek jsem si vybral přístup umožňující horizontální „scrollování“.

Další problém představují obrazovky s příliš velkým rozlišením. Při nastavení šířky tabulky v poměru vůči šířce obrazovky utrpí na čitelnosti tabulky s malým počtem sloupců, neboť hodnoty v rámci jednoho řádku tabulky budou od sebe velmi vzdálené.

Pokud nastavíme pevnou šířku tabulky, utrpí na tom tabulky s velkým počtem sloupců nebo širokými sloupci, protože data v nich budou působit „namačkaně“.

Bylo by také možné nastavit pro každou tabulku jinou šířku podle toho, jak moc je široký její obsah. Toto by vyřešilo předchozí problémy, avšak znehodnotilo by to konzistenci celkového vzhledu aplikace.

Proměnná šířka tabulek také nejde příliš kombinovat s omezením přijatým v předchozí sekci. Jednotlivé buňky tabulky by se sice zvětšovaly, avšak prvky pro výběr by měly stále stejnou šířku, což nevypadá příliš dobře.

Toto omezení se nevztahuje pouze na prvky spojené s výběrem jedné z existujících možností, avšak platí pro všechna textová pole. Textová pole se snaží roztáhnout na poměrně velkou šířku, pokud jim šířka není nastavena pevně. Toto vede ke značné deformaci tabulek, které se z principu snaží uspokojit maximální šířku jejich obsahu.

Třetím problémem je aplikace těchto pravidel na tabulky, u nichž není předem známý počet sloupců. Jedná se především o nastavení obecné komponenty pro zobrazování tabulek. Šířka tabulky by se měla řídit podle kaskádového stylu, který má ovšem statický charakter.

Po přihlédnutí ke všem zmíněným problémům a omezením jsem se rozhodl nastavit všem tabulkám stejnou (pevnou) šířku. Její hodnota je nastavena tak, aby ani ty z největších tabulek nebyly příliš „namačkané“ a nejmenší tabulky nebyly příliš řídké.

3.3.3 Průhlednost

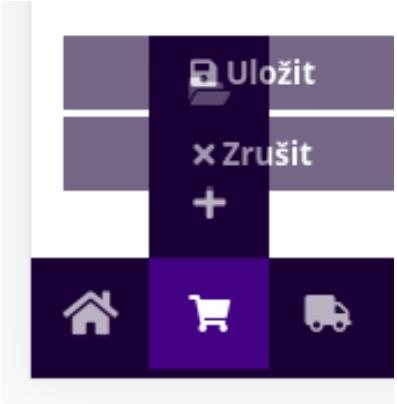
Během přizpůsobování vzhledu tlačítek jsem narazil na zajímavý problém spojený s atributem kaskádových stylů ovlivňujícím průhlednost prvků.

Zvýraznění faktu, že tlačítko není aktivní, jsem chtěl docílit zvýšením jeho průhlednosti. V rozložení pro mobilní zařízení jsem ovšem záhy zjistil,

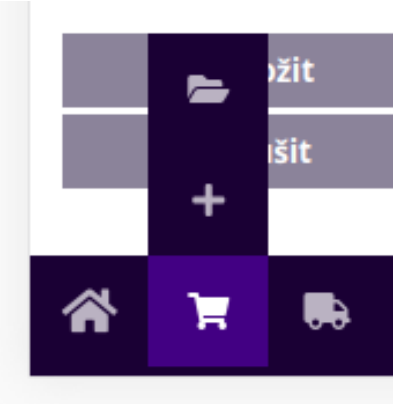
že ačkoliv po vysunutí druhé úrovně menu mají položky menu nastavený vyšší „z-index“, zůstává tlačítko „pod nimi“ stále částečně viditelné.

Při hledání vysvětlení tohoto jevu jsem narazil na odpověď na stránce *StackOverflow* [20], která zmiňuje, že ačkoliv je toto chování neintuitivní, bylo tak původně zamýšlené.

Tento problém jsem tedy vyřešil (nebo spíše obešel) manuální úpravou barev tlačítka v neaktivním stavu. Pro porovnání stavu při použití průhlednosti a bez ní vizte obrázky 3.3 a 3.4.



Obrázek 3.3: Překryv tlačítek při použití průhlednosti



Obrázek 3.4: Překryv tlačítek bez použití průhlednosti

3.3.4 Variabilní validace

Framework Nette [15] nabízí mimo jiné velmi elegantní způsob validace formulářů. Tato validace funguje spolehlivě při vyřizování a kontrole klasických jednoprůchodových formulářů.

Napříč vytvořeným systémem se ovšem nachází několik formulářů, během jejichž vyplňování je nutné přejít na jinou stránku. V takovém případě je potřeba uložit současný stav vyplňovaného formuláře, přejít na zmíněnou stránku a při návratu data obnovit či doplnit.

Problém nastává v momentě, kdy takovýto formulář obsahuje nevyplněná či špatně vyplněná pole. Základní validace by v tomto případě nedovolila uživateli provést žádnou akci s formulářem, neboť není správně vyplněný. Z tohoto důvodu by se hodilo mít možnost říci frameworku, v jakých situacích se má validace aplikovat a kdy nikoliv.

Během hledání řešení se mi bohužel nepodařilo takovýto způsob nalézt. Proto jsem si vytvořil vlastní systém pro validaci formulářů, jež je volán pouze na základě toho, které z odesílacích tlačítek uživatel zvolil.

3.3.5 Zabezpečení obnovy hesla

Systém nabízí uživatelům možnost obnovit si heslo v případě, že ho zapomenou. Tento proces ovšem musí být alespoň minimálně zabezpečený proti možnosti napadení.

Po zvolení možnosti „Zapomenuté heslo?“, vyzve systém uživatele, aby zadal emailovou adresu spojenou s jeho účtem. Jakmile uživatel zadá emailovou adresu a potvrdí žádost o obnovení hesla, zkontroluje systém, zda daná emailová adresa je spojená s některým z uživatelských účtů v systému.

Pokud se v systému uživatel s danou emailovou adresou nachází, systém vygeneruje náhodný autorizační token, přiřadí tento token danému uživateli, odešle mu email s dalšími instrukcemi a informuje ho o skutečnosti, že by mu měl přijít zmíněný email. Tato informace je zobrazena i v případě, že se v systému uživatel se zadanou emailovou adresou nenachází, aby nebylo možné podle odpovědi serveru dohledat emailové adresy uživatelů.

V emailu s instrukcemi je stručný text popisující proces obnovy hesla a odkaz, jehož součástí je vygenerovaný token. Jakmile uživatel na odkaz klikne, bude přesměrován na stránku pro obnovu hesla.

Zde systém ověří, že přiřazený token odpovídá tokenu, jež byl předán v rámci odkazu. Pokud ano, zobrazí pole pro zadání a potvrzení nového hesla. V opačném případě zobrazí uživateli stránku s chybovou hláškou.

Po vytvoření nového hesla systém toto heslo asociuje s daným uživatelem a autorizační token pro daného uživatele vymaže.

3.4 Udržitelnost a rozšiřitelnost

V případě úspěšného nasazení a začlenění vytvořeného systému do pracovních postupů zaměstnanců agentury Haul je velmi pravděpodobné, že bude systém nutno udržovat a rozšiřovat. Z tohoto důvodu bylo nasnadě klást důraz na to, aby byl kód jednoduše pochopitelný a modifikovatelný nejenom pro mě, coby autora, ale také pro programátory, kteří štafetu starání se o tento systém převezmou po mně.

V rámci snahy docílit tohoto záměru jsem přidal standardní dokumentaci kódu do všech tříd a jejich metod. Zároveň jsem se snažil zavést jmenné konvence a granularitu kódu v takové úrovni, abych zkrátil na minimum dobu potřebnou pro porozumění napsanému zdrojovému kódu.

Společně s tím jsem během procesu vytváření systému udržoval lineární a konzistentní historii v rámci verzovacího systému Git. Ze zmíněné historie lze vyčíst pořadí, v němž byly přidávány jednotlivé části systému, a částečně i návaznost jednotlivých komponent.

Jedním ze základních kamenů rozšiřitelnosti a udržitelnosti systému je dodržování principu *DRY* [21] (*Do not Repeat Yourself*). Jeho smyslem je vyčleňování společných a opakujících se částí zdrojového kódu pomocí abs-

trakce či přerozdělení odpovědností za účelem snížení duplicit napříč zdrojovým kódem.

Dodržování zmíněného principu vede k výrazně snazší udržitelnosti kódu, neboť pokud je potřeba udělat změnu v prvku systému, jež se vyskytuje na více místech, stačí většinou změnit pouze minimální počet částí zdrojového kódu.

Tento přístup má pozitivní dopad také na rozšiřitelnost kódu. Využívaje vyčleněných společných částí stačí většinou pouze specifikovat rozdíly v chování nové části kódu oproti společné, namísto vytváření celé oblasti tak říkajíc od píky.

V následujících částech této sekce uvedu několik příkladů použití tohoto principu z výsledné implementace vytvořeného systému.

3.4.1 Vytváření tabulek

Podstatou evidenčních systémů je uchovávání a zobrazování dat spojených s chodem určitého zařízení. Tato data bývají velmi často organizována do tabulek pro lepší přehlednost a možnost vzájemného porovnání.

Ani systém, jehož vývoj je předmětem této práce, není výjimkou z tohoto pravidla, a obsahuje tudíž nemalé množství dat, jež je vhodné zobrazit formou tabulky. Vytvoření nástroje pro usnadnění vytváření tabulek se proto přímo nabízelo.

V rámci vývoje systému jsem vytvořil třídu `GridControl`, která představuje obecný prvek uživatelského rozhraní pro zobrazování tabulek. Základními prvky pro definování tabulky jsou sloupce a akce.

Sloupce tabulky umožňují zobrazování dat daného řádku a jejich úpravu. Kromě toho je možné nastavit, aby šlo data v tabulce podle daného sloupce řadit. V současné době systém podporuje sloupce pro celá a desetinná čísla, text a data.

Pokud by ovšem vznikla potřeba pro zobrazování nového typu dat (například výběr z hodnot), stačí vytvořit potomka třídy `GridColumn` a definovat, jakým způsobem se má v tabulce vykreslovat. Zbytek si třída `GridControl` zařídí sama.

Akce tabulky se dělí na dva druhy – obecné a spojené s řádkem. Obecné akce se zobrazují pod tabulkou jako tlačítka, zatímco akce spojené s řádkem se zobrazují ve sloupci „Akce“ v řádku, s nímž jsou spojené. U obou druhů akcí je potřeba specifikovat, co se má stát po jejich zvolení. Zároveň je možné tyto akce podmínit potvrzením ze strany uživatele.

Kromě těchto volitelných akcí nabízí třída `GridControl` také možnost definovat dvě základní akce. Těmi jsou přidání (obecná akce) a úprava (akce spojená s řádkem) položky. Tyto akce jsou interně spravovány tabulkou a jejich definování se řídí lehce odlišnými pravidly.

Vzhledem k velké variabilitě parametrů spojených s procesem definování tabulky jsem se rozhodl vytvořit pro tyto účely pomocné třídy podle návrhového vzoru *Stavitel* (z angl. *Builder*) [22]. Tyto pomocné třídy výrazně zpře-

hledňují inicializaci složitějších objektů, odstraňují nutnost předávat výchozí hodnoty pro nepoužité parametry a zaručují, že objekt bude vždy vytvořen ve validním stavu.

Zmíněný návrhový vzor je použit pro vytváření tabulek samotných (třída `GridBuilder`), jejich sloupců (třída `BaseGridColumnBuilder` a její potomci) a akcí (třídy `GeneralGridActionBuilder` a `ItemGridActionBuilder`).

Pro upevnění představy si může čtenář prohlédnout ukázku kódu 1, která zobrazuje (pro přehlednost zjednodušené) využití zmíněného návrhového vzoru při vytváření tabulky na stránce pro správu zboží.

3.4.2 Přidávání a úprava položek

Ačkoliv třída `GridControl` poskytuje podporu pro velkou část tabulek systému, není ji možné kvůli její vnitřní reprezentaci použít uvnitř jiného formuláře. Tato skutečnost bohužel znemožňuje její použití na stránkách pro úpravu obsahu nákupu, požadavku a jim podobných.

Přesto i stránky, které umožňují práci s položkami (nákupů, požadavků, receptů, převozů, ...), nabízejí při přidávání a úpravě těchto položek velmi podobnou funkcionalitu. I zde se nachází prostor pro vyčlenění společných prvků, který jsem využil.

Vyčleňování společných částí je zde uskutečněno ve dvou úrovních. Základním prvkem pro celou hierarchii je třída `BaseItemForm`, která zajišťuje procesy pro práci s položkami (tj. zbožím a jeho typem a množstvím).

V následující úrovni dědičné hierarchie jsou třídy, jež upravují chování podle kontextu, v němž jsou položky použity. Jedná se o třídy `PurchaseItemForm` pro nákupy, `RecipeItemForm` pro recepty, `RequirementItemForm` pro požadavky, `TakeOutItemForm` pro výdej zboží ze skladu a `TransportItemForm` pro převozy.

Přidávání a úprava položek se v rámci těchto skupin liší pouze rozložením prvků na zobrazované stránce a procesem, jež má být proveden po dokončení přidání (resp. úpravy) položky. Z tohoto důvodu je každá ze zmíněných tříd dále rozšiřována třídami pro přidání (prefix `Add`) a úpravu (prefix `Edit`) daných položek. Tyto třídy upřesňují části, v nichž má být chování těchto stránek v rámci dané skupiny odlišné.

Díky této struktuře je možné novou stránku v rámci existující části hierarchie přidat za pomocí napsání necelých padesáti řádků kódu včetně dokumentace.

3.4.3 Pomocné třídy

Během vývoje systémů se velmi často programátoři setkávají s problémy, které je potřeba řešit na velkém počtu míst. Jedná se například o řazení polí, porovnávání hodnot s plovoucí desetinnou čárkou, práci se soubory nebo kontrolu hodnot zadaných uživateli.

```
GridBuilder::newBuilder( 'Management:fsContinue', function () {
        return GridFoodstuff::fromFoodstuff( ... );
}, 'V tuto chvíli není v systému žádné zboží.' )
->addColumn(
        TextGridColumnBuilder::newBuilder( 'Typ', 'type' )
                ->makeComparable()
                ->build() )
->addColumn(
        TextGridColumnBuilder::newBuilder( 'Název', 'name' )
                ->makeComparable()
                ->makeEditable()
                ->build() )
->addColumn(
        TextGridColumnBuilder::newBuilder( 'Jednotky', 'unit' )
                ->build() )
->setEditAction( function ( $foodstuff, $values ) {
        ...
} )
->setEditActionValidator( function ( $form,
        $editedFoodstuff ) {
        return ...
} )
->addItemAction(
        ItemGridActionBuilder::newBuilder( 'fas fa-trash',
            'Smazat', function ( $foodstuff ) {
                ...
        } )->addConfirmation( function ( $foodstuff ) {
                return 'Opravdu chcete smazat toto zboží?';
        } )->addConfirmation( function () {
                return 'Jste si opravdu jistí?';
        } )->build() )
->addGeneralAction(
        GeneralGridActionBuilder::newBuilder( 'fas fa-plus',
            'Přidat zboží', function () {
                ...
        } )->build() )
->build( $this->gridFactory );
```

Ukázka kódu 1: Využití třídy `GridBuilder` pro vytvoření tabulky na stránce pro správu zboží

Zároveň se často během programování stává, že je potřeba aplikovat určitou výpočetní logiku na místě, které je již samo o sobě dost komplikované na to, aby do něj bylo vloženo dalších sto řádků výpočtů.

V obou těchto případech je záhodno zmíněné kusy zdrojového kódu vyčlenit do tzv. „pomocných tříd“. Tento přístup vede ke zvýšení čitelnosti částí kódu, z nichž se zmíněné pomocné třídy volají, a prevenci chyb při opravování či upravování kódu, který se v pomocných třídách nachází.

Při vývoji systému jsem identifikoval a vymezil pomocné třídy, jež zajišťují:

- správné porovnávání čísel s desetinnou čárkou a dat,
- odesílání emailů,
- práci se soubory,
- usnadnění práce s kolekcemi (inspirováno jazykem LINQ [23]),
- usnadnění práce s formuláři a jejich validaci,
- korektní správu entit systému,
- správu menu,
- práci s uživatelskými upozorněními,
- výpočty položek přehledů.

V rámci ukázky kódu 2 je vidět příklad práce s pomocnou třídou pro validaci formulářů `FormChecker`. Tato třída je navržena na základě návrhového vzoru *Stavitel* (z angl. *Builder*) [22].

3.5 Budoucí rozšíření

Ačkoliv jsem k vytváření systému přistupoval zodpovědně a každou z jeho částí jsem udělal podle svého nejlepšího vědomí a svědomí, systém stále obsahuje spoustu místa pro vylepšení a rozšíření, na něž v rámci této práce již nezbyl čas.

V následujících odstavcích daná místa vymezím. Zároveň zmíním několik poznámek, které mohou pomoci komukoliv, kdo by se rozhodl ve vývoji systému pokračovat.

Vylepšení vzhledu

Současný vzhled systému bych osobně označil za ucházející, avšak k dokonalému má ještě daleko. Tuto situaci bych přisoudil svým velmi skromným zkušenostem s kaskádovými styly a naprosto minimálním zkušenostem v oblasti grafického návrhu.

```php
/**
 * Validates the input from the "Record" forms.
 *
 * @param Form $form The form this function reacts to.
 *
 * @return bool True in case the input is correct;
 *          false otherwise.
 */
protected function validateForm( Form $form ) : bool
{
    $isValid = true;
    $values = $form->getValues();

    FormChecker::newChecker( $form, $values->name )
            ->notEmpty( 'Není zadán název' )
            ->updateValidationStatus( $isValid );

    FormChecker::newChecker( $form, $values->amount )
            ->notEmpty( 'Není zadáno množství' )
            ->isInteger( 'Počet musí být celé číslo' )
            ->isNonNegative( 'Počet musí být nezáporné číslo' )
            ->updateValidationStatus( $isValid );

    FormChecker::newChecker( $form, $values->from )
            ->notEmpty( 'Není zadáno datum začátku' )
            ->isDate( 'j. n. Y', 'Datum začátku nemá správný'
                .' formát - pro výběr použijte kalendář' )
            ->updateValidationStatus( $isValid );

    FormChecker::newChecker( $form, $values->to )
            ->notEmpty( 'Není zadáno datum konce' )
            ->isDate( 'j. n. Y', 'Datum konce nemá správný'
                .'formát - pro výběr použijte kalendář' )
            ->isNotBefore( 'j. n. Y', $values->from, 'Datum'
                .'konce nesmí být před datem začátku' )
            ->updateValidationStatus( $isValid );

    return $isValid;
}
```

Ukázka kódu 2: Využití třídy `FormChecker` pro validaci formuláře pro přidání či úpravu záznamů o počtech strávníků

Přesto pevně věřím, že pro kohokoliv se zájmem pro grafickou stránku webových aplikací nebude velký problém dát systému takříkajíc nový kabát. Jeho práci by měl usnadnit i současný způsob sestavení kaskádových stylů, který umožňuje kontrolovat barevné schéma a základní velikosti prvků uživatelského rozhraní z jednoho souboru.

AJAX

Jedním z běžně používaných prostředků pro zlepšení uživatelské přívětivosti moderních webových aplikací je využití technologie AJAX [24]. Tato technologie umožňuje pomocí asynchronní komunikace načítat a překreslovat při požadavcích pouze určité části webové stránky.

Ačkoliv framework Nette [15] poskytuje podporu této technologie, rozhodl jsem se nakonec od jejího využití upustit. Důvodem pro toto rozhodnutí byl nedostatek zkušeností se zmíněnou technologií a komplikace, jež vyvstaly při jejím zprovozňování.

Šířka a škálování tabulek

V sekci 3.3.2 jsem rozebíral problematiku spojenou se šířkou a škálovatelností tabulek v závislosti na rozlišení obrazovky a obsahu škálované tabulky.

Přestože se mi během implementace podařilo najít pouze suboptimální kompromis, věřím, že tento problém bude mít jednoduché a elegantní řešení. Jeho nalezení však nechávám jako předmět snah budoucích.

Doplňkové požadavky

Dalším ze zdrojů inspirace pro budoucí rozšíření systému jsou doplňkové požadavky. Současná implementace systému jejich velkou část pokrývá, nikoliv ovšem všechny.

Jmenovitě nebyly v rámci první implementace systému splněny následující doplňkové požadavky:

- F4 – vytváření tisknutelné verze receptů
- F11 – přidání možnosti zobrazení a filtrování historie provedených akcí

Kromě zmíněných doplňkových požadavků byla během implementace navržena následující potenciální rozšíření:

- zahrnutí informací o dodavatelích do přehledů
- zobrazení počtu strávníků ve zvoleném areálu pro blízké dny

Tato rozšíření by měla být před jejich začleněním do systému schválena zodpovědnou osobou ze strany agentury Haul.

Vylepšení třídy `GridControl`

I přese všechnu funkcionalitu a benefity, které jsou spojené s využíváním třídy `GridControl`, mám stále pocit, že by bylo možné využití této komponenty ještě více rozšířit.

Většina omezení, s nimiž jsem se nebyl schopen vypořádat, totiž spočívá v mé neschopnosti použít framework takovým způsobem, aby bylo možné vložit interní formulář spojený se třídou `GridControl` do formuláře řídícího nadřazenou stránku. Pokud by se ovšem budoucí rozšiřující vyvinutého systému uměl s tímto omezením vypořádat, otevírala by se pro třídu `GridControl` úplně nová dimenze možností využití.

Výběr existující hodnoty pomocí šipek

Jedním z nejčastěji používaných prvků uživatelského rozhraní je prvek pro volbu jedné z existujících hodnot. Ačkoliv je jeho používání odladěno pro většinu způsobů použití, jeden z klíčových implementován nebyl.

Jedná se o navigaci v rámci nabízeného seznamu podobných existujících hodnot pomocí šipek nahoru a dolů. Tato funkce je využívána především v případě, že seznam podobných existujících hodnot obsahuje na prvních několika místech vzájemně podobné hodnoty a uživatel potřebuje zvolit jinou než první nabízenou.

Zmíněná funkcionalita nejspíše představuje nevelký zásah do souboru kontrolujícího chování prvků pro výběr existující hodnoty (`picker.js`). Pro její implementaci v rámci této práce však již nezbyl čas.

Větší využití šablonovacího systému

Na rozdíl od částí *model* a *presenter* vytvořeného systému, které byly zbaveny drtivé většiny duplicit, část *view* v tomto ohledu zaostává.

Součástí frameworku Nette [15] je mimo jiné také šablonovací systém Latte [25], který by těmto situacím měl pomáhat předcházet. Přesto se mi i přes mé četné snahy nepodařilo přijít na způsob, jakým vyčlenit společné prvky při zachování možnosti jejich přizpůsobení různým kontextům, ve kterých jsou použity.

Podobně jako v ostatních případech viním z této skutečnosti nedostatek mých zkušeností s tímto šablonovacím systémem a věřím, že v rukou profesionála bude tento problém snadno odstraněn.

Vylepšení napovídání

Napříč systémem existuje několik různých situací, ve kterých systém zobrazuje výsledky podobné výrazu zadanému uživatelem. V současné době se za relevantní položky považují ty, pro něž platí, že zadaný výraz je souvislým podřetězcem jedné z prohledávaných vlastností.

Ačkoliv toto napovídání plní svůj účel, je ve své podstatě velmi naivní a nedokáže správně vyhodnocovat složitější vyhledávací vzory. Toto může představovat nepříjemnost pro zdatnější uživatele systému.

Řešením by mohlo být použít jednu ze sofistikovanějších heuristik pro hledání relevantních položek jako například rozumně nastavenou Levenshteinovu vzdálenost [26] či hledání nesouvislých podřetězců. Při volbě heuristiky je ovšem nutné brát v potaz, že toto prohledávání nesmí být příliš náročné, neboť je prováděno při každé změně zadávaného výrazu.

3.6 Nasazení

Nedílnou částí vývoje libovolného systému je jeho nasazení. Tento proces by měl v ideálním případě představovat co nejmenší psychickou a časovou zátěž pro koncové uživatele.

Vzhledem k tomu, že vytvořený systém je webová aplikace, proces nasazení pro koncové uživatele kompletně odpadá, neboť jim pro přístup k systému stačí mít nainstalovaný internetový prohlížeč a znát adresu, na níž se systém nachází.

Břímě nasazení systému na dedikovaný server tedy připadá na jeho správce. Pro usnadnění tohoto procesu byla vytvořena série skriptů, které se starají o konfiguraci částí systému, jež jsou nezávislé na typu serveru, na němž má systém fungovat. Nastavení samotného serveru se přímo odvíjí od jeho typu, a je tudíž ponecháno na jeho správci.

Návod pro použití zmíněné série skriptů společně s podrobnějším popisem jejich parametrů a průběhem konfigurace jsem sepsal do instalační příručky. Tu je možné nalézt na médiu přiloženém k této práci.

Vytvořený systém byl úspěšně nasazen pro účely testování. Prostor na jednom ze svých serverů mi pro tyto účely poskytla Fakulta informačních technologií ČVUT.

Nasazení produkční verze systému bohužel zkomplikovala současná situace spojená s onemocněním COVID-19. Agentura Haul se, podobně jako většina společností, v současné době potýká s problémy spojenými s omezeními, která krizová situace přináší. Zároveň bylo vedení agentury Haul nutno pozměnit pracovní postupy většiny svých zaměstnanců.

S nasazením systému jsou spojené dodatečné náklady na provoz serveru, na němž systém poběží, a dobou nutnou pro seznámení zaměstnanců s tímto systémem. Z obavy o nadměrné zatížení svých zaměstnanců a dodatečné finanční náklady se vedení agentury Haul rozhodlo odložit nasazení vyvinutého systému až na sezónu roku 2021.

Tato skutečnost pro mě ovšem neznamená žádné větší komplikace. Systém bude v době odevzdání této práce připraven k nasazení včetně uživatelské a instalační příručky. Po domluvě s vedením agentury Haul jsem souhlasil, že nasazení pro příští sezónu provedu.

Zapojení vytvořeného evidenčního systému do procesů zaměstnanců agentury Haul až v rámci příští sezóny ovšem nemá pouze stinné stránky. Tento odklad poskytuje více než dostatečný prostor pro opravu chyb nalezených při testování a implementaci doplňkových požadavků. Zároveň dává zaměstnancům agentury Haul mnohem více času pro sžití se s novým systémem.

Kapitola 4

Testování

I přes maximální úsilí a pečlivost, jež programátoři věnují vývoji softwarových systémů, je statisticky velmi nepravděpodobné, že by výsledek jejich práce byl ihned po dokončení vývoje perfektní. Často se stává, že i po mnohonásobných kontrolách zůstávají v nápovědách překlepy, hodnoty z polí se nepropisují do databáze a položky v menu se řadí každé liché úterý v měsíci v opačném pořadí, než by měly.

Zmíněné vady nemívají velký dopad na samotnou funkcionalitu systému, ovšem mohou mít značný dopad na pověst vývojářů, jež systém s těmito vadami dodali. Z tohoto důvodu se mezi dokončení vývoje a nasazení systému do produkčního prostředí zařazuje testování, během něhož je většina těchto chyb odhalena a napravena.

Předmětem této kapitoly, jak již název a předmluva napovídají, je testování vyvinutého systému. Nejprve čtenáře seznámím s výsledky podrobení systému Nielsenově heuristické analýze [27]. Následně popíši průběh a přínosy uživatelského testování. V závěru kapitoly se budu věnovat vytvořeným integračním testům.

4.1 Nielsenova heuristická analýza

V rámci testování jsem se rozhodl nejprve podrobit vytvořený systém Nielsenově heuristické analýze [27]. Jedná se o deset aspektů systému, u nichž je zkoumáno, zda jsou v souladu s ověřenými praktikami.

Tato heuristická analýza se primárně používá jako pomůcka při testování uživatelského rozhraní experty. Zároveň je pomocí ní možné odhalit některé neduhy systému ještě před zahájením uživatelských testů. Toto může předejít výtkám, které by se jinak nejspíše objevily ve většině názorů testujících uživatelů.

Ve zbytku této části projdu postupně všech deset bodů Nielsenovy heuristické analýzy. U každého nejprve zmíním, k jaké části systému se vztahuje

a co se v souvislosti s ním od analyzovaného systému očekává. V návaznosti na to popíši, zda vyvinutý systém očekávání splnil.

Na konci této části shrnu dílčí výsledky a ke stavu systému se vyjádřím.

4.1.1 Viditelnost stavu systému

Prvním aspektem Nielsenovy heuristické analýzy [27] je viditelnost stavu systému. Uživatel by měl mít při práci se systémem neustále přehled o tom, co se právě děje.

Tento aspekt se nejvíce projevuje u delších operací, při nichž jinak může uživatel nabýt pocitu, že se systém zasekl a přestal odpovídat. Korektní informovanost uživatele zajišťují většinou ukazatele postupu, odhady zbývajícího času či zvuková upozornění.

Vzhledem k tomu, že vyvinutý systém je webová aplikace, pokrývá kratší čekací dobu základní chování prohlížeče, které o čekání informuje uživatele zobrazením indikátoru načítání.

Během vývoje a testování se žádný z uživatelů nedostal do situace, v níž by na odpověď systému musel čekat déle než vteřinu. Tento fakt je ovšem přímo spojený s omezeným množstvím dat, s nimiž systém operoval.

Při delším používání v produkčním procesu může objem dat v systému mnohonásobně narůst, čímž se může podstatně prodloužit reakční doba systému. To by vedlo k nutnosti použití dodatečných indikátorů stavu (např. zobrazení odhadu zbývajícího času), které v současné době systém nepodporuje.

Tento nedostatek mám v plánu řešit podle potřeby a případného přání koncových uživatelů.

4.1.2 Shoda mezi systémem a realitou

Druhý z aspektů si klade za cíl zachování konvencí a metafor z reálného světa. Důležité je především, aby s akcemi systému byly asociovány ikony, které si většina uživatelů s danou akcí přirozeně spojí (např. nota u hudby nebo domeček u návratu na domovskou obrazovku).

Vytvořený systém využívá ikony z volně dostupné sady *Font Awesome* [28], která obsahuje ikony představující většinu běžných akcí. Zároveň jsem ikony volil tak, aby co nejvíce odpovídaly akci, se kterou souvisí.

Přes veškerou mou snahu a obsáhlost sady ikon se přesto v systému nacházejí akce, u nichž použité metafory nejsou ideální. Jedná se především o akci „Rozdělit“ u nákupů a požadavků a akci „Přidělit speciální práva“ u správy uživatelů. Použité ikony jsou k nalezení na obrázcích 4.1 a 4.2.

Pro obě akce byly zvoleny co nejvhodnější dostupné alternativy ze zmíněné sady ikon. Přesto zvolené metafory nejsou zcela intuitivní a mohou uživatele systému mást.

Obrázek 4.1: Použitá ikona akce „Rozdělit“

Obrázek 4.2: Použitá ikona akce „Přidělit speciální práva“

4.1.3 Minimální zodpovědnost a stres

Ke zvýšení produktivity uživatelů při práci se systémem je vhodné zajistit, aby se uživatelům s daným systémem pracovalo pohodlně. Tomu ve velké míře přispívá vědomí, že při práci se systémem nemohou nic pokazit.

Primárními nástroji pro docílení tohoto efektu jsou zpřístupnění operace „Zpět“ (angl. „Undo“), možnost zrušení dlouho trvajících procesů či zobrazení varování před provedením nevratné akce.

Vytvořený systém je webová aplikace, jež umožňuje většímu množství uživatelů operovat nad jednou společnou sadou dat. Jakákoliv změna provedená jedním uživatelem může přímo ovlivnit rozhodování uživatelů dalších.

Během návrhu systému jsem bohužel nepřišel na způsob, jakým bych mohl implementovat operaci „Zpět“, aniž bych narušil plynulost spolupráce uživatelů.

Většina akcí systému je ovšem buď lehce napravitelná (např. nechtěné odebrání položky z nákupu), nebo je před jejím vykonáním nutné provedení dané akce potvrdit. U akcí, jež mají velký dopad na systém (např. odebrání areálu) je vyžadováno potvrzení dvojí.

Kombinace těchto přístupů by měla vést k minimalizaci strachu z nechtěného provedení negativního zásahu do systému a alespoň částečně kompenzovat nepřítomnost operace „Zpět“.

4.1.4 Shoda s použitou platformou a obecnými standardy

Seznamování se s novým systémem představuje pro mnohé uživatele velkou výzvu. Většinou proto uvítají, když narazí na komponenty nebo určité části systému, na něž jsou zvyklí z jiných aplikací.

Během vytváření systému jsem se držel termínů, jež se pojí s gastroekonomickou doménou, neboť je s ní vytvořený systém úzce spjatý. Zároveň jsem vyvinul snahu o udržení konzistence v používání elementů systému (ikony, názvy, pořadí prvků, ...).

Největším prohřeškem vytvořeného systému v rámci tohoto aspektu prováděné analýzy je vzhled systému. V současné době systém nepoužívá žádné ze standardních řešení této problematiky (jako např. Bootstrap [29]), ne-

boť se mi při jejich používání nepodařilo správně nastavit některé prvky uživatelského rozhraní.

4.1.5 Prevence chyb

Během interakce se systémem se běžně stává, že se uživatelé pokusí nedopatřením zadat neplatnou hodnotu. Systém by měl tuto situaci detekovat a co nejdříve na ni uživatele upozornit. Čím déle totiž uživatel dále pracuje s neplatnou hodnotu, tím více stoupá riziko nutnosti opakování daného úkolu, což vede k (pochopitelně) velké nelibosti uživatelů.

Vyvinutý systém poskytuje dva základní přístupy pro zadávání dat. První představují klasické formuláře (např. pro přihlášení, přidání uživatele, ...), druhým je možnost interaktivní úpravy dat v tabulkách.

Formuláře obsahují mechanismus pro okamžitou kontrolu zadávaných dat a případné chyby zobrazují přímo u chybně vyplněného pole. Tím tento aspekt heuristické analýzy zcela pokrývají.

To samé se bohužel nedá říct o úpravě hodnot v tabulkách. Prvním problémem je variabilní validace dat (pro podrobnosti vizte sekci 3.3.4), která znemožňuje použití okamžité kontroly zadávaných dat. Případné chybové hlášky jsou zobrazovány v prostoru nad upravovanou tabulkou, neboť omezený prostor buněk tabulky neumožňuje jejich vzhledné zobrazení.

Ve výsledku je tedy uživatel na chybný vstup upozorněn v případě tabulek až po odeslání dat a kvůli rozměrům tabulky mohou být chybové hlášky velmi vzdáleny polím, ke kterým se vztahují. Dalším problémem tabulek je, že neposkytují informaci o tom, která z polí jsou při editaci záznamu povinná.

4.1.6 Kouknu a vidím

Vykonávání činností uvnitř systému často vyžaduje většinu pozornosti uživatelů. Systém by proto měl klást co nejmenší dodatečné nároky na uživatelovu paměť a pozornost. Kýženého efektu je možné docílit zobrazením všech informací, které jsou potřeba pro dokončení aktuálně prováděného úkolu, aktuální pozice v systému či v rámci složitějšího procesu.

Systém zmíněné kritérium z velké části splňuje. Přesto existuje několik případů užití, při nichž je uživatel nucen si nemalé množství informací z různých částí systému pamatovat. Nejzářnějším příkladem tohoto nedostatku je proces dokoupení (či zažádání o nákup) chybějících potravin. V něm si uživatel musí pamatovat, které potraviny chybí (částečně nahrazeno upozorněními) a jaké je pro ně nastaveno minimální vyžadované množství.

Dalším prohřeškem je nedostatečná indikace pozice v rámci systému. Největší komplikací v tomto ohledu je struktura systému, která není stromová, a tudíž znemožňuje typické využití tzv. „drobečkového menu“ či naznačení pozice zvýrazněním použité položky menu. Tento nedostatek je alespoň částečně kompenzován názvem aktuální činnosti v horní části stránky.

4.1.7 Flexibilita a efektivita

Spektrum úrovní zdatnosti uživatelů při práci s počítačem je velmi široké. Na jednom konci zmíněného spektra jsou uživatelé, jejichž hlavním komunikačním prostředkem při práci se systémem je počítačová myš a klávesnice se dotýkají pouze, pokud není jiného zbytí. Protipólem těchto uživatelů jsou profesionálové, kteří se cítí osobně dotčeni, pokud není možné systém ovládat výhradně pomocí klávesových zkratek.

Ideální systém by měl být schopný poskytnout komfort uživateli z jakékoliv části zmíněného spektra. Pro ovládání systému by tedy mělo být možné používat funkční klávesy či komplikovanější makra. Systém by ovšem tento přístup neměl vyžadovat a pokročilé funkce by měly být dostupné pouze na vyžádání, aby svojí přítomností nemátly méně zkušené uživatele.

Vyvinutý systém je vzhledem k požadavkům na jednoduchost primárně přizpůsoben méně zkušeným uživatelům. Přesto poskytuje některé z pokročilých funkcí jako například používání tabulátoru pro přepínání mezi prvky uživatelského rozhraní. Počet těchto funkcí ovšem nepovažuji za dostatečný, abych mohl s klidným svědomím prohlásit tento bod heuristické analýzy za dostatečně splněný.

4.1.8 Minimalita

Smyslem většiny systémů je maximalizovat efektivitu uživatelů při plnění úkonů, k nimž byl daný systém vytvořen. K docílení tohoto záměru je potřeba udržet uživatele co nejvíce soustředěného na úkol, jež má plnit. Systém by se tedy měl snažit zobrazovat pouze informace relevantní pro splnění současného úkolu.

Tento aspekt prováděné analýzy považuji za jediný, který se vytvořenému systému podařilo bez výhrady splnit. Jedinou částečnou výjimkou jsou obrazovky pro interaktivní úpravu položek tabulek, na nichž jsou kromě tabulek samotných zobrazena i znepřístupněná tlačítka s akcemi souvisejícími s celou tabulkou. Tato tlačítka byla na zmíněných obrazovkách ponechána, aby uživatele nemátlo, „kam všechna tlačítka zmizí, když začnou upravovat řádek“.

4.1.9 Smysluplné chybové hlášky

V ideálním případě by systémy vůbec neměly mít potřebu pro zobrazování chybových hlášek, neboť by všem chybám měly předcházet (vizte sekci 4.1.5). Pokud ovšem systém potřebuje určitou chybovou hlášku zobrazit, měla by tato chybová hláška splňovat určité charakteristiky.

V první řadě by měla uživateli poskytnout vyčerpávající ale stručný popis toho, co a jak se stalo. Kromě toho by měla uživatele poučit, jak podobné chybě v budoucnu předejít, a toto vysvětlení případně doplnit demonstrativními příklady.

Osobně považuji první dvě kýžené vlastnosti chybových hlášení ve vytvořeném systému za dostatečně splněné. V rámci tabulek jsou zároveň problematická místa zvýrazněna červenou barvou a ikonou vykřičníku, která je opatřena dodatečným vysvětlením, jež je zobrazeno při pohybu kurzoru nad onou ikonou.

Prostor pro zlepšení vidím ve zvýšení naučné hodnoty chybových hlášení, neboť v současné době pouze minimum z nich toto kritérium dostatečně splňuje. Na druhou stranu by u mnohých z chybových hlášení bylo dodatečné vysvětlení zbytečné (např. u chyby „Je potřeba zadat množství“).

4.1.10 Nápověda a dokumentace

Ačkoliv by ideální systém měl aspirovat na naprostou pochopitelnost bez dodatečného vysvětlování, přítomnost nápovědy je nutností. Tato nápověda by měla být jednoduše dostupná přímo ze systému a měla by podporovat vyhledávání. Kromě této nápovědy je možné také použít kontextovou nápovědu (vysvětlení zkratek či složitějších akcí na místě jejich použití).

Vytvořený systém přímo přístupnou nápovědou bohužel nedisponuje. Částečnou kompenzací tohoto nedostatku jsou kontextové nápovědy, které jsou přítomny u všech akcí, jež jsou reprezentovány pouze ikonou, a některých složitějších možností (např. vytvoření každodenního nákupu). Systém je zároveň vybaven uživatelskou příručkou.

4.1.11 Zhodnocení analýzy

Vyvinutý systém si v rámci provedené heuristické analýzy nevedl špatně. Většina bodů je buď dostatečně pokryta, nebo obsahuje pouze drobné nedostatky, které by neměly bránit jeho nasazení. I přes tento lichotivý výsledek ovšem rozhodně nepovažuji vyvinutý systém za dokonalý.

Provedená analýza identifikovala nezanedbatelné množství aspektů, které skýtají prostor pro vylepšení. Některé z těchto aspektů (např. vylepšení vzhledu systému nebo vylepšení práce s tabulkami) jsem identifikoval již během implementace (vizte sekci 3.5), avšak mnohé by bez provedení analýzy zůstaly opomenuté.

Za nově objevené oblasti pro budoucí vylepšení, kterým by měla být věnována největší pozornost, považuji prevenci chyb při práci s tabulkami, přidání funkcí usnadňujících práci pokročilým uživatelům a vytvoření nápovědy, jež by byla dostupná přímo ze systému.

4.2 Uživatelské testování

Hlavní složku testování systému vytvořeného v rámci mé diplomové práce představuje uživatelské testování. Během tohoto typu testování musí systém

obstát v situacích, které jsou navrženy tak, aby v co největší míře odpovídaly jeho následnému reálnému použití.

Uživatelské testování umožňuje také ověřit, zda přístupy a případy užití popsané během návrhu systému odpovídají způsobu, jakým uživatelé budou systém používat. Velmi často se proto stává, že testující uživatelé přijdou s nápadem, jak proces práce se systémem zpřehlednit či zjednodušit.

Odlišný přístup k řešení předestřeného problému může zároveň vytvořit situace, s nimiž se při vývoji příliš nepočítalo, a které jsou tudíž mnohem náchylnější na výskyt chyb.

V rámci této části tedy nejprve popíši způsob, jakým byly uživatelské testy vytvořeny a strukturovány. Následně vylíčím průběh testování s jednotlivými uživateli. Na závěr shrnu získané výsledky a zpětnou vazbu uživatelů a vyvodím z nich závěry.

Pro účely testování mi byl poskytnut prostor na jednom ze serverů Fakulty informačních technologií ČVUT, na který jsem umístil testovací verzi vytvořeného systému. Tento server je v době testování dostupný z adresy: haul.fit.cvut.cz/sklad.

4.2.1 Struktura

Během vytváření uživatelských testů je potřeba najít rozumný kompromis mezi délkou testování a jeho pokrytí funkcí systému. Pokud by mělo testování pokrýt každou funkci systému, jeho délka by se prodloužila do rozměrů, které by pro většinu uživatelů nebyly snesitelné. Příliš krátké testování by ovšem nemělo dostatečnou vypovídající hodnotu.

Z těchto důvodů jsem se v rámci tohoto typu testování rozhodl pokrýt situace, s nimiž se koncoví uživatelé systému budou potkávat nejčastěji. I přes mé snahy omezit rozsah testování trvalo testování s jedním uživatelem průměrně zhruba padesát minut. Na základě pozorování testujících uživatelů si myslím, že tato doba je hraniční a její prodloužení by se negativně podepsalo na náladě a soustředěnosti testujících uživatelů.

Samotné testování je rozdělené do dvou částí. V první části plní uživatelé úkoly spojené s procesy zaměstnanců areálu. Druhá část je věnována procesům, s nimiž se běžně potkávají zaměstnanci obchodního oddělení.

Podrobná struktura uživatelského testování je zachycena v testovacím scénáři, který se nachází v příloze D této práce.

4.2.2 Průběh

Tato část je věnována průběhu uživatelského testování s jednotlivými testujícími uživateli. U každého uživatele zmíním věkovou skupinu, ke které náleží, a jeho zaměstnání.

Následně projdu rámcový průběh testování s daným uživatelem, ve kterém se primárně zaměřím na úkoly, které dotyčnému činily obtíže. Následně

uvedu jeho připomínky k systému, nápady, jež by podle dotyčného vedly k vylepšení testovaného systému, a případně chyby systému, které se mu podařilo během testování odhalit.

Původně jsem chtěl provádět každé testování osobně. Bohužel kvůli omezením spojeným s onemocněním COVID-19 jsem byl nucen tuto strategii přehodnotit. Většinu testování s uživateli jsem tedy prováděl tak, že jsem se s daným uživatelem spojil pomocí videokonference a jeho akce jsem sledoval (a nahrával) z přenosu jeho obrazovky, o nějž se ho předtím požádal.

Ačkoliv jsem se bál negativního vlivu tohoto způsobu vedení testování na jeho průběh, mé obavy se ukázaly být zbytečné. Uživatelé projevili maximální snahu mi s případnými komplikacemi technického charakteru vypomoci. Zároveň je možné, že se při testování cítili pohodlněji, neboť mohli testovat z pohodlí svých domovů a nebyli rušeni mou fyzickou přítomností.

Další komplikací, kterou s sebou přinesla omezení spojená s onemocněním COVID-19, je výrazné omezení počtu zaměstnanců areálu Haul, kteří se mohli zapojit do uživatelského testování. Původně jsem doufal v účast alespoň dvou zaměstnanců areálu a dvou zaměstnanců obchodního oddělení. V důsledku nastalé situace jsem ovšem byl nakonec rád, že jsem byl schopný systém otestovat alespoň s Mgr. Janou Dvořákovou, vedoucí obchodního oddělení, za což jsem jí neskonale vděčný.

Předtím než se pustím do popisu jednotlivých testování bych rád uvedl, že všichni testující uživatelé byli seznámeni s kontextem provedeného testování. Zároveň všichni souhlasili s uvedením jejich údajů a poznatků získaných během testování v rámci této diplomové práce.

Viktorie Fedrselová

Prvním z uživatelů, kteří testovali vytvořený systém, byla slečna Viktorie Fedrselová. Slečna Fedrselová náleží do věkové skupiny lidí mladších dvaceti let a v současné době dokončuje svá studia na svitavském gymnáziu.

Slečna Fedrselová měla v porovnání s ostatními testujícími uživateli podstatně těžší práci, neboť při testování s ní byla použita první verze testovacího scénáře. Zadání úkolů ve zmíněné verzi jsem musel během testování dovysvětlovat a upřesňovat, neboť jeho kvalita nebyla odpovídající a vedla ke značné frustraci testující slečny.

Původní verze testovacího scénáře obsahovala mimo zadání úkolů i narativní složku, jejímž smyslem bylo přiblížit testovaným pracovní prostředí zaměstnanců agentury Haul. Zároveň měla tato složka zadání sloužit k uvolnění atmosféry mezi mnou a testujícím uživatelem.

Přítomnost narativní složky ovšem znatelně prodloužila dobu testování. Z tohoto důvodu jsem se rozhodl ji z upravené verze testovacího scénáře vynechat. Přesto si myslím, že zkoumání „vlivu zapojení narativní složky do uživatelského testování“ by mohl být zajímavý námět na disertační práci.

Největší problémy slečně Fedrselové činilo vytváření porovnávacího přehledu. Tuto skutečnost ovšem přisuzuji spíše nejasnosti zadání, neboť po dostatečném vysvětlení zadaného úkolu již kýžený přehled vytvořila během krátké chvíle.

Během testování také odhalila chybu spojenou se zobrazováním současného množství potravin na jednotlivých skladech v rámci vytváření převozu mezi areály. Zároveň přišla testující s podnětem pro přidání kontextové nápovědy pro akce na hlavní obrazovce systému.

Ing. Oldřich Malec

O účast na uživatelském testování vytvořeného systému jsem požádal také Ing. Oldřicha Malce. Ing. Malec se ve své diplomové práci [9] věnoval vytváření frontendu skladového systému, což byl důvod, proč jsem se rozhodl ji rozebrat v rámci analýzy a proč jsem Ing. Malce o účast v testování požádal.

Ing. Malec je projektovým manažerem specializujícím se na vývoj webových aplikací. Jeho věk ho řadí do skupiny lidí mezi dvaceti a dvaceti devíti lety.

Během testování bylo patrné, že se Ing. Malec na vytváření i testování podobných systémů již někdy podílel. Většinu úkolů zvládl rychle a s přehledem, u některých navíc vyzkoušel odolnost uživatelského rozhraní testovaného systému.

Při testování jsme odhalili chybu spojenou s výběrem hodnoty, která se nachází v části seznamu podobných hodnot, která není při prvotním rozbalení vidět. Vybrání takové hodnoty vedlo pouze na sbalení seznamu podobných hodnot, avšak nikoliv ke zvolení vybrané hodnoty.

Kromě odhalení zmíněné chyby přišel testující s celou řadou nápadů na potenciální vylepšení systému. Mezi nejpodstatnější patřily:

- usnadnění procesu pořízení chybějících surovin pomocí přidání dedikovaného tlačítka na obrazovku s limity potravin, které by umožňovalo vytvořit nákup nebo požadavek s obsahem podle chybějících potravin,
- přidání hodnoty „Vše“ mezi povolené hodnoty pro výběr zboží na obrazovkách, v nichž není zadání konkrétního zboží povinné,
- přidání výzvy pro uživatele, který se pokusí vybrat ze skladu více zboží typu, než je na skladě, zda chce vytvořit novou položku se stejným typem zboží a množstvím, o které bylo v předchozí položce překročeno množství na skladě,
- ovládání výběru hodnot pomocí šipek (již evidováno v sekci 3.5).

Jitka Müllerová

V rámci shánění lidí, jež by mi pomohli s testováním vytvořeného systému, jsem se snažil volit takové jedince, kteří mají co nejblíže k cílové doméně, abych kompenzoval nedostatek testujících z řad zaměstnanců agentury Haul. Při svém hledání jsem narazil na Jitku Müllerovou.

Paní Jitka Müllerová je obchodní zástupkyní velkoobchodu Wastex. Svým věkem spadá do kategorie lidí mezi čtyřiceti a čtyřiceti devíti lety. Podle svých vlastních slov představuje spíše méně zkušeného uživatele, neboť ke své práci počítač téměř nepoužívá.

Paní Müllerové ze začátku testování chvíli trvalo, než se sžila se základními principy ovládání systému. Nejméně intuitivní pro ni byl způsob interaktivního přidávání a úpravy položek v tabulkách (snažila se přidávat položky klikáním na záhlaví tabulky). Jakmile se se základními principy sžila, plnění zbylých úkolů jí již nečinilo téměř žádné potíže.

Nejpodnětnějším nápadem paní Müllerové pro vylepšení testovaného systému bylo přidání možnosti zadávání jednotkové ceny při nákupu potravin. Podle její zkušenosti je to typický způsob, jakým její společnost potraviny prodává. Zároveň toto řešení nenutí uživatele cenu potravin dopočítávat.

Při zvažování o možnosti zakomponování tohoto podnětu jsem ovšem narazil na problém nejednotnosti jednotkové ceny. V ideálním případě by uživatelé chtěli pouze zadat položku nákupu ve smyslu „kup dvacet balení špaget Panzani po sedmnácti korunách za kus“.

Tento přístup by ovšem vyžadoval specifikování jednotkové ceny pro zmíněné špagety Panzani. Problém se specifikací jednotkové ceny by se dal vyřešit přidáním dalšího sloupce do tabulky pro nákup potravin. Zmíněné rozšíření bude před jeho implementací nutno projednat s vedením agentury Haul.

Ing. Věra Netolická, MBA

Další osobou, jež se rozhodla přispět svými názory ke zvýšení kvality testovaného systému, je Ing. Věra Netolická, MBA. Ing. Netolická je povoláním auditorka a v rámci demografické části předloženého dotazníku se zařadila do věkové skupiny lidí mezi padesáti a padesáti devíti lety.

Ačkoliv o sobě testující tvrdila, že jí je práce s počítači cizí, podařilo se jí velmi rychle seznámit se se základy ovládání systému. Většinu zadaných úkolů následně zvládla bez větších obtíží ve velmi krátkém čase.

Podobně jako paní Müllerová se i Ing. Netolická při seznamování se systémem pokoušela jako první přidat do tabulky nový záznam pomocí klikání na záhlaví tabulky. Vzhledem k tomu, že se jedná o opakovaný úkaz, si začínám klást otázku, zda by nebylo vhodné pokusit se přizpůsobit přístup k editaci dat v tabulkách tak, aby umožňoval přístup i přes vertikální osu.

Během výběru jedné z existujících položek zatím jako jediná používala posuvníku pro pohyb v seznamu podobných položek. V rámci této činnosti si

stěžovala, že má problém posuvník používat, neboť není dostatečně široký.

Vzhledem k tomu, že jsem posuvník bral spíše jako indikátor možnosti posunu, jsem neočekával, že by ho uživatelé mohli chtít pro tyto účely chtít používat. Testující jsem tedy vyhověl a zmíněný posuvník rozšířil na dvojnásobek jeho původní šířky.

Kromě příliš úzkého posuvníku vadila testující při vybírání hodnot také nemožnost vybírat hodnoty ze seznamu podobných pomocí šipek (evidováno v sekci 3.5).

Bc. Ing. Jarka Neklová

Řady testujících uživatelů rozšířila také učitelka v mateřské škole, Bc. Ing. Jarka Neklová. Tato testující spadá do věkové skupiny lidí mezi čtyřiceti a čtyřiceti devíti lety a podle vlastních slov s počítači zacházet umí, avšak ke své práci je používá pouze za účelem vytváření pomocných materiálů pro práci s dětmi.

V porovnání s ostatními testujícími uživateli se jí dařilo nejlépe se orientovat při pohybu aplikací. Z jejích komentářů při plnění úkolů bylo zjevné, že dokázala rychle odhadnout důvod a princip fungování jednotlivých částí procesů spojených s doménou, do níž systém svým zaměřením spadá.

Největší problém Ing. Neklové činilo, podobně jako ostatním testujícím uživatelům, vytvoření požadavku zakoupení chybějících potravin.

Během testování se testující také několikrát setkala s problémem při vybírání hodnoty ze seznamu podobných hodnot. Na základě jejích postřehů ohledně nezvyklého přeskupování částí názvů některých z nabízených položek se mi podařilo odhalit a odstranit skutečnou příčinu chyby, která vedla ke skrývání seznamu podobných hodnot bez uložení hodnoty, kterou uživatel zvolil.

Mgr. Jana Dvořáková

Posledním osobou, jež jsem požádal o uživatelské testování, byla Mgr. Jana Dvořáková. Její názor na vytvořený systém pro mě byl velmi cenný, neboť provázela celý vývoj systému coby představitel postoje agentury Haul. Testování v jejím případě bylo také testováním akceptačním.

Mgr. Dvořáková je zaměstnankyní obchodního oddělení agentury Haul a svým věkem spadá do kategorie lidí mezi třiceti a třiceti devíti lety.

Testování s Mgr. Dvořákovou mělo vzhledem k jejímu postavení v rámci vývojového procesu lehce odlišný charakter než u ostatních testujících. Během průchodu testovacím scénářem velmi často zkoušela dodatečné alternativy a krajní případy, aby si ověřila, že se systém bude schopen přizpůsobit potřebám všech zaměstnanců agentury Haul. Zároveň nastiňovala hypotetické scénáře a zajímalo ji, jak by se v nich systém zachoval.

Vzhledem k tomu, že doména, s níž vytvořený systém souvisí, je Mgr. Dvořákové vlastní, zvládla splnit všechny zadané úkoly (včetně vytvoření požadavku na chybějící zboží) bez nejmenších problémů.

Po skončení uživatelského testování jsem pokračoval představením částí systému, jež tímto testováním pokryty nebyly. Zároveň jsem s Mgr. Dvořákovou prodiskutoval některé z návrhů na potenciální vylepšení, jež vzešly z uživatelského testování s ostatními testujícími.

Za jediný nedostatek vytvořeného systému považovala označení zboží a jeho typu, které by mělo být podle jejího názoru opačně. To by znamenalo, že například u bílých rohlíků Nopek by bylo zboží „bílý rohlík“ a jeho typ by byl „Nopek“.

Společně s požadavkem na změnu označení zboží a jeho typu projevila zájem o zařazení návrhu Ing. Oldřicha Malce na zjednodušení procesu pořizování chybějících potravin. Kromě zmíněných dvou aspektů byla s vytvořeným systémem nadmíru spokojená. Nejvíce ocenila jednoduchost uživatelského rozhraní, jež obsahuje pouze prvky, které jsou potřeba. Tento aspekt systému podle jejího názoru odstraní většinu neochoty ze strany zaměstnanců začít nový systém používat.

4.2.3 Výsledky

V rámci testování vytvořeného systému jsem provedl uživatelské testování s šesti lidmi. Tuto skupinu tvořili zástupci různých věkových kategorií, úrovní dosaženého vzdělání a zkušenosti práce s počítačem i různých zaměstnání. Každý z testujících přinesl svůj osobitý náhled a názor na vytvářený systém, čímž mi umožnili odhalit chyby a potenciál, o nichž jsem předtím netušil.

Největší problém pro většinu testujících uživatelů představovalo vytváření požadavku na nákup chybějících potravin z pohledu zaměstnance areálu (devátý úkol první části testování). Tuto skutečnost připisuji především nutnosti pamatovat si data ze dvou různých částí systému. Zbytek úkolů nečinil nikomu z testujících větší obtíže.

Většina testujících systém hodnotila jako uživatelsky přívětivý, intuitivní a přehledný. Zároveň se testující jednomyslně shodli, že po pochopení základních principů práce s vytvořeným systémem by byli schopni v něm efektivně a spolehlivě pracovat.

Během uživatelského testování bylo odhaleno několik chyb systému a nadneseno mnoho zajímavých podnětů. Všechny nalezené chyby jsem opravil a z nadnesených návrhů jsem dva implementoval. Jednalo se o úpravu označení „zboží“ a „typ“, jež si vyžádala Mgr. Jana Dvořáková, a usnadnění procesu pořizování chybějících potravin podle nápadu Ing. Oldřicha Malce.

Zbylé podněty jsou zaznamenány v sekcích uživatelského testování testujících, kteří danými podněty přispěli. Tato vylepšení jsou zařazena mezi budoucí vylepšení, a nebudou tedy součástí první verze vytvořeného systému.

4.3 Integrační testování

Smyslem integračního testování vytvořeného systému je ověření, že jeho části dokáží správně komunikovat mezi sebou. Jedná o nadstavbu nad tzv. „jednotkovými testy“, které si kladou za cíl ověřit korektní funkcionalitu v rámci jednotlivých částí vytvořeného systému.

V případě vytvořeného systému nemá ovšem tento typ testování velkou přidanou hodnotu, neboť kvůli charakteru systému a základní struktuře dané frameworkem Nette [15], by integrační testy primárně testovaly, zda se správně volají interní metody použitého frameworku. Další nevýhodou použití integračního testování v tomto případě je velký překryv s aspekty, které jsou testovány v rámci testování s uživateli.

Prostor pro využití tohoto typu testování ovšem nalezneme, pokud uděláme pomyslný krok vzad a budeme o vytvořeném systému uvažovat jako o části v procesu jeho propojení se systémem registrací, který ve své bakalářské práci [13] vytváří Anna Vitmanová. V tomto případě je potřeba testovat samotný proces (nikoliv funkčnost frameworku) a tuto oblast uživatelské testy pokrýt nemohou, neboť se jedná o automatizovaný proces, který probíhá periodicky bez zásahu uživatelů.

Integrační testy této oblasti by ovšem těsně svázaly vytvářené systémy. Systém registrací by musel vystavit speciální data určená pro testování, kterým by odpovídaly na straně evidenčního systému testovací případy. Tento přístup by vyžadoval udržování konzistence testovacích dat mezi dvěma systémy, což mi nepřijde ideální.

Další komplikací bylo, že systém registrací neměl v době testování evidenčního systému potřebný přístupový bod připravený. V tomto místě bych se ovšem rád zastal Anny Vitmanové, neboť požadavek na vytvoření zmíněného přístupového bodu byl označen jako doplňkový a dodán až v průběhu implementace. Evidenční systém je přesto na propojení připraven a já jsem chtěl mít jistotu, že v případě propojení systémů s sebou nepřinese kvůli mé nedbalosti chyby.

Z tohoto důvodu jsem vytvořil sadu jednotkových testů, které se zaměřují na procesy spojené se zpracováním dat, jež budou přijímány od systému registrací. Tato sada obsahuje dvanáct komplexních testů, které testují, že evidenční systém zvládne zpracování validních dat a jejich sloučení s existujícími daty v systému. Zároveň ověřují korektní reakci na situace, v nichž dojde k chybě spojení či budou přijata nekompletní nebo z jiného důvodu neplatná data.

Závěr

Cílem této práce bylo vytvoření a nasazení evidenčního systému pro agenturu Haul v souladu se standardy softwarového inženýrství.

V rámci práce jsem nejprve analyzoval současný stav procesů zaměstnanců agentury Haul používaných pro evidenci zboží. Společně s analýzou procesů jsem vymezil funkční a nefunkční požadavky, jež byly na vytvářený systém kladeny. V kontextu těchto požadavků jsem analyzoval některé existující systémy, jež se zabývají problematikou evidence potravin. Na závěr této části diplomové práce jsem definoval případy užití, kterými jsem pokryl vymezené požadavky.

S pomocí poznatků získaných v předchozí části jsem se nejprve rozhodl, že systém bude mít podobu webové aplikace. Následně jsem zvolil technologie, které měly být použity pro implementaci systému, a vytvořil jsem (Lo-Fi) návrh uživatelského rozhraní, jenž byl použit pro ověření, že navržený systém odpovídá představám zaměstnanců agentury Haul. V rámci návrhu jsem také vytvořil a popsal databázový model vytvářeného systému.

Po dokončení návrhu jsem systém pomocí něj implementoval. Výsledná implementace splňuje všechny vymezené klíčové požadavky. Zároveň se mi podařilo realizovat i větší část požadavků doplňkových a přidat několik dalších vylepšení. Během vývoje jsem kladl důraz na to, aby byl zdrojový kód vytvořeného systému lehce udržovatelný a rozšiřitelný.

Zároveň jsem vymezil nemalé množství oblastí, jež skýtají prostor pro budoucí vylepšení vytvořeného systému. Těmito oblastmi jsou především grafická stránka systému, vylepšení komponenty pro práci s tabulkami a poskytnutí větší podpory pokročilým uživatelům při práci se systémem.

Po dokončení implementace byl systém podroben Nielsenově heuristické analýze, uživatelskému a integračnímu testování. Tato testování pomohla odhalit několik chyb a přinesla náměty na zjednodušení některých ze složitějších procesů systému.

Následně byl vytvořený systém představen odpovědné osobě z agentury Haul, která s jeho stavem byla až na pár nedostatků velice spokojena. Vytčené nedostatky se mi podařilo odstranit před vydáním první verze systému. Tímto

byl splněn první z cílů diplomové práce.

Kvůli komplikacím spojeným s onemocněním COVID-19 se vedení agentury rozhodlo odložit zapojení vytvořeného systému do svých pracovních procesů na sezónu roku 2021. Z tohoto důvodu nedošlo k nasazení vytvořeného do produkčního prostředí.

I přes tyto komplikace jsem vytvořil sérii skriptů, jež z velké části usnadňují jeho nasazení a samotný systém pro účely uživatelského testování nasadil na server poskytnutý Fakultou informačních technologií ČVUT. Pro systém jsem také vytvořil instalační a uživatelskou příručku, jež jsou k nalezení na médiu přiloženém k této práci. Zároveň jsem po dohodě s vedením agentury Haul souhlasil, že vytvořený systém do produkčního prostředí nasadím, jakmile to situace dovolí.

Vzhledem k omezením daným současnou situací a rozhodnutí vedení agentury Haul o odložení nasazení systému považuji i tuto část zadání za splněnou i přesto, že k nasazení do produkčního prostředí nedošlo.

Literatura

[1] Haul: Agentura Haul – specialista na rekreaci všeho druhu pro malé i velké. [online], listopad 2019, [cit. 2019-11-30]. Dostupné z: https://www.haul.cz/

[2] Google: Tabulky Google. [online], listopad 2019, [cit. 2019-11-30]. Dostupné z: https://www.google.com/intl/cs_cz/sheets/about/

[3] Altisima: Magdalena LK. [online], listopad 2019, [cit. 2019-11-30]. Dostupné z: https://www.altisima.cz/software/magdalena-lk

[4] Ing. Antonín Marek: Dokumentace Magdalena v.1.66.03. [online], březen 2013, [cit. 2020-02-03]. Dostupné z: https://www.altisima.cz/sites/default/files/files/manuals/184-sklad-dok.pdf

[5] Altisima: GEKON. [online], únor 2020, [cit. 2020-02-04]. Dostupné z: https://www.altisima.cz/software/gekon

[6] VIS Plzeň: Program MSklad. [online], únor 2020, [cit. 2020-02-19]. Dostupné z: https://web.visplzen.cz/produkty/skladova-evidence/program-msklad/

[7] Z-Ware.cz: BonAp MSklad. [online], únor 2020, [cit. 2020-02-19]. Dostupné z: https://www.z-ware.cz/bonap-sklad

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

[9] Bc. Oldřich Malec: *Frontend skladového systému*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, leden 2020. Dostupné z: https://gitlab.fit.cvut.cz/malecold/master-thesis

[10] Jagu s.r.o.: Jagu – Software na míru, webové prezentace, grafika, webhosting. [online], únor 2020, [cit. 2020-02-17]. Dostupné z: https://www.jagu.cz/

[11] Vue.js Core Team: Vue.js. [online], únor 2020, [cit. 2020-02-17]. Dostupné z: https://vuejs.org/

[12] Vuetify LLC: Vue Material Design Component Framework. [online], únor 2020, [cit. 2020-02-17]. Dostupné z: https://vuetifyjs.com/en/

[13] Vitmanová, A.: *Registrační systém agentury Haul*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, květen 2020.

[14] Microsoft: ClickOnce Security and Deployment. [online], únor 2020, [cit. 2020-02-21]. Dostupné z: https://docs.microsoft.com/en-gb/visualstudio/deployment/clickonce-security-and-deployment?view=vs-2019

[15] Nette Foundation: Nette – Pohodlný a bezpečný vývoj webových aplikací v PHP. [online], únor 2020, [cit. 2020-02-24]. Dostupné z: https://nette.org/cs/

[16] SitePoint: The Best PHP Framework for 2015: SitePoint Survey Results. [online], únor 2020, [cit. 2020-02-24]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[17] Balsamiq Studios LLC: Balsamiq. Rapid, effective and fun wireframing software. [online], březen 2020, [cit. 2020-03-04]. Dostupné z: https://balsamiq.com/

[18] Glenn E. Krasner, S. T. P.: A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *ParcPlace Systems, Inc*, 1988, [cit. 2020-05-13].

[19] Nette Foundation: Životní cyklus aplikace. [online], květen 2020, [cit. 2020-05-13]. Dostupné z: https://doc.nette.org/cs/3.0/request-lifecycle

[20] 0b10011 (https://stackoverflow.com/users/526741/0b10011): What has bigger priority: opacity or z-index in browsers? [online], srpen 2014, [cit. 2020-05-11]. Dostupné z: https://stackoverflow.com/a/11742116

[21] Hunt, A.: *The Pragmatic Programmer: From Journeyman to Master*. Reading, Mass: Addison-Wesley, 2000, ISBN 978-0-201-61622-4.

[22] Gamma, E.; Helm, R.; Johnson, R.; aj.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, první vydání, 1994, ISBN 978-0201633610.

[23] Microsoft: Úvod do dotazů LINQ (C#). [online], květen 2020, [cit. 2020-05-12]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries

[24] Nette Foundation: AJAX & snippety. [online], květen 2020, [cit. 2020-05-12]. Dostupné z: https://doc.nette.org/cs/3.0/ajax

[25] Nette Foundation: Latte – nejbezpečnější & opravdu intuitivní šablony pro PHP. [online], květen 2020, [cit. 2020-05-12]. Dostupné z: https://latte.nette.org/cs/

[26] Miller, F. P.; Vandome, A. F.; McBrewster, J.: *Levenshtein Distance: Information Theory*. Alpha Press, 2009, ISBN 978-613-0-21690-0.

[27] Nielsen, J.: Finding Usability Problems through Heuristic Evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, New York, NY, USA: Association for Computing Machinery, 1992, ISBN 0897915135, str. 373–380, doi:10.1145/142750.142834. Dostupné z: https://doi.org/10.1145/142750.142834

[28] Fonticons, .: Font Awesome. [online], květen 2020, [cit. 2020-05-16]. Dostupné z: https://fontawesome.com/

[29] Mark Otto, a., Jacob Thornton: Bootstrap – The most popular HTML, CSS, and JS library in the world. [online], květen 2020, [cit. 2020-05-16]. Dostupné z: https://getbootstrap.com/

Příloha **A**

Seznam použitých zkratek

AJAX Asynchronous JavaScript And XML

AS Alternativní scénář

ČVUT České vysoké učení technické

DRY Do not repeat yourself

GUCE Zobecněný prvek případu užití (z angl. „generalized use case element“)

JSON JavaScript Object Notation

Lo-Fi Nízká přesnost či věrnost (z angl. „low fidelity“)

UC Případ užití (z angl. „use case“)

Příloha **B**

Pokrytí požadavků případy užití

	F1 – Nákupy	F2 – Požadavky nákupu	F3 – Příjem nákupu	F4 – Recepty	F5 – Šablony receptů	F6 – Výběr potravin ze skladu	F7 – Převoz potravin	F8 – Zobrazování přehledů	F9 – Přehled o stavu potravin	F10 – Počet strávníků	F11 – Správa systému	F12 – Správa účtu	F13 – Oblíbené položky	F14 – Komunikace
Výběr potraviny	✓	✓		✓		✓	✓	✓						
Výběr konkrétní potraviny	✓	✓		✓		✓	✓	✓						
Nákup														
Vytvoření	✓													
Úprava	✓													
Požadavek nákupu														
Vytvoření		✓												
Úprava		✓												
Vyřízení		✓												
Příjem nákupu			✓											
Recept														
Vytvoření				✓										
Úprava				✓										
Smazání				✓										

Pokrytí požadavků případy užití

	F1 – Nákupy	F2 – Požadavky nákupu	F3 – Příjem nákupu	F4 – Recepty	F5 – Šablony receptů	F6 – Výběr potravin ze skladu	F7 – Převoz potravin	F8 – Zobrazování přehledů	F9 – Přehled o stavu potravin	F10 – Počet strávníků	F11 – Správa systému	F12 – Správa účtu	F13 – Oblíbené položky	F14 – Komunikace
Vytvoření šablony receptu					✓									
Vytvoření receptu ze šablony					✓									
Výběr ze skladu														
Manuální						✓								
Podle receptu						✓								
Převoz potravin														
Vytvoření záznamu							✓							
Naložení							✓							
Příjem							✓							
Přehled výdajů														
Zobrazení								✓						
Vytvoření								✓						
Úprava								✓						
Smazání								✓						
Stav skladu									✓					
Kontrola množství potravin									✓					
Počet strávníků										✓				
Zobrazení historie											✓			
Správa hodnot											✓			
Správa uživatelů											✓			
Správa areálů											✓			
Pokrytí požadavků případy užití														

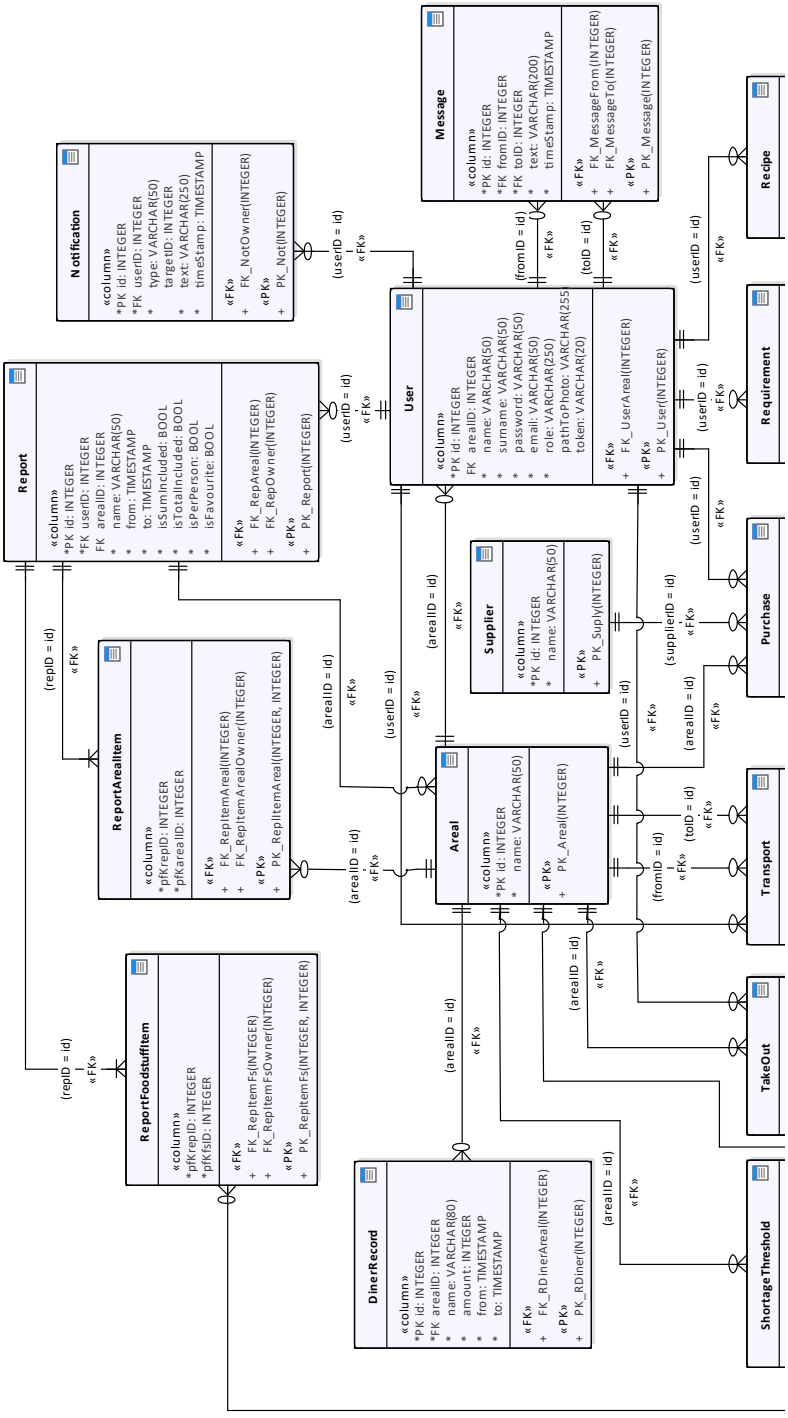
	F1 – Nákupy	F2 – Požadavky nákupu	F3 – Příjem nákupu	F4 – Recepty	F5 – Šablony receptů	F6 – Výběr potravin ze skladu	F7 – Převoz potravin	F8 – Zobrazování přehledů	F9 – Přehled o stavu potravin	F10 – Počet strávníků	F11 – Správa systému	F12 – Správa účtu	F13 – Oblíbené položky	F14 – Komunikace
Šablony receptů														
Schválení											✓			
Smazání											✓			
Přihlášení do systému												✓		
Obnova hesla												✓		
Správa účtu												✓		
Oblíbené položky													✓	
Posílání zpráv														✓

Tabulka B.1: Pokrytí požadavků případy užití.

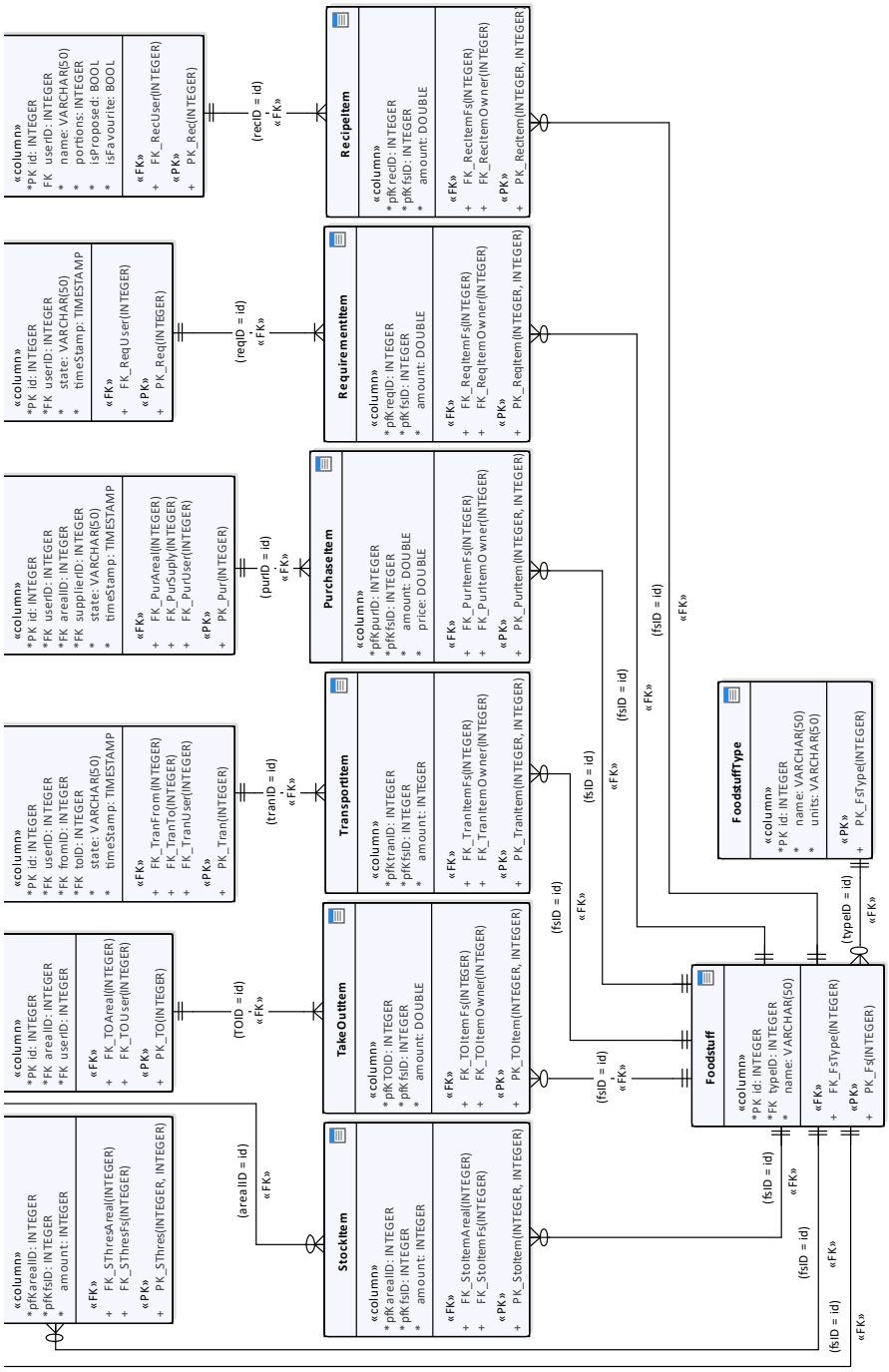
Příloha C

Kompletní databázový model

Tato kapitola obsahuje kompletní databázový model. Kvůli jeho velikosti byl však obrázek rozdělen na polovinu a pro větší pohodlí čtenáře přesunut na následující dvojstranu.



Obrázek C.1: Kompletní databázový model (první část)



Obrázek C.2: Kompletní databázový model (druhá část)

Příloha D

Scénář uživatelského testování

Na následujících řádcích se nachází scénář popisující způsob uživatelského testování systému vyvinutého v rámci této práce. Pro udržení co největší možné úrovně objektivity, a tím i vypovídající hodnoty testování, se doporučuje testujícím držet se co nejvíce textu tohoto scénáře.

Text scénáře (podobně jako tomu bývá u scénářů divadelních) je doplněn o poznámky pro testující. Tyto poznámky nejsou čteny testovaným, neboť slouží primárně k řízení průběhu testování. Pro odlišení od textu určeného ke čtení testovaným budou zvýrazněny kurzívou a umístěním do závorek.

V průběhu testování je však dovoleno (a doporučeno) odpovídat testovaným na jejich případné dotazy. V rámci odpovědí by se ovšem testující měli snažit neuhýbat k jiným tématům a co nejméně ovlivňovat průběh testování.

Zároveň je dovoleno pomoci testovaným vrátit se na úvodní stránku systému (v případě, že se ztratí) či vyřešit technické komplikace (např. pád systému). Obě zmíněné okolnosti by však měly nastávat ojediněle či by neměly nastat vůbec.

Scénář testování je vytvořen pro formální styk s testovanými. Z tohoto důvodu je v něm pro interakci s testovanými použito vykání. Pokud to ovšem testovaným vyhovuje více, je dovoleno upravit text tak, aby testující testovanému tykal.

Scénář se skládá ze čtyř částí, jimiž jsou postupně: úvod, testování z pohledu zaměstnance areálu, testování z pohledu zaměstnance obchodního oddělení a závěr. Úvod a závěr jsou ve scénáři vymezeny pouze rámcově. Jednotlivé části na sebe navazují, a není tedy možné měnit jejich pořadí.

Od testovaného systému se očekává, že bude primárně používán na noteboocích či stolních počítačích. Z tohoto důvodu by mělo testování probíhat na odpovídajících zařízeních.

D.1 Úvod

(V rámci úvodu projděte s testovanými následující body:

1. *Uvítání a poděkování za účast v testování*
2. *Seznámení s kontextem a účelem testování – diplomová práce Matěje Sháněla, zlepšení kvality vyvíjeného systému*
3. *Seznámení s požadavkem na monitorování testování a žádost o souhlas s monitorováním – nahrávání pro účely podrobnější evaluace, možnost vynechání osobních dat*
4. *Seznámení s obecným postupem testování – webová aplikace, části testování, vžití do rolí, komentování postupu, plnění úkolů, jakékoliv problémy hned hlásit*
5. *Vysvětlení, že se nemají bát dělat chyby – testujete systém, nikoliv je*
6. *Ujištění se, že testovaný pokynům rozuměl, dotazy*

Pokud testovaný souhlasil s pořizováním záznamu, zapněte nyní nahrávání. Postupujte do další části.)

D.2 Testování z pohledu zaměstnance areálu

1. Přejděte na stránku systému *www.haul.fit.cvut.cz/sklad*.
2. Přihlaste se do systému. Váš pracovní email je *rusnavit@haul.cz* a heslo k Vašemu účtu je *123*.
3. Na hlavní stránce zjistěte, co je potřeba udělat. *(Je potřeba vytvořit každodenní nákup.)*
4. Vyřiďte tento nákup. Zároveň se postarejte, aby Vám systém připomněl vytvoření podobného nákupu i zítra.
5. Vyberte ze skladu 5 kg jablek, 200 bílých rohlíků, 2 kg margarínu Ramy a 5 kg šunkového salámu. U jablek, rohlíků a šunkového salámu je jedno, jaké konkrétní zboží zvolíte. *(Pro dokončení úkolu je potřeba přidat dvě položky se šunkovým salámem.)*
6. Vydejte ze skladu potraviny potřebné k uvaření „Boloňských špaget se strouhaným sýrem“ pro počet strávníků, který je právě ve Vašem areálu. Konkrétní zboží zvolte podle libosti, olej nahraďte stejným množstvím octa.
7. Postarejte se, aby Vás systém příště upozornil, jakmile na skladě nebude alespoň 10 litrů (libovolného) oleje.
8. Naskladněte příchozí nákup od společnosti Nopek.

9. Vytvořte požadavek pro zaměstnance obchodního oddělení na doplnění potravin, jež ve Vašem areálu chybí. Pokud u chybějících potravin není specifikováno konkrétní zboží, nevolte ho ani v požadavku. Množství potravin zvolte tak, aby se po jeho naskladnění již nezobrazovalo upozornění, že dané potraviny chybí.

10. Odhlaste se.

D.3 Testování z pohledu zaměstnance obchodního oddělení

1. Přihlaste se do systému. Váš pracovní email je *neprosim@haul.cz* a heslo k Vašemu účtu je *heslo (Zdůrazněte, že je to všechno malými písmeny).*

2. Vyřiďte požadavek Víta Rusňáka z areálu Lopatov. Cenu potravin odhadněte, konkrétní zboží zvolte dle libosti. Mletou papriku nakupte od dodavatele Zelenina Brázda, zbytek položek nakupte v Makru.

3. Zjistěte, který ze zaměstnanců vytvořil návrh na šablonu a jak se navržený recept jmenuje. *(Vít Rusňák, Vzduch omaštěný větrem)*

4. Následně zjistěte, pro kolik je lidí je nastavený a jaké obsahuje potraviny. Nakonec žádost na základě svého uvážení schvalte nebo zamítněte. *(500 lidí, neobsahuje nic)*

5. Vytvořte nový záznam o počtu ubytovaných, který začíná zítřkem, končí ve stejný den jako Adaptační kurz z Chocně a eviduje dvacet účastníků. Následně upravte původní záznam o Adaptačním kurzu z Chocně, aby končil dnešním dnem.

6. Zjistěte, zda je možné z některého z areálů dovézt 150 čokoládových králíčků Lindor a 150 kusů (libovolných) vajec do areálu Vrchlabí. Pokud ano, daný převoz do Vrchlabí zahajte.

7. Vytvořte přehled, v němž porovnáte náklady spojené s nákupem kuřecího masa a bílých rohlíků za poslední měsíc napříč areály. Při vytváření přehledu zajistěte, ať jsou výsledky vztažené vůči areálu Kocourkov a berou v potaz počet lidí ubytovaných v jednotlivých areálech v daném období.

8. Vytvořte roční přehled pro sezónu 2020. Výsledný přehled by měl obsahovat stejné údaje jako přehled pro loňskou sezónu, jenom posunutý na současný rok (od 1.6.2020 do 1.10.2020). Původní přehled musí v systému zůstat.

9. Odhlaste se.

D.4 Závěr

(V rámci závěru projděte s testovanými následující body:

1. *Požádání o vyplnění dotazníku*
2. *Názor na průběh testování – nejasnosti, sporné body, pocity*
3. *Ukázání optimálního řešení v jinak řešených částech*
4. *Poděkování za účast v testování*
5. *Dotazy*

Jakmile už testovaní nemají žádné další dotazy či připomínky, vypněte nahrávání a rozlučte se s nimi. Tímto uživatelské testování končí.)

Příloha **E**

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `app` ..... zdrojové kódy implementace
  - `text` ..... zdrojové formy souvisejících textů ve formátu LaTeX
    - `thesis` ..... zdrojová forma práce
    - `userManual` ..... zdrojová forma uživatelské příručky
    - `instManual` ..... zdrojová forma instalační příručky
- `text` ..... související texty ve formátu PDF
  - `thesis.pdf` ..... text práce
  - `userManual.pdf` ..... text uživatelské příručky
  - `installationManual.pdf` ..... text instalační příručky